Ransomware Maze

mcafee.com/blogs/other-blogs/mcafee-labs/ransomware-maze/

March 26, 2020

Alexandre Mundo

Mar 26, 2020

29 MIN READ

EXECUTIVE SUMMARY

The Maze ransomware, previously known in the community as “ChaCha ransomware”, was discovered on May the 29th 2019 by Jerome Segura[1].

The main goal of the ransomware is to crypt all files that it can in an infected system and then demand a ransom to recover the files. However, the most important characteristic of Maze is the threat that the malware authors give to the victims that, if they do not pay**, they will release the information on the Internet[2].**

This threat has not been an idle one as the files of one company were indeed released on the Internet. Even though the company sued, the damage was already done. This is a behavior increasingly observed in new ransomware[3], such as Sodinokibi, Nemty, Clop and others.

It was highlighted last year[4] how ransomware would head in this direction to obtain money from victims who may be reluctant to pay for decryption.

TELEMETRY MAP

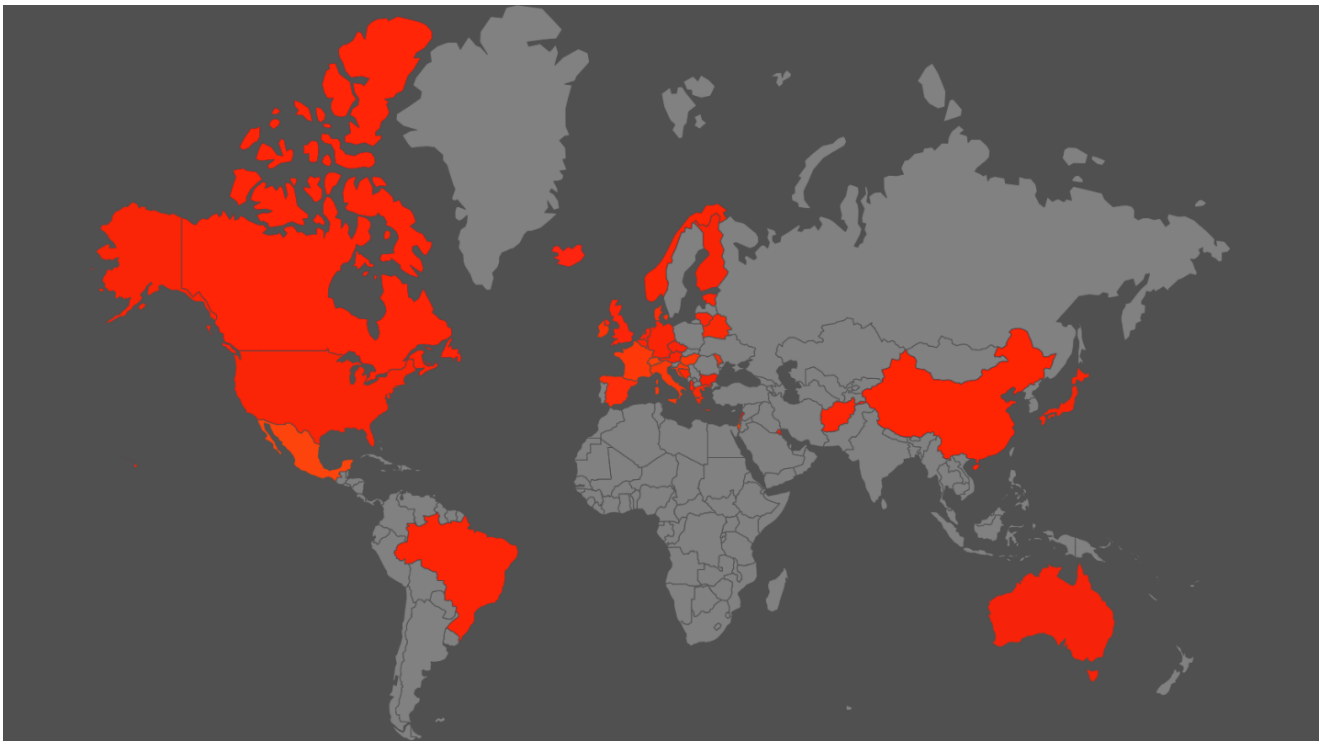


FIGURE 1. MAP OF MAZE INFECTIONS

INTRODUCTION

On the 29th of October a campaign distributing the Maze malware to Italian users was detected. Historically, the malware has used different techniques to gain entry, mainly using exploits kits, remote desktop connections with weak passwords or via email impersonation or, as in the Italian case, via different agencies or companies[5], i.e. the Italian Revenue Agency. These emails came with a Word attachment that was using macros to run the malware in the system.

The exploit kits used most often were Fallout and Spelevo[6].

The malware is hard programmed with some tricks to prevent reversing of it and to make static analysis more difficult. This report covers these protections and the behavior of the malware in an infected system.

The developers have inserted messages to provoke malware researchers, including the email address of Lawrence Abrams, owner of “BleepingComputer”, who they contacted directly. They are very active on social media sites such as Twitter.

McAfee protects its customers against the threats that we talk about in this report in all its products, including personal antivirus, endpoint and gateway.

MAZE OVERVIEW

The malware is a binary file of 32 bits, usually packed as an EXE or a DLL file. This report focuses on the EXE file.

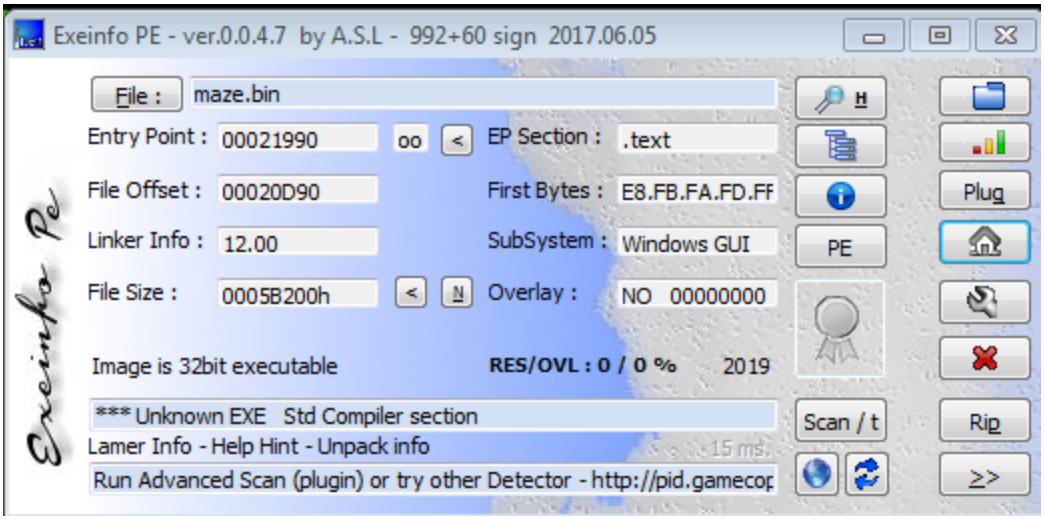


FIGURE 2. INFORMATION ABOUT THE MALWARE

More information about the sample used in this report appears in this table:

Name	maze.bin (sample name to make the analysis)
Size	373.248 bytes (can change between samples)
File-Type	EXE 32 bits (can change between samples)
SHA 256	dee863ffa251717b8e56a96e2f9f0b41b09897d3c7cb2e8159fcb0ac0783611b (changes between samples)
SHA 1	31c3f7b523e1e406d330958e28882227765c3c5e (changes between samples)
Compile time	11-01-2019 (can change between samples)

TECHNICAL DETAILS

Maze is a complex piece of malware that uses some tricks to frustrate analysis right from the beginning.

The malware starts preparing some functions that appear to save memory addresses in global variables to use later in dynamic calls though it does not actually use these functions later. Whether it is residual code existing in the entry point of the malware or a trick to mislead researchers is up for debate.

```
.text:00401490
.text:00401490
.text:00401490 MazePrepareFirstFunctionsInGlobalVarsFunction proc near
.text:00401490                                         ; CODE XREF: .text:start↓p
.text:00401490                 lea     eax, MazeDeviceIoControlFunction
.text:00401496                 mov     MazeGlobalVarForDeviceIoControlFunction, eax
.text:0040149B                 lea     eax, MazeMutexCheckAndCreateIfIsNeededFunction
.text:004014A1                 mov     MazeGlobalVarForMutexCheckAndCreateIfIsNeededFunction, eax
.text:004014A6                 retn
.text:004014A6 MazePrepareFirstFunctionsInGlobalVarsFunction endp
```

FIGURE 3. SAVE ADDRESS OF FUNCTIONS TO USE LATER IN A DYNAMIC WAY

Later, the malware enters in a big block of trash code that also includes some elements to decrypt strings and important information for later. The malware uses some tricks to detect debuggers at this point.

The most important of those are:

> A big use of the PEB field "*IsDebuggerPresent*". This field is a Boolean field that is filled from Windows with 1 (True) if the application is running inside of a debugger or 0 (False) if it is not.

```
.text:00421E41                 cmp     ebx, 198h
.text:00421E47                 jnz     short loc_421E33
.text:00421E49                 mov     ecx, large fs:30h ; get PEB structure
.text:00421E50                 push    ecx
.text:00421E51                 and     eax, ecx
.text:00421E53                 or      edx, esi
.text:00421E55                 and     dh, 48h
.text:00421E58                 pop     eax
.text:00421E59                 mov     ah, [eax+2]     ; IsDebuggerPresent
.text:00421E5C                 test    ah, ah
.text:00421E5E                 jz      short loc_421E6D
```

FIGURE 4. HIGH USE OF THE "ISDEBUGGERPRESENT" PEB FIELD TO DETERMINE IF THE APPLICATION IS RUNNING IN A DEBUGGER

If the malware detects a debugger it will remain in an infinite loop without making anything while wasting system resources.

c9ea6430da4...	50	788 KB	c9ea6430...
cmd.exe *32	00	108 KB	Procesad...
csrss.exe	00	2.088 KB	
dwm.exe	00	1.468 KB	Administr...
explorer.exe	00	53.592 KB	Explorado...

FIGURE 5. MAZE CATCHES THE DEBUGGER AND REMAINS RUNNING, WASTING RESOURCES

The malware gets all processes in the system but ignores the first one (the ‘idle process’ in Windows which is simply a tool to let the user know what percentage of system resources are being used). Using the name of each process it makes a custom name with a custom algorithm, along with a hash that is checked against a hardcoded list. If the hash is found in this list the process will be terminated.

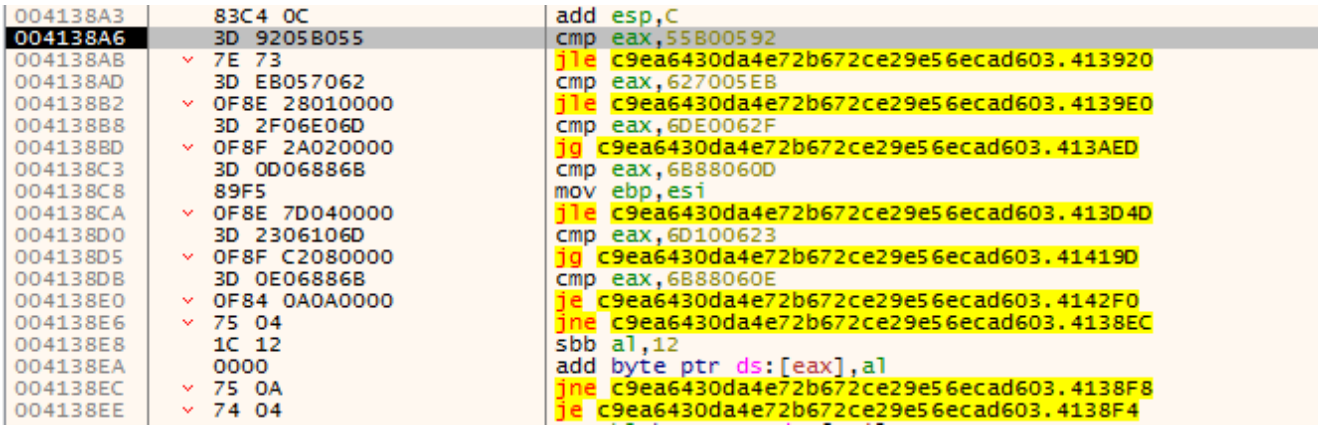


FIGURE 6. CHECK OF HASHES FROM CUSTOM NAME OF THE PROCESSES OF THE SYSTEM

For example, the process of the debugger “x32dbg”, is caught at this point:

FIGURE 7. X32DBG PROCESS CAUGHT BY THE MALWARE WITH THE HASH

It can terminate IDA debugger, x32dbg, OllyDbg and more processes to avoid dynamic analysis, close databases, office programs and security tools.

A partial list of the processes that can be cracked using a dictionary list terminated by the malware is shown below:

dumpcap.exe -> 0x5fb805c5
excel.exe -> 0x48780528
fiddler.exe -> 0x5e0c05b1
msaccess.exe -> 0x6a9c05ff
mysqld-nt.exe -> 0x79ec0661
outlook.exe -> 0x615605dc
pipanel.exe -> 0x5fb805c4
procexp64.exe -> 0x78020640
procexp.exe -> 0x606805d4
procmon64.exe -> 0x776e0635

procmon.exe -> 0x600005c9
python.exe -> 0x55ee0597
taskkill.exe -> 0x6c2e0614
visio.exe -> 0x49780539
winword.exe -> 0x60d805d5
x32dbg.exe -> 0x5062053b
x64dbg.exe -> 0x50dc0542

This short list shows the name of the process to kill and the custom hash from the special name generated from the original process name.

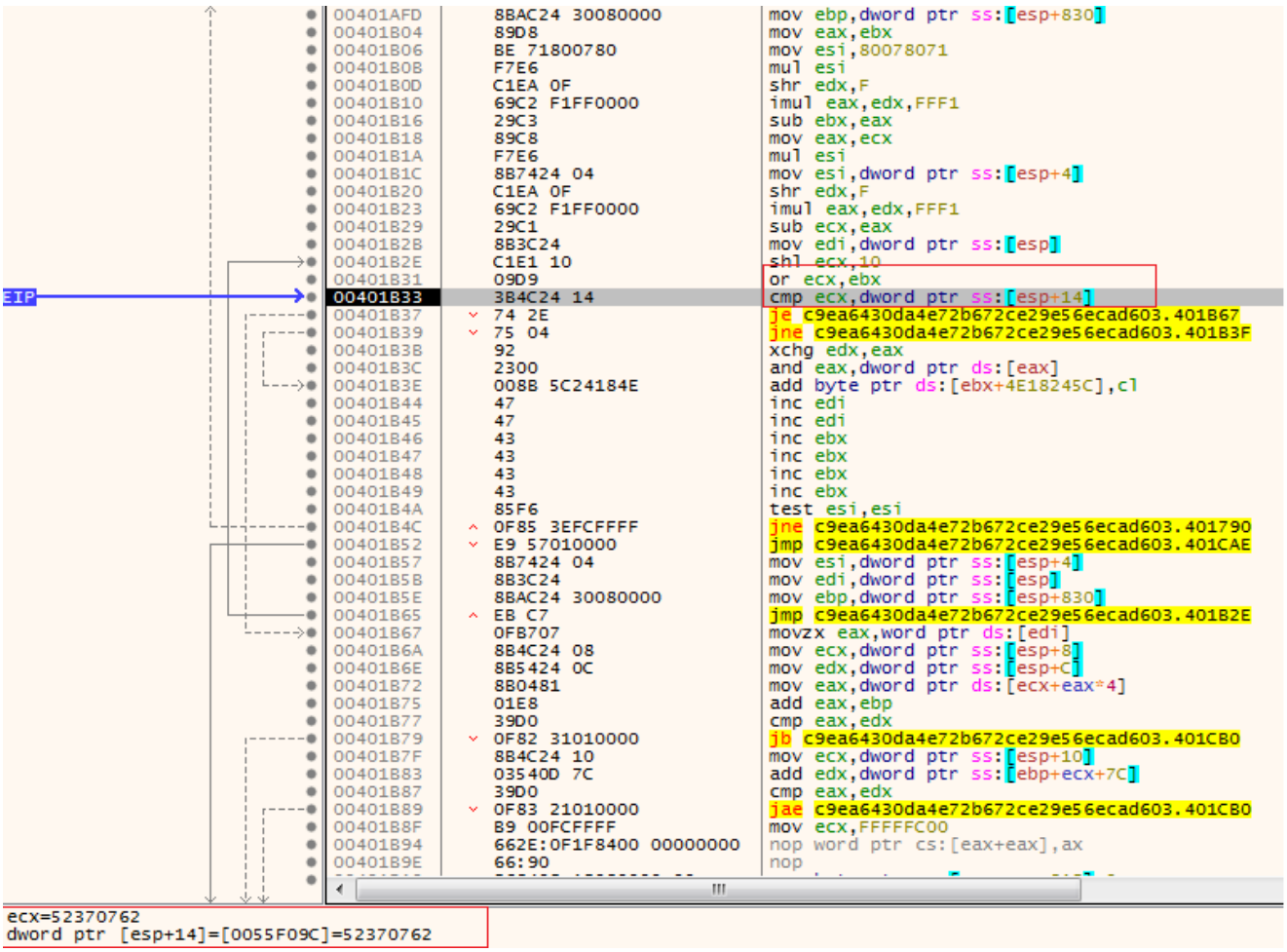


FIGURE 8. TERMINATEPROCESS FUNCTION TAKEN FROM THE EXPORT ADDRESS TABLE (EAT) OF KERNEL32 AND PASSING THE HASH NAME CHECK

The malware will kill the process with the function “TerminateProcess” that it gets from the EAT (Export Address Table) of the module “kernel32.dll” to increase obfuscation, comparing the name with a custom hash taken from the name in high caps.

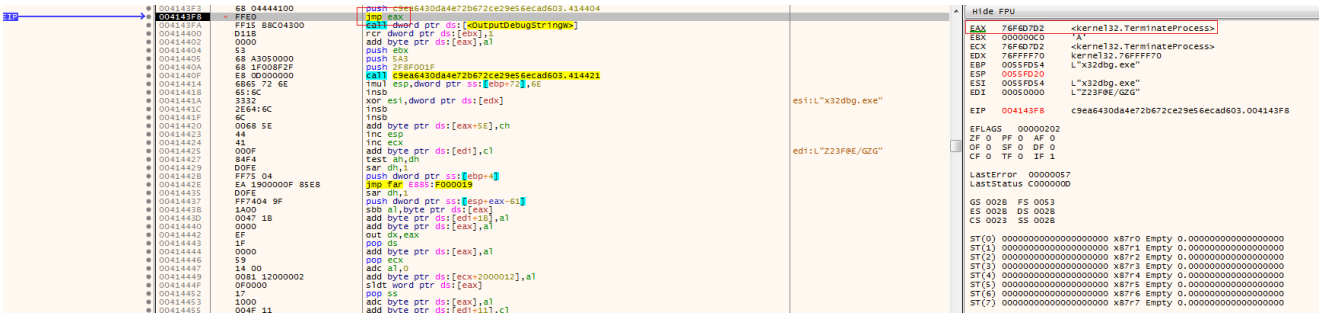


FIGURE 9. CALL TO TERMINATEPROCESS IN A DYNAMIC WAY TO OBFUSCATE THIS CALL

The malware calls Windows functions in a unique way to aid obfuscation, i.e. getting the first process in the system to use the function “Process32FirstW”. However, instead of calling it directly, it puts the parameters needed for the function on the stack, followed by a memory address with a “push” opcode and then makes a direct jump to the Windows function. When the function ends, Windows makes a “ret” opcode then gets the last memory address that the malware pushed inside the stack, returning to this address and continuing the flow. An example of this can be seen in this image:

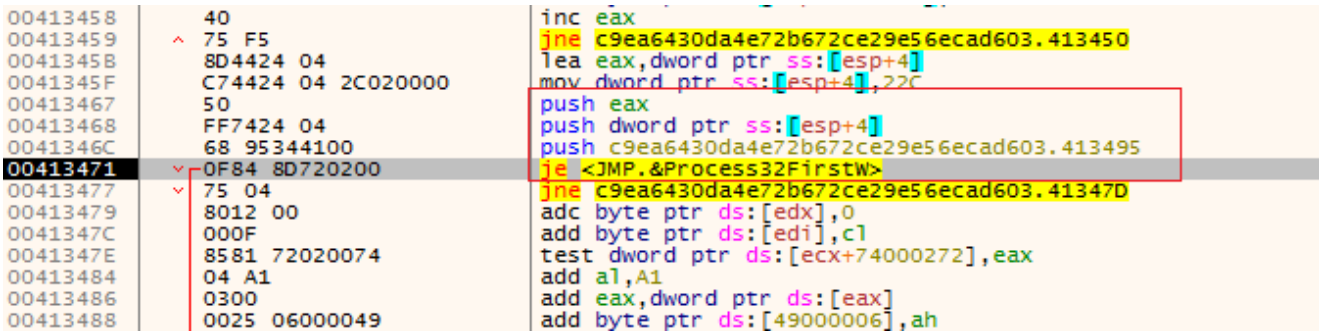


FIGURE 10. HIGH OBFUSCATION TO TRY TO SLOW ANALYSIS AND MAKE IT MORE DIFFICULT

Another ploy utilized by the malware (depending of the sample) is to get the function “DbgUIRemoteBreakin”, using the function “GetProcAddress”, before employing a trick to avoid having a debugger attach to it in runtime[7].

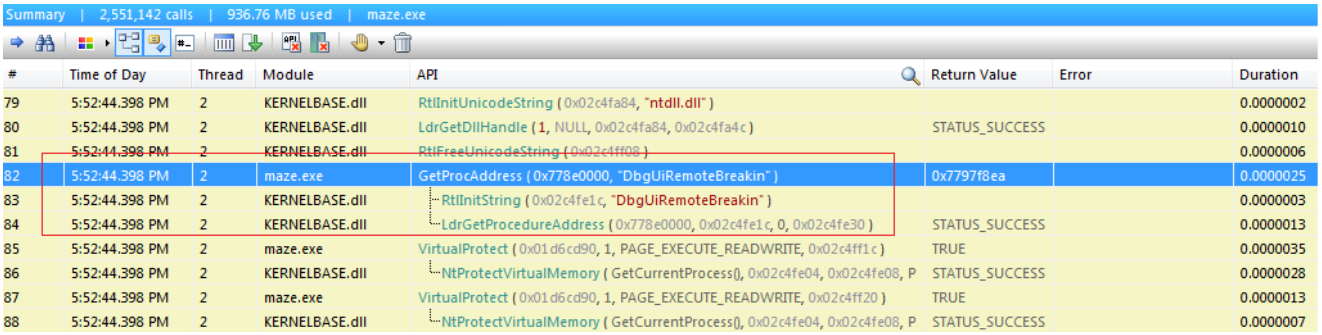


FIGURE 11. GET DBGUIREMOTEBREAKIN USING GETPROCADDRESS TO AVOID HAVING A DEBUGGER ATTACK IT

The trick used here is “VirtualProtect” to give the function memory address of “DbgUIRemoteBreakin” permission to write to it:

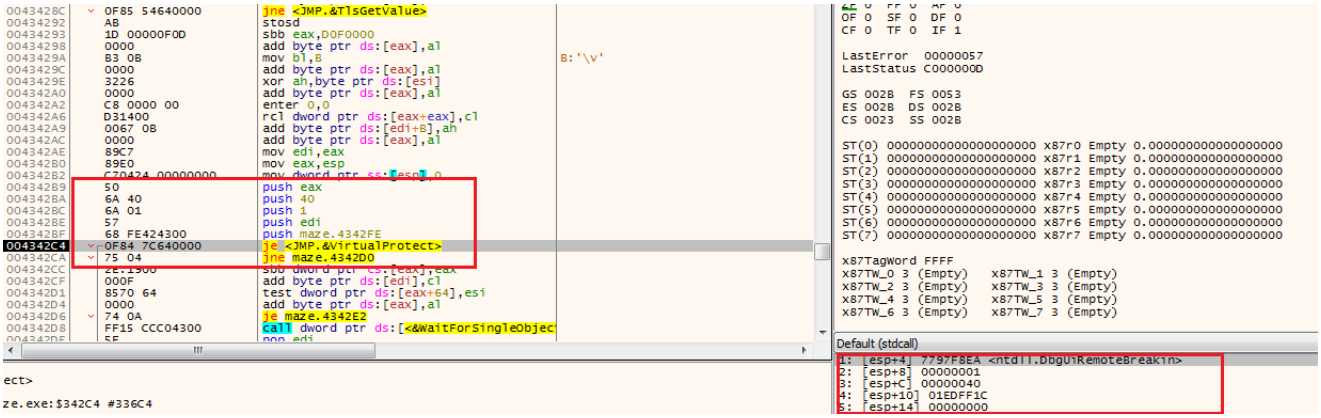


FIGURE 12. GIVE WRITE PERMISSIONS IN MEMORY

After gaining permission, which is granted only for 1 byte, the malware patches this byte with a 0xC3 value (the opcode of “ret”) and restores the previous permissions with “VirtualProtect”, again in the same address and byte, removing the write permission.

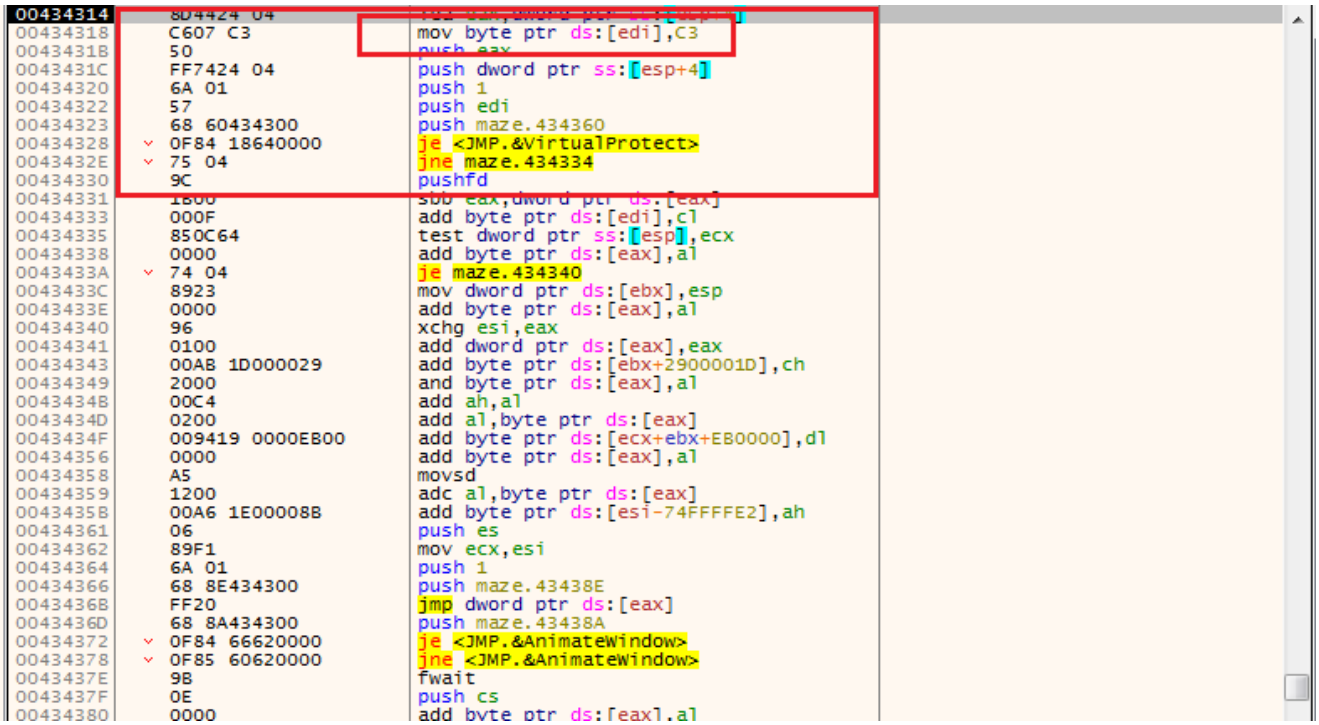


FIGURE 13. PATCH THE FUNCTION WITH A RET OPCODE AND RESTORE MEMORY PERMISSIONS

This is done to avoid having a debugger attach to it in runtime. This way, when a debugger attaches to the process internally, the system calls this function but, instead of creating a thread to start the debugging, the “ret” opcode forces the function to return without creating it. In brief, it prevents a debugger from being attached correctly. It is done before enumerating the system process.

The malware checks the language of the machine with function “GetUserDefaultUILanguage” and saves the value in the stack; it is not checked automatically after the call, but it is important later.

Maze creates a mutex with the name “Global\x” where x is a special value that is unique per machine. For example, in the next screenshot (some information has been deleted to anonymize the machine used for the analysis) is an example of this behavior. It is done to avoid two or more executions at the same time.

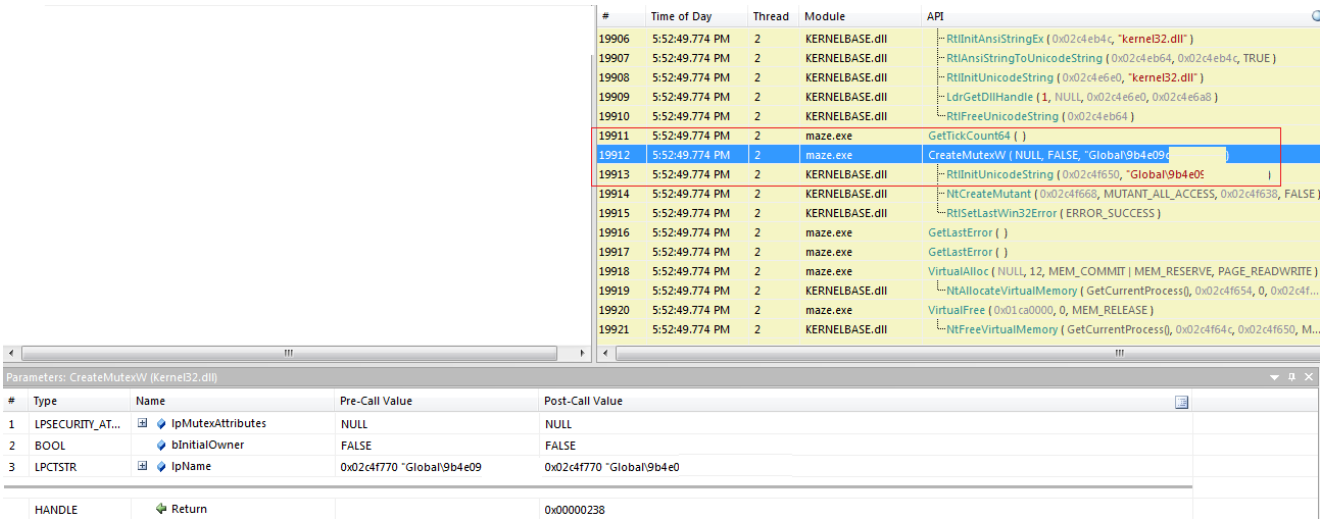


FIGURE 14. CREATION OF A MUTEX TO AVOID DOUBLE EXECUTION. UNIQUE PER MACHINE

The malware, after creating the mutex, makes calls to the function “GetLastError” to check against two errors:

- 0x05 -> ERROR_ACCESS_DENIED. If the malware gets this error, it means that the mutex already exists in the system but, for some reason, the malware cannot access it (perhaps privileges, policies, etcetera).
- 0xb7 -> ERROR_ALREADY_EXISTS. If the malware gets this error, it means that the mutex already exists in the system and can be accessed.

If either of the above occur, the malware remains in execution but does not crypt any files in the system or use any resources of the machine. It means that it will appear in the program list using 0% of the processor.

The mutex value changes either per sample or on a periodic basis to avoid the possibility of vaccines being made against it. The malware also has a command to avoid the 'problem' of vaccines which will be explained later.

After the mutex, the malware checks the language previously saved in the stack against, for example, language 0x419 (Russian from the Russian Federation, ru-RU[8]).

The checks are done in an obfuscated way within the jumble of the code that the malware has (in the virtual machine used here the Spanish language of Spain (es-ES) was used; it is the code 0xC0A that appears in the stack in the screenshot):

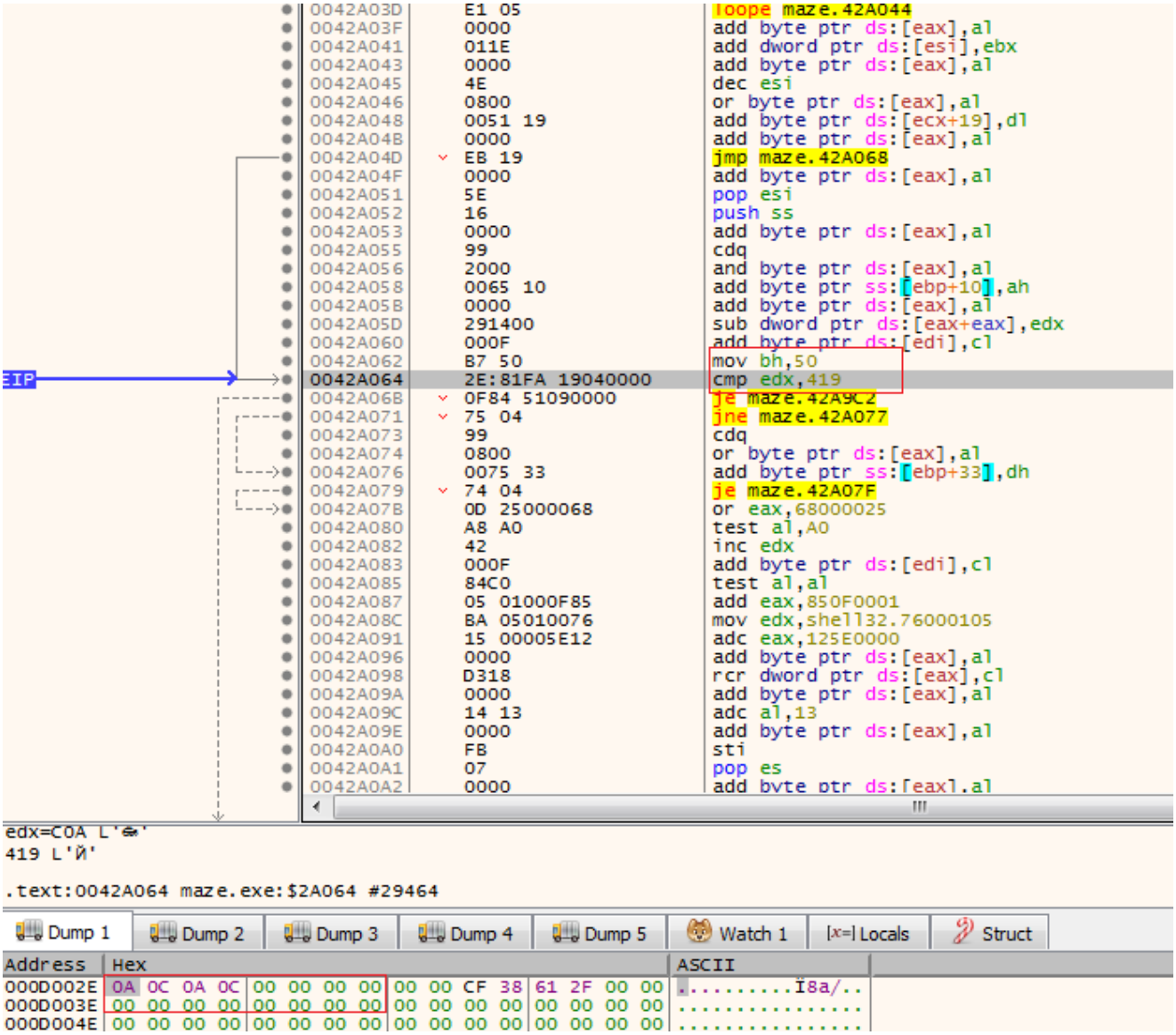


FIGURE 15. CHECKING THE LANGUAGE AGAINST THE RUSSIAN LANGUAGE FROM THE RUSSIAN FEDERATION

If the language matches any of those in the list below, the malware will clean the memory and exit the main thread without wasting any resources or making any files.

- 0x419 -> ru-RU (Russian from Russian Federation)
- 0x422 -> uk-UA (Ukranian from Ukraine)
- 0x423 -> be-BY (Belarusian from Belarus)
- 0x428 -> tg-Cyrl-TJ (Tajik (Cyrilic from Tajikistan)
- 0x42B -> hy-AM (Armenian from Armenia)
- 0x42C -> az-Latn-AZ (Azerbaijani (Latin from Azerbaijan))
- 0x437 -> ka-GE (Georgian from Georgia)
- 0x43F -> kk-KZ (Kazakh from Kazakhastan)
- 0x440 -> ky-KG (Kyrgyz from Kyrgyzstan)
- 0x442 -> tk-TM (Turkmen from Turkmenistan)
- 0x443 -> uz-Latn-UZ (Uzbek (Latin from Uzbekistan))
- 0x444 -> tt-RU (Tatar from Russia Federation)
- 0x818 -> ro-MD (Romanian from Moldova, **NOT Romanian from Romania!**)
- 0x819 -> ru-MD (Russian from Moldova)
- 0x82C -> az-Cyrl-AZ (Azerbaijani (Cyrilic from Azerbaijan))
- 0x843 -> uz-Cyrl-UZ (Uzbek (Cyrilic from Uzbekistan))
- 0x7C1A -> sr (Serbian)
- 0x6C1A -> sr-Cyrl (Serbian in Cyrilic)
- 0x1C1A -> sr-Cyrl-BA (Serbian (Cyrilic from Bosnia and Herzegovina))
- 0x281A -> sr-Cyrl-RS (Serbian (Cyrilic from Serbia))
- 0x81A -> sr-Latn-CS (Serbian (Latin)) (this language code starts from Windows Vista)

The malware tries to delete the shadow volumes in the system using the “wmic.exe” program with the switches “shadowcopy” and “delete”. Prior to this, the malware gets the function of “WoW64DisableWow64FsRedirection” with “GetProcAddress” and uses it to avoid redirection by default in 64-bit operating systems and calls it in a dynamic way.

The malware tries to delete the shadow copies two times, once before crypting the files in the infected system and secondly after crypting them.

This execution is done with the function “CreateProcessW” but, to increase the level of obfuscation, the malware is launched with this command:

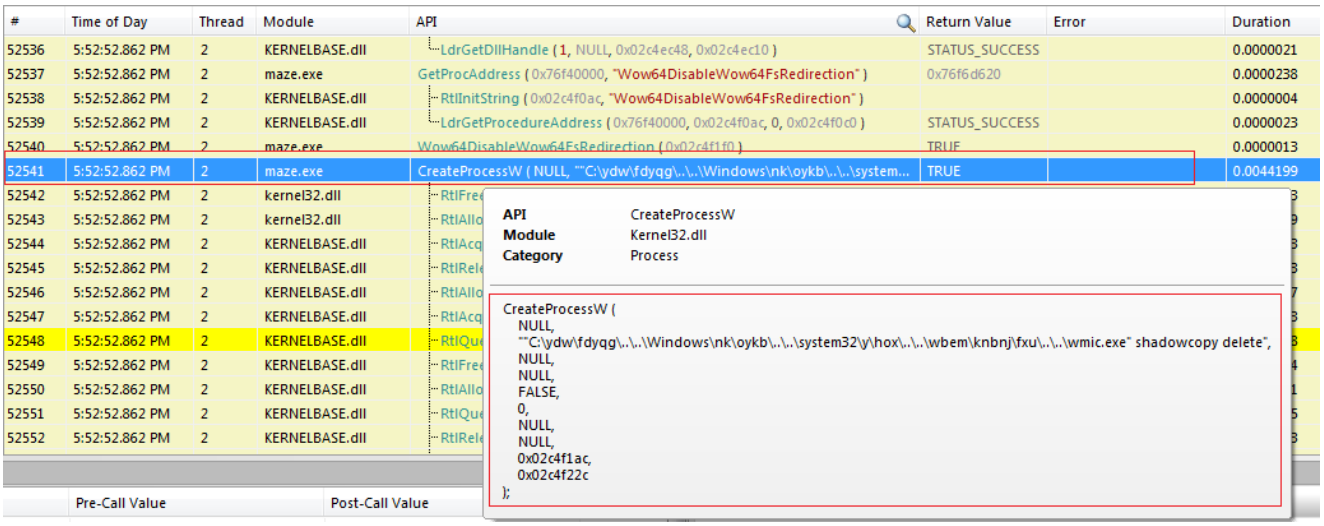


FIGURE 16. DELETION OF SHADOW COPIES IN THE INFECTED SYSTEM WITH THE WMIC COMMAND

As you can see in the image above, the malware uses a command with the name of folders that do not exist by default in Windows, except “Windows”, “system32” and “wbem”. It enters these folders but then promptly exits them using the command “..”, meaning it returns to the previous folder in the path.

For example, in the beginning it enters the folders “ydw” and “fdygg” but later returns to the root of the Windows installation unit with two “..” commands that lead to “C:\” in this case. It later concatenates with the “Windows” folder and continues with the same behavior to finally enter into “system32” where it calls the “wmic.exe” program with the switches to delete the shadow volumes. This is done to try obfuscating this call, though such suspicious behavior may cause an antivirus program to stop it anyway, but it is proof that the malware coders have skills in programming and a good understanding of Windows behavior.

It is important to understand that this “path” used in the command with non-existent folders is random and does not need to use the same number of folders to make the obfuscation.

After the deletion process, the malware gets the function “Wow64RevertWow64FsRedirection” using the function “GetProcAddress” and calls it in a dynamic way to leave the system in the same state as before.

52683	5:52:52.862 PM	2	KERNELBASE.dll	NtClose (0x000003c0)	STATUS_SUCCESS		0.0000009
52684	5:52:52.862 PM	2	maze.exe	CloseHandle (0x000003bc)	TRUE		0.0000013
52685	5:52:52.862 PM	2	KERNELBASE.dll	NtClose (0x000003bc)	STATUS_SUCCESS		0.0000007
52686	5:52:52.862 PM	2	maze.exe	GetModuleHandleW ("kernel32")	0x76f40000		0.0000037
52687	5:52:52.862 PM	2	KERNELBASE.dll	RtlInitUnicodeString (0x02c4ec48, "kernel32")			0.0000003
52688	5:52:52.862 PM	2	KERNELBASE.dll	LdrGetDllHandle (1, NULL, 0x02c4ec48, 0x02c4ec10)	STATUS_SUCCESS		0.0000021
52689	5:52:52.862 PM	2	maze.exe	GetProcAddress (0x76f40000, "Wow64RevertWow64FsRedirection")	0x76f6d638		0.0000023
52690	5:52:52.862 PM	2	KERNELBASE.dll	RtlInitString (0x02c4f0ac, "Wow64RevertWow64FsRedirection")			0.0000003
52691	5:52:52.862 PM	2	KERNELBASE.dll	LdrGetProcedureAddress (0x76f40000, 0x02c4f0ac, 0, 0x02c4f0c0)	STATUS_SUCCESS		0.0000010
52692	5:52:52.862 PM	2	maze.exe	Wow64RevertWow64FsRedirection (NULL)	TRUE		0.0000006
52693	5:52:52.862 PM	2	maze.exe	VirtualFree (0x03560000, 0, MEM_RELEASE)	TRUE		0.0000041
52694	5:52:52.862 PM	2	KERNELBASE.dll	NtFreeVirtualMemory (GetCurrentProcess(), 0x02c4f098, 0x02c4f09c, M...	STATUS_SUCCESS		0.0000033
52695	5:52:52.862 PM	2	maze.exe	VirtualFree (0x01f00000, 0, MEM_RELEASE)	TRUE		0.0000025
52696	5:52:52.862 PM	2	KERNELBASE.dll	NtFreeVirtualMemory (GetCurrentProcess(), 0x02c4f098, 0x02c4f09c, M...	STATUS_SUCCESS		0.0000019
52697	5:52:52.862 PM	2	maze.exe	GetTickCount ( )	783547781		0.0000003
52698	5:52:52.862 PM	2	maze.exe	GetModuleHandleA ("kernel32.dll")	0x76f40000		0.0000111
52699	5:52:52.885 PM	2	KERNELBASE.dll	RtlInitAnsiStringEx (0x02c4f0bc, "kernel32.dll")	STATUS_SUCCESS		0.0000005

FIGURE 17. RECOVER THE FS REDIRECTION IN 64-BIT OPERATING SYSTEMS

Maze affects network resources too, using the functions “WNetOpenEnumW”, “WNetEnumResourceW”, “WNetCloseEnum” and “WNetAddConnection2W”.

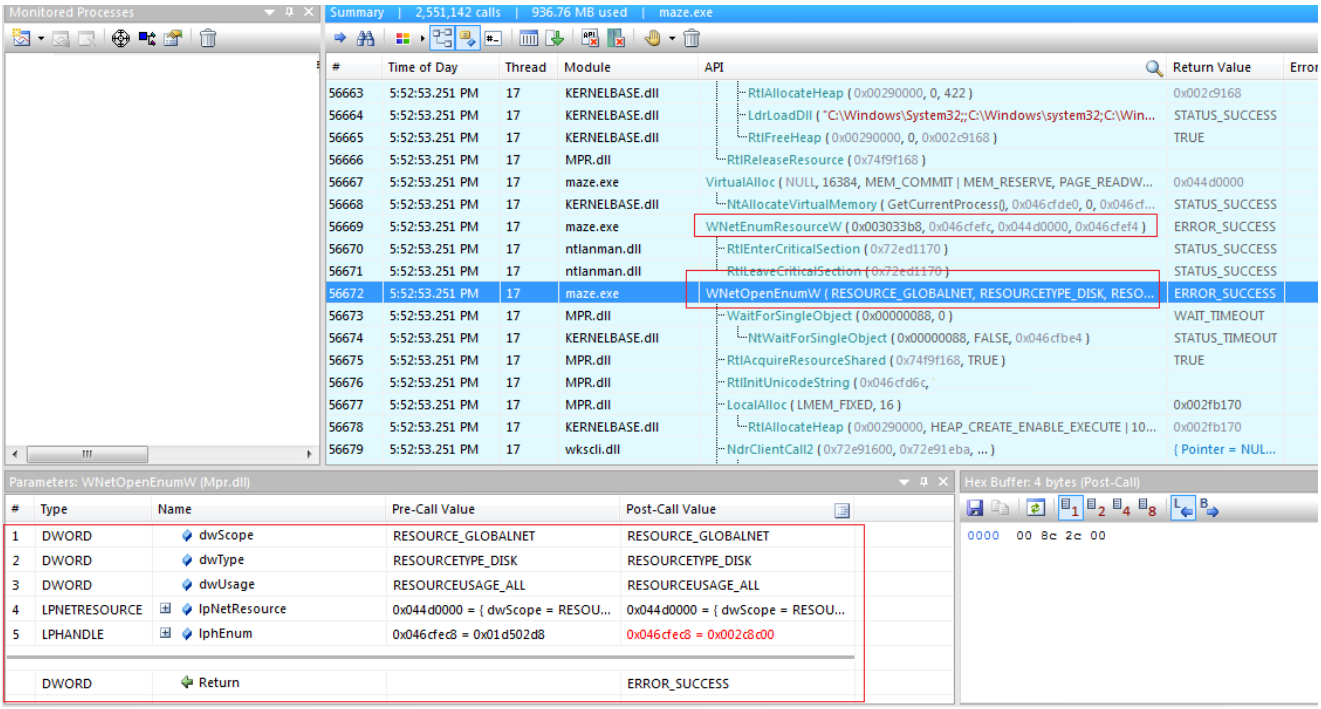


FIGURE 18. ENUMERATING THE NETWORK RESOURCES OF THE DISK TO CRYPT FILES INSIDE OF THEM

The malware uses two algorithms to crypt the files, ChaCha which is based on the Salsa20 algorithm that is symmetric and, for protection, an RSA algorithm that is asymmetric

In each execution the malware creates a Public BLOB of one RSA key that will be used to crypt the part that holds the information to decrypt the files, and one Private BLOB with an RSA key that allows decryption of the information crypted with the public RSA blob created previously.

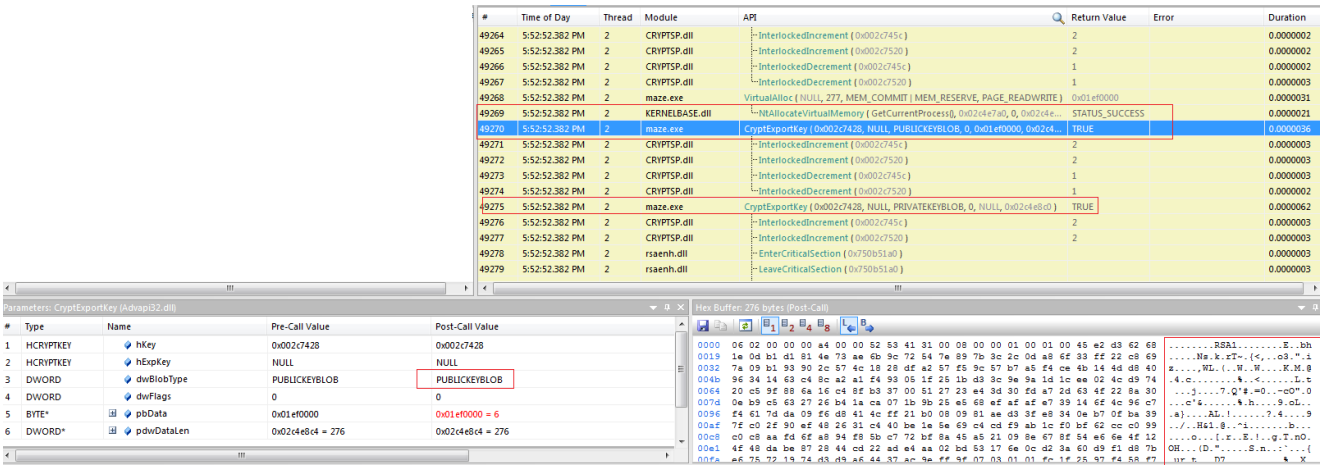


FIGURE 19. EXPORT OF THE RSA PUBLIC KEY BLOB GENERATED IN RUNTIME

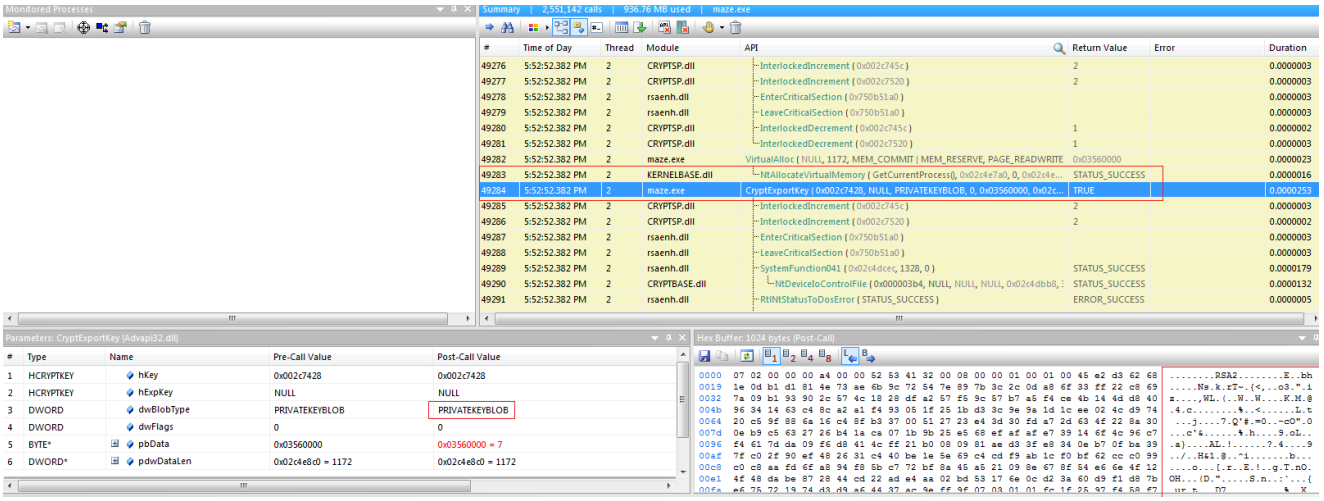


FIGURE 20. EXPORT OF THE RSA PRIVATE KEY BLOB GENERATED IN RUNTIME

Just like other ransomware, this malware has an RSA Public BLOB embedded that will be imported to protect the RSA private BLOB of the victim. Only the malware developers have the RSA private blob to decrypt their public RSA Blob.

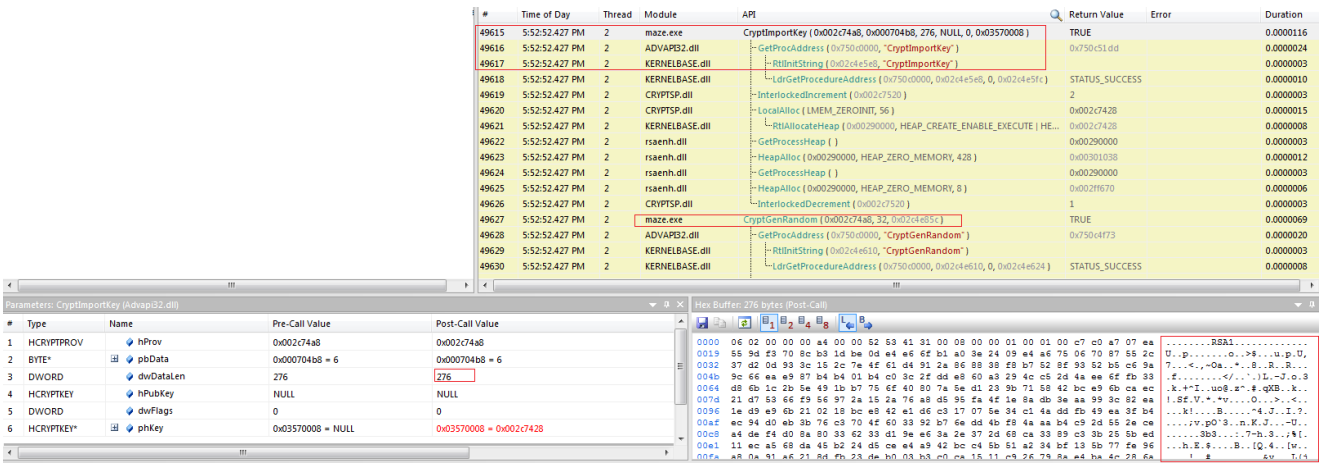


FIGURE 21. IMPORT OF THE RSA PUBLIC BLOB FOR THE MALWARE DEVELOPERS

This key is protected with a crypto using a key of 32 bits and iv of 8 bytes using the function “CryptGenRandom” to avoid memory dumps but, later, it will need to be decrypted before use.

After this, the malware starts the procedure of crypting the files, searching in units, before importing the RSA public BLOB key generated in runtime. After this, it creates the ransom note prepared for this infected machine in the root folder and then starts looking for folders and files to crypt.

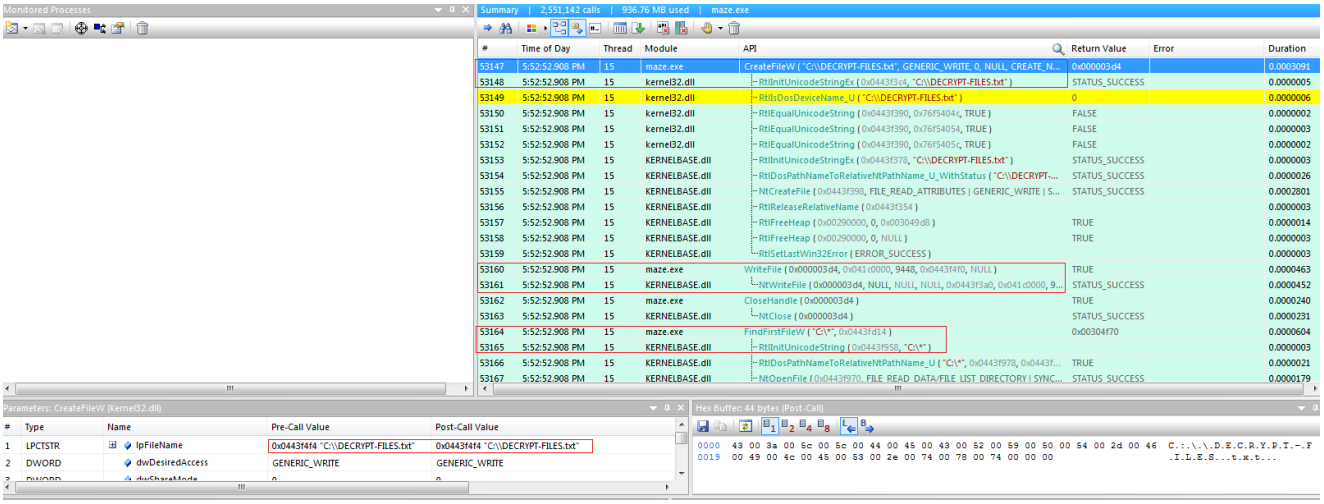


FIGURE 22. CREATION OF RANSOM NOTE IN ROOT FOLDER AND LOOKING FOR FOLDERS AND FILES

An example ransom note, with some data anonymized, is shown below:

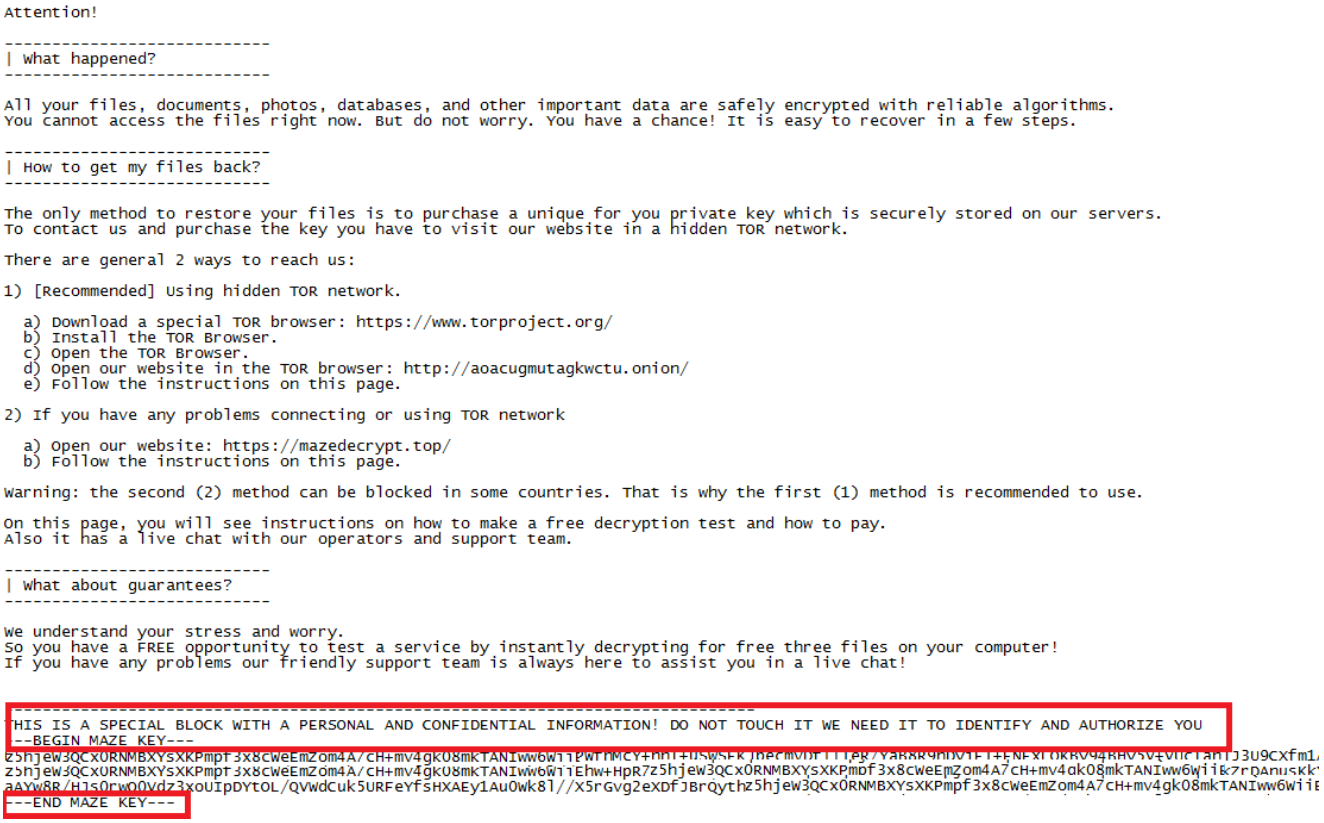


FIGURE 23. EXAMPLE OF A MAZE RANSOM NOTE

The procedure to crypt the files is easy, with the malware taking the following steps:

- Check the existence of the file with the function “SetFileAttributesW” with the attribute “FILE_ATTRIBUTE_ARCHIVE”.
- Reserve memory to the file with a call to “Virtual Alloc” for the key and iv.
- Open the file with read and write permissions with the function “CreateFileW” with the flag “OPEN_EXISTING”.
- Get the file size with the function “GetFileSizeEx” (it is important for managing big files, “GetFileSize” is not good for bigger files).
- Create a file mapping with the functions “CreateFileMappingW” and “MapViewOfFile”
- Generate a random key of 32 bytes with the function “CryptGenRandom”.
- Generate a random iv of 8 bytes with the function “CryptGenRandom”.
- Reserve 264 bytes of memory with the function “VirtualAlloc”.
- Generate a new random extension for the victim file. Each file has a different extension but does not lose the original extension; the new one is appended to the old one. For example, “1.zip” becomes “1.zip.gthf”.
- Crypt the file with the ChaCha algorithm and the key and iv with the RSA public key generated in runtime.
- Write this new block with the key and iv to decrypt at the end of the file.

- Rename the file with the function “MoveFileExW”. That way it is not possible to use forensic tools to recover the files because they use the same sector on the raw disk. The malware does not delete the file using the function “DeleteFileW” and later create a new one with the crypted data. Instead, all changes are applied in the mapping directly, in memory, without using a file pointer on the disk to read and write, which makes the process much quicker.
- The image of the file is unmapped, and handles closed.
- The process is repeated with new files.

The list of folders that the malware avoids are:

- Windows main directory.
- Games
- Tor Browser
- ProgramData
- cache2\entries
- Low\Content.IE5
- User Data\Default\Cache
- All Users
- Local Settings
- AppData\Local
- Program Files

The malware ignores these file extensions:

- LNK
- EXE
- SYS
- DLL

The malware also has a list of filenames that will not be crypted:

- inf
- ini
- ini
- dat
- db
- bak
- dat.log
- db
- bin
- DECRYPT-FILES.txt

However, it does crypt the file “ntuser.ini” to prevent other ransomwares from crypting it. It creates the ransom note in each folder that it can.

When the malware finishes crypting all files it changes the desktop wallpaper to this image:

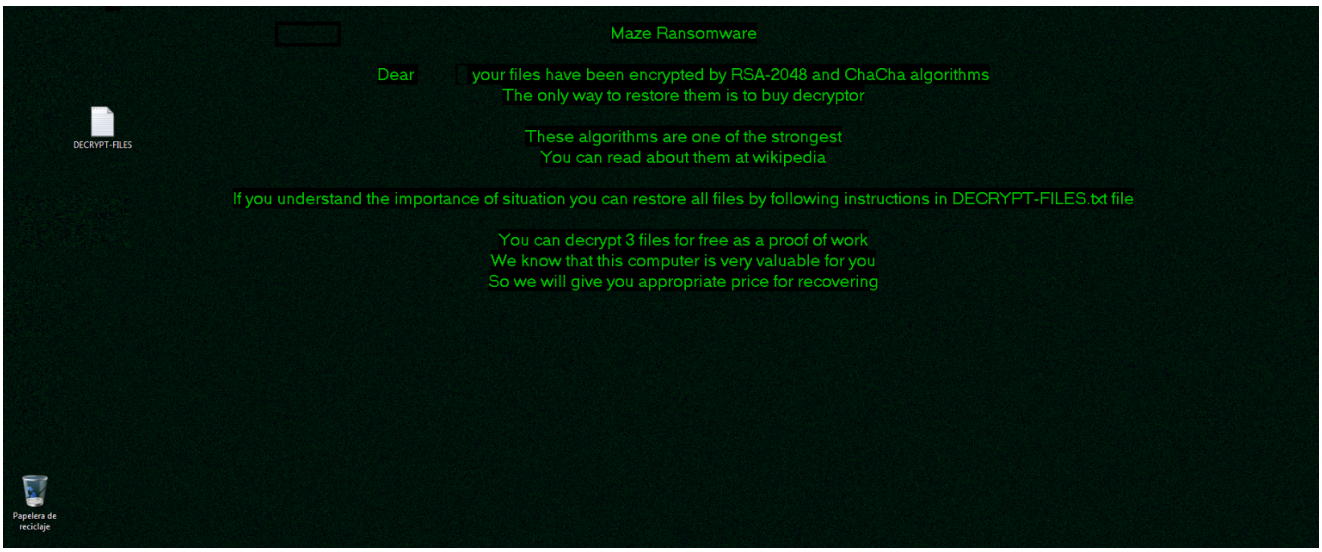


FIGURE 24. THE MALWARE CHANGES THE DESKTOP WALLPAPER AFTER CRYPTING THE FILES

The malware tries to make connections to IP addresses that have been crypted in the binary to send information about the infected machine, as seen below:

hxxp://91.218.114.4/nwjknpeevx.action?
pw=g1y652l&kyn=21y3vvhh&dvr=5e&us=g25e3582a

hxxp://91.218.114.11/forum/siaib.jspx?v=h&xyna=0vip863&eul=xsn3q0

hxxp://91.218.114.26/view/ticket/pigut.jspx?o=664quo0s&fp=ot52

hxxp://91.218.114.25/xrr.jspx?ygad=r35e2cx&e=6as6ta

hxxp://91.218.114.4/j.php

hxxp://91.218.114.11/payout/view/fa.aspx?y=y&qbx=4&kws=n2&iuy=8k7

hxxp://91.218.114.25/lxh.asp?mtxm=l7&r=836wy5

hxxp://91.218.114.26/signin/ticket/eq.action?x=yk6rr&e=50b&q=327dr5&ofk=065cdp

hxxp://91.218.114.31/signin/rnmnnekca.jsp?kdn=6snl5&e=7a50cx4hyp

hxxp://91.218.114.31/forum/a.aspx?byx=56&bc=62t0h&u=75w6n6&sot=2v0l761or6

hxxp://91.218.114.32/withdrawal/checkout/l.do?nuny=qj6&sdv=45g2boyf5q&dnr=rh8lk31ed

hxxp://91.218.114.77/task/bxfbpx.jspx?nq=cge63

hxxp://91.218.114.38/account/payout/ujwkjhoui.shtml

hxxp://91.218.114.37/imrhhjitop.phtml?wto=344dsc84&sp=x&oml=c173s71u&iy=m3u2

hxxp://91.218.114.38/auth/login

hxxp://91.218.114.79/logout/hfwdmugdi.php?upaj=mj7g

hxxp://91.218.114.38/sepa/juel.php?ars=51qse4p3y&xjaq=r5o4t4dp

hxxp://91.218.114.32/fwno.cgi?yd=410&o=y7x5kx371&p=m3361672

hxxp://91.218.114.37/sepa/signout/mjsnm.aspx?
r=7o47wri&rtew=uu8764ssy&bri=51gxx6k5&opms=72gy0a

hxxp://91.218.114.77/payout/analytics/lrkaaosp.do?y=62h&aq=3jq8k6&v=0svt

hxxp://91.218.114.79/create/dpcwk.php?u=28qy0dpmt&qwbh=k&f=g1ub5ei&ek=3ee

It is important to take into consideration that the malware forges the POST string to make the connection with a random choice from a list of possible strings such as “forum”, “php”, “view”, etc., to make detection harder with IPS or other filters on the network.

The IP addresses are detected as from the Russian Federation but that does not prove that the malware came from this country; it could be deliberate misdirection but, with the language checks of CIS countries, it certainly appears possible.

The use of IP addresses instead of domain names is to avoid DNS resolutions that can be altered or redirected to a loopback, for example using the “host” file in Windows. This makes the trace of IPs more complicated and avoids having the connection blocked.

The malware uses this agent to make the connection, but it can change between samples:

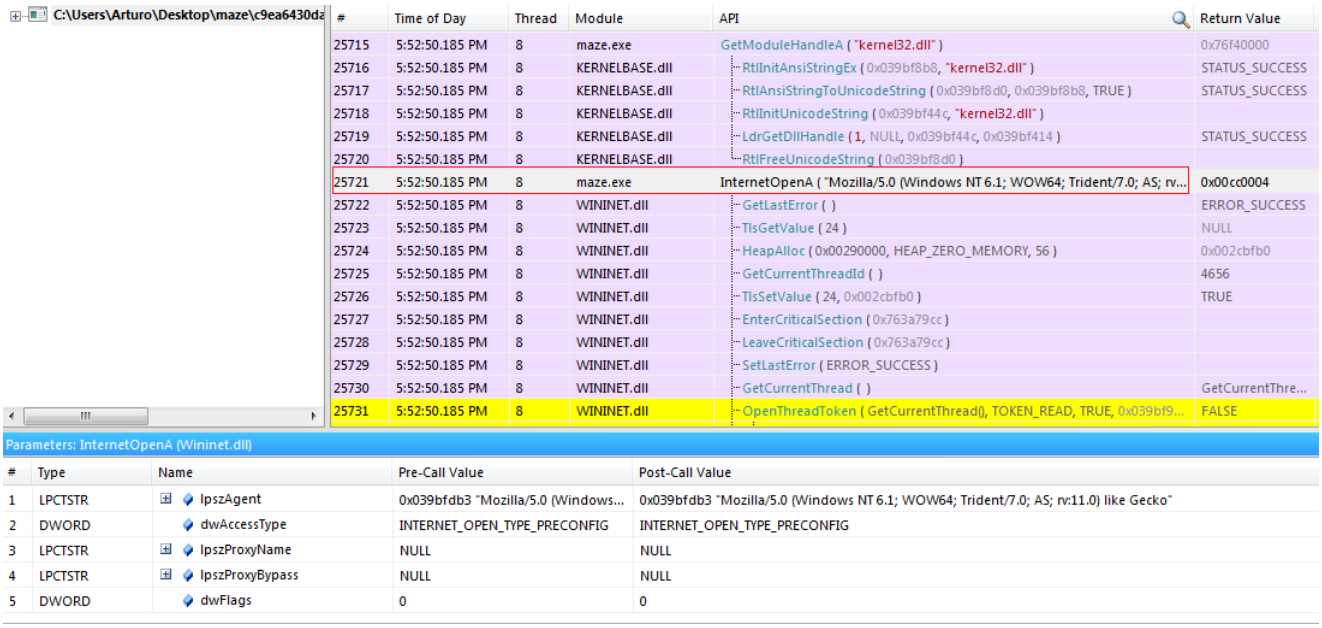


FIGURE 25. AGENT USED TO MAKE CONNECTIONS TO THE C2C IP ADDRESSES

From a memory dump we can extract the IPs used by these connections, as well as a curious string that talks about Lawrence Abrams, the admin of the web site “bleepingcomputer” who was contacted directly by the developers. It is not known why they included this email address because it has no relation to the ransom note and is not used anywhere else. Perhaps it is a means of mocking the administrator of a site that frequently reports on ransomware?

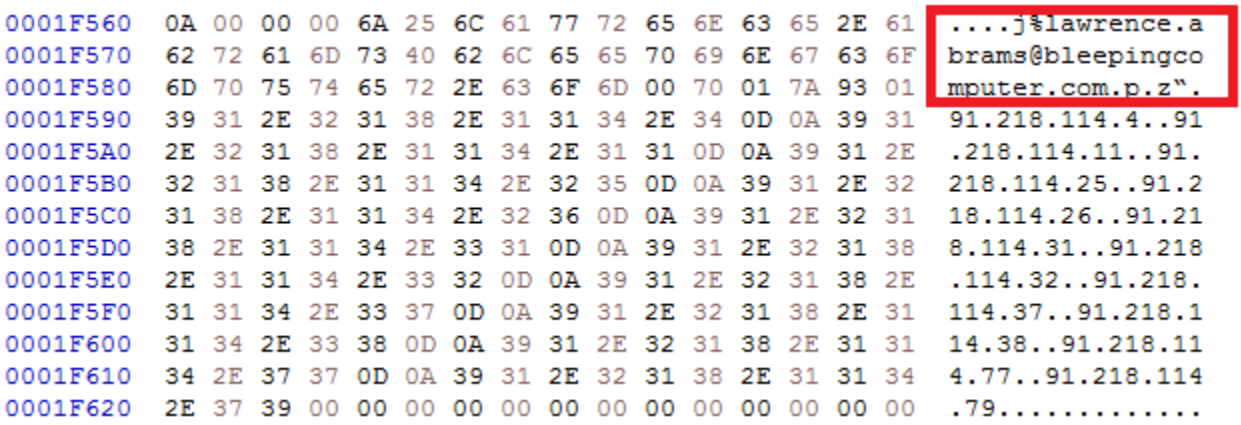


FIGURE 26. C2C IP ADDRESSES EXTRACTED FROM THE MEMORY

The connections to the C2C IP addresses, in a pcap using Wireshark, can be seen perfectly:

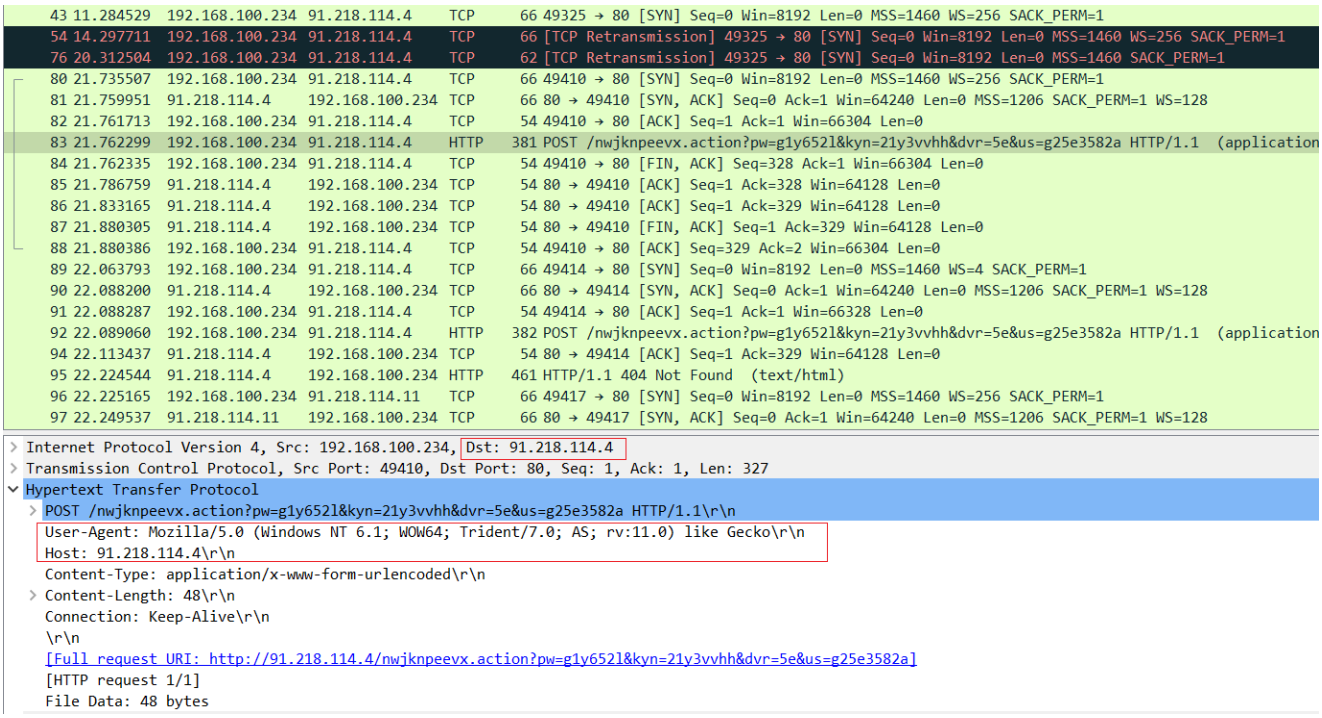


FIGURE 27. CONNECTION IN PCAP WITH THE C2C IP ADDRESSES

Maze has some strings in memory that are interesting and something that may be worth further analysis in the future:

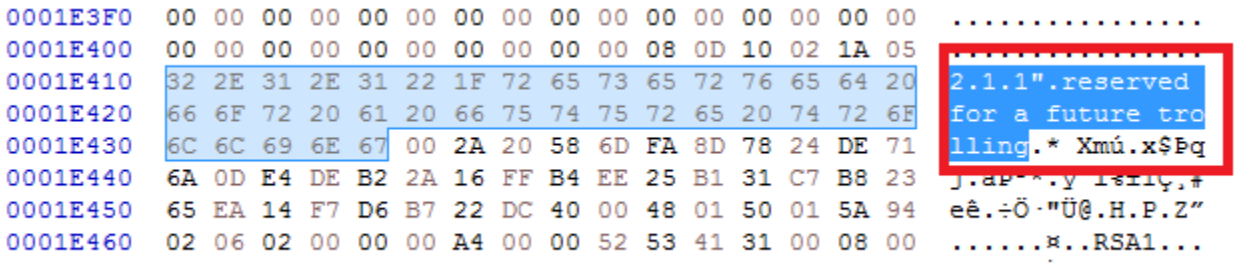


FIGURE 28. CURIOUS STRING FOR FUTURE INVESTIGATION

The webpage for making the payment requested in the ransom note gives a price and verifies that all is correct.

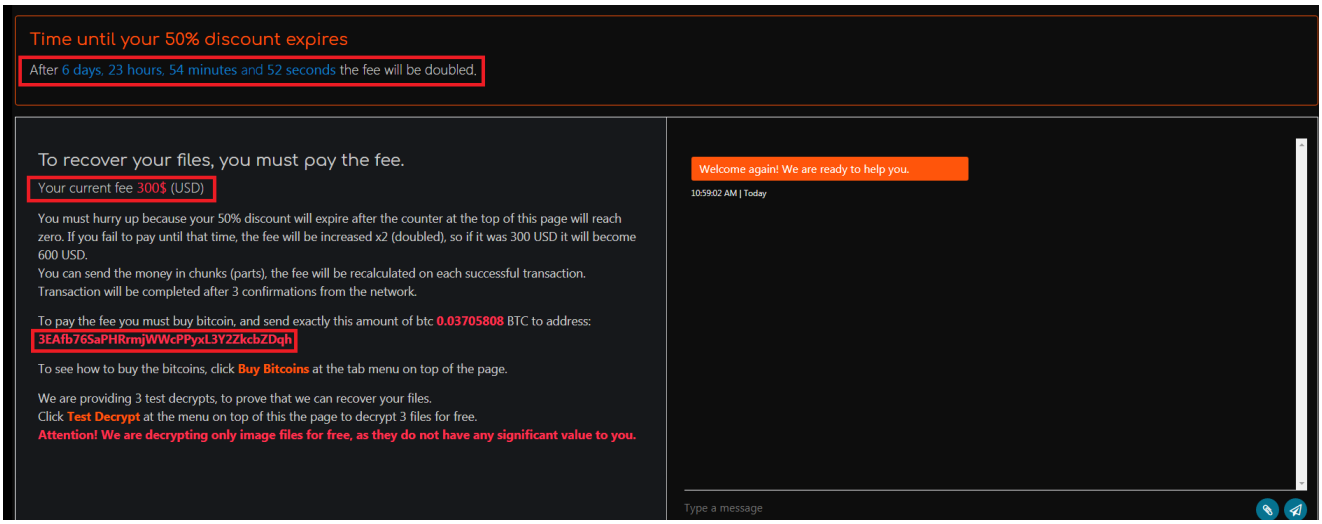


FIGURE 29. MAZE PAYMENT WEBPAGE AFTER DECRYPTING THE RANSOM NOTE

Maze has a chat function to contact the operators and receive information about how to obtain the cryptocurrency required to make payment.

Of course, as with many types of ransomware, there is an offer to decrypt three images for free and that service has been verified as working:

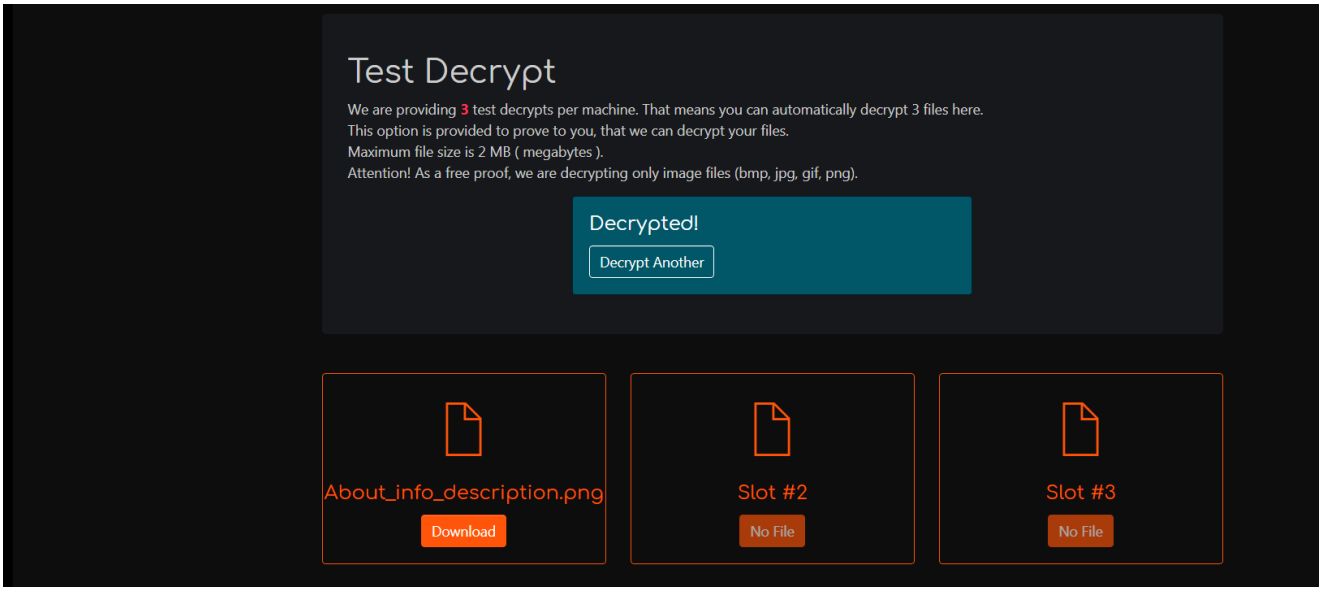


FIGURE 30. FREE DECRYPTION WORKS SO THE MALWARE SAMPLE IS CORRECT

SWITCHES

The malware has some switches that can be used in the command line to launch. These switches can either disable some elements or enable logging.

The switches are:

- –nomutex -> This switch prevents checking the mutex so that it can run more than one instance on the same machine. It can also be used to avoid vaccines that are made before the malware creates the mutex name in the machine.
- –noshares -> With this switch the malware will not crypt network shares, only the local machine.
- –path x -> Where x is a full path. In this case the malware will crypt all files in all folders starting from this path unless they are blacklisted names, extensions or folder names. This is useful for the malware developers to attack a special path instead of losing time going after a full machine and it makes the attack more targeted.
- –logging -> If this switch is enabled the malware will log all the steps it makes. Useful to the malware developers in debug environments, or in the attack phase to know that all was ok, step by step. Here is a small example of this information:

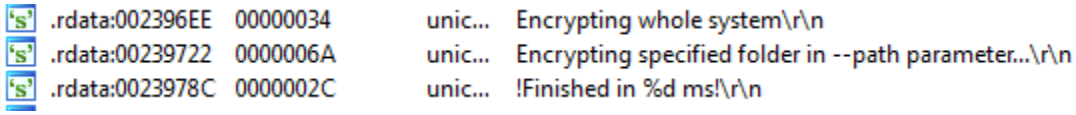


FIGURE 31. EXAMPLE OF THE INFORMATION THAT THE MALWARE CAN GIVE WITH THE LOG SWITCH

OTHER SAMPLES

In January 2020 a new version of the malware appeared with a special text dedicated to some researchers in the security field. The malware developers appear to have chosen those individuals to be provocative and make fun of them.

The sample was discovered by malwrhunterteam[9] on the 28th of January 2020. The sample has some differences when compared with the previous one that was analyzed in this report. Those differences will be covered later via another sample that was found by Luca Nagy[10] on the 30th of January 2020.

The most important thing here is that the developers appear to have carefully selected the researchers and waited for them to answer as a psychological trick, and it worked, because all of them replied, trolling the malware developers over the version of their malware detected on the 28th.

Here is one response from a malware developer to this trolling that contains some interesting facts:

```
C:\JDUIHiuf\IDisjopjcnb
@malwrhunterteam, good last discussion in twitter thread.
Answering to @MalwareTechBlog @kravietz_ @hasherezade @f0wlsec and others, It is unfortunately neither paranoia nor insulting nor marketing etc. It serves
like honeypots on shitty AV which are 90% of AVs used in enterprise (anyway they dont read your twitter as it is painful for them), who just places
signatures on data section in packer layer. It is funny to change these strings everytime and see how it is FUDing packer. Keep doing it, conspiracy theory
adepts :)
@hasherezade, I dont know why you took this as insulting, but indeed I always liked your tools. I even use some of them in my regular malware analysis.
Also, lets play some game. Write IDAPython script to deobfuscate the code of the core payload working for all samples correctly without breaking
conditional jumps and then you can write whatever you want about us, right?
Finally, literally all researcher mentions in both Maze and other malwares are neither insulting nor fan-syndrome or air conditioner. It is just like to
have some fun&games with each other, otherwise it takes to be too boring, doesnt it? FUDing sample for each target on the one side and reversing shit-
packers (literally what all Maze analysis do) on the daily basis on the other side of infosec. So Fuck infosec. Without malware your work will be boring as
hell, what will you cover? Breaches? (Oh wait...) I know you hate us, but you need to know that we love you researchers, without you our job also would be
fucking boring as hell.
--DIdsjdjdj
```

FIGURE 32. RESPONSE FROM A MALWARE DEVELOPER

- It is not known if one person is behind the malware or not. It is curious that they said “I” instead of “we” twice in their answer. So, perhaps it was written by one person for trolling purposes, or perhaps the developer of the malware really is only one person (or they want researchers to think that is the case).
- Another important fact in the note is the talk about the tools used by one of the researchers for regular malware analysis. Why are they mentioning regular malware analysis? Is it because they are reversing malware themselves for fun or could it be their day job? Could it be that perhaps the developer is a researcher (because of the way that they talk with others and provoke them)? Secondly, malware analysis is mentioned more than once and, thirdly, they said that they made an IDAPython script to remove all obfuscated code that the malware has (the ransomware may have got the name ‘Maze’ because of how analysis of it is like walking through a labyrinth). So, it may be either a researcher who knows IDAPro very well or is an advanced developer (and the obfuscated code in Maze is very well done) or perhaps it is a developer that has another job in normal life besides the creation of malware? Of course, these are just possibilities, not facts.
- The malware developer achieved their goal with this interaction as their target audience saw the answer and talked about their malware, as noted in the final line of their response “ …but you need to know that we love you researchers without you our job also would be fuc**** boring as hell”.

It is curious that here they said “we” instead of “I” as before but perhaps they were talking about all malware development?

The differences that these samples have are:

- Usually comes as a DLL instead of an EXE file. It does not run on Windows operating systems older than Vista as this makes analysis harder. By using the malware as a DLL, they can inject this module into a target process more easily than if they use an EXE sample of the malware.
- Instead of deleting the “Shadow Volumes” the developers instead use WMIC with the special trick of the path as mentioned earlier, using WMIC classes to control the Shadow Volumes. An example of this use can be seen in the next image.

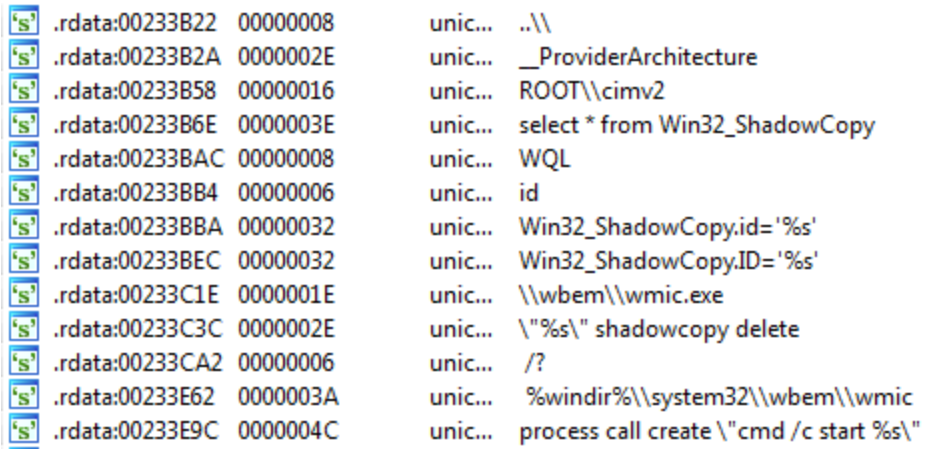


FIGURE 33. USING WMIC CLASSES IF NEEDED TO GET THE SHADOW VOLUMES

Each sample of the malware uses different strings as PDB to send messages or to make the sample unique, for example:

- C:\somerandomsh**\sh**\obama.pdb
- C:\kill\yourself\<nickname>\chinese\idio*.pdb

(In these examples some things were removed or changed to remove sensitive information in the report).

The new samples discovered in January 2020 make these connections to the C2 (or try to make them):

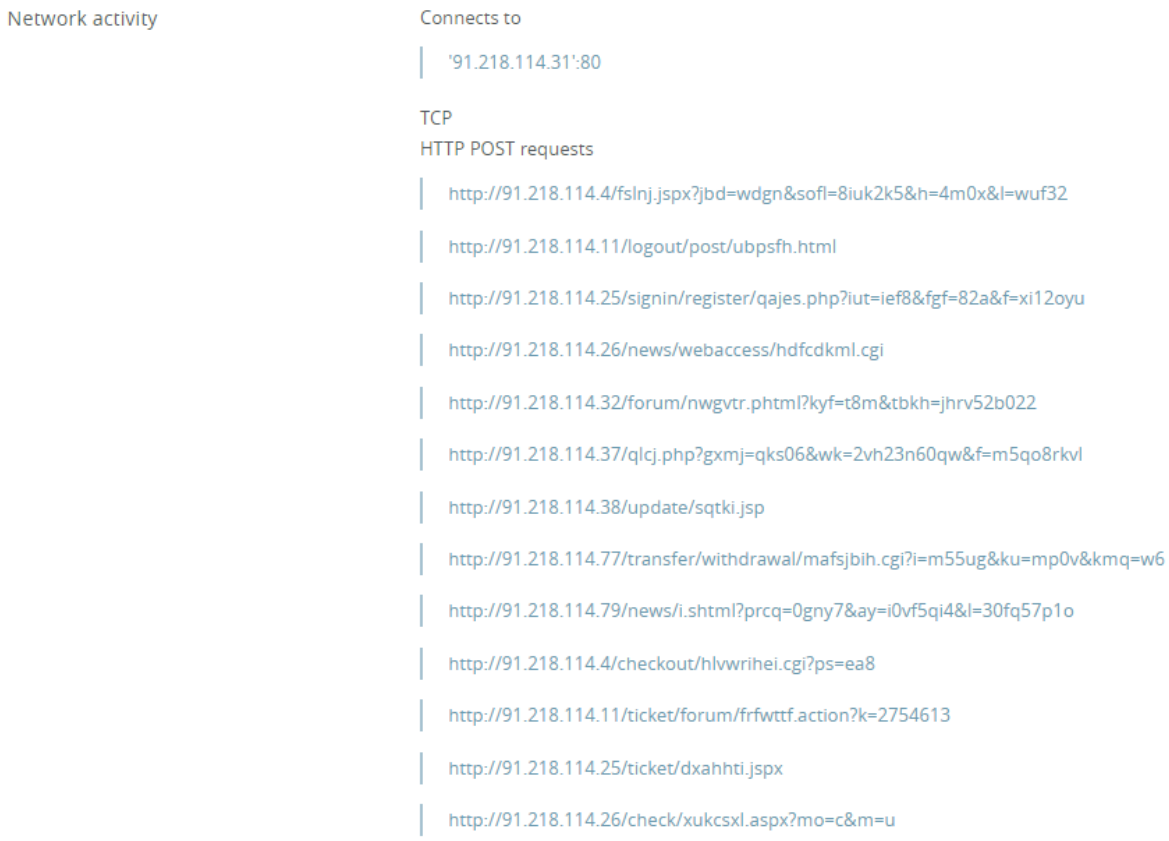
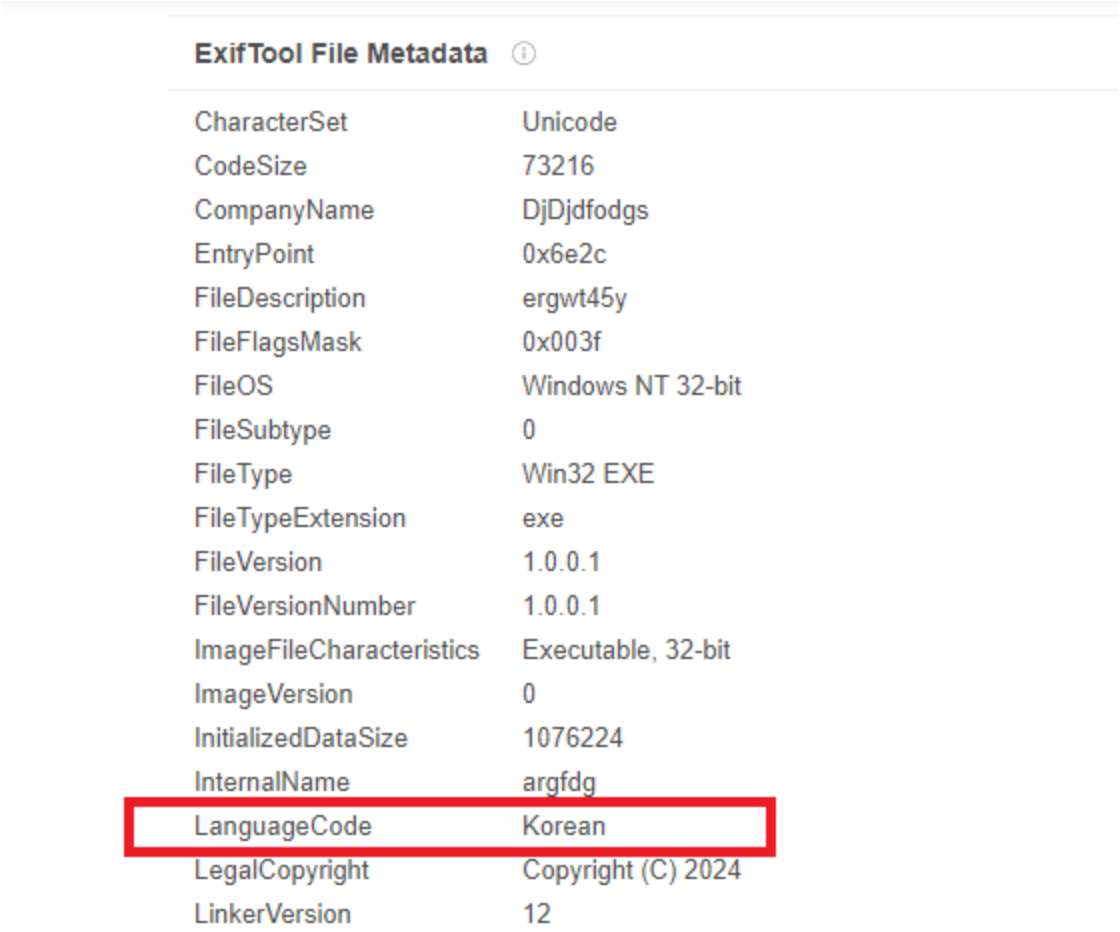


FIGURE 34. CONNECTIONS TO C2 IP OF THE NEW SAMPLES

As we can see, they are the same IPs as seen in the previous versions of the malware.

The samples' compile dates are from the 24th of January 2020 (the first version with the strings that provoked the researchers) to the 28th of January 2020 (the version with the answers to the researchers), meaning they were made on the same day the responses to the previous version were published on Twitter.

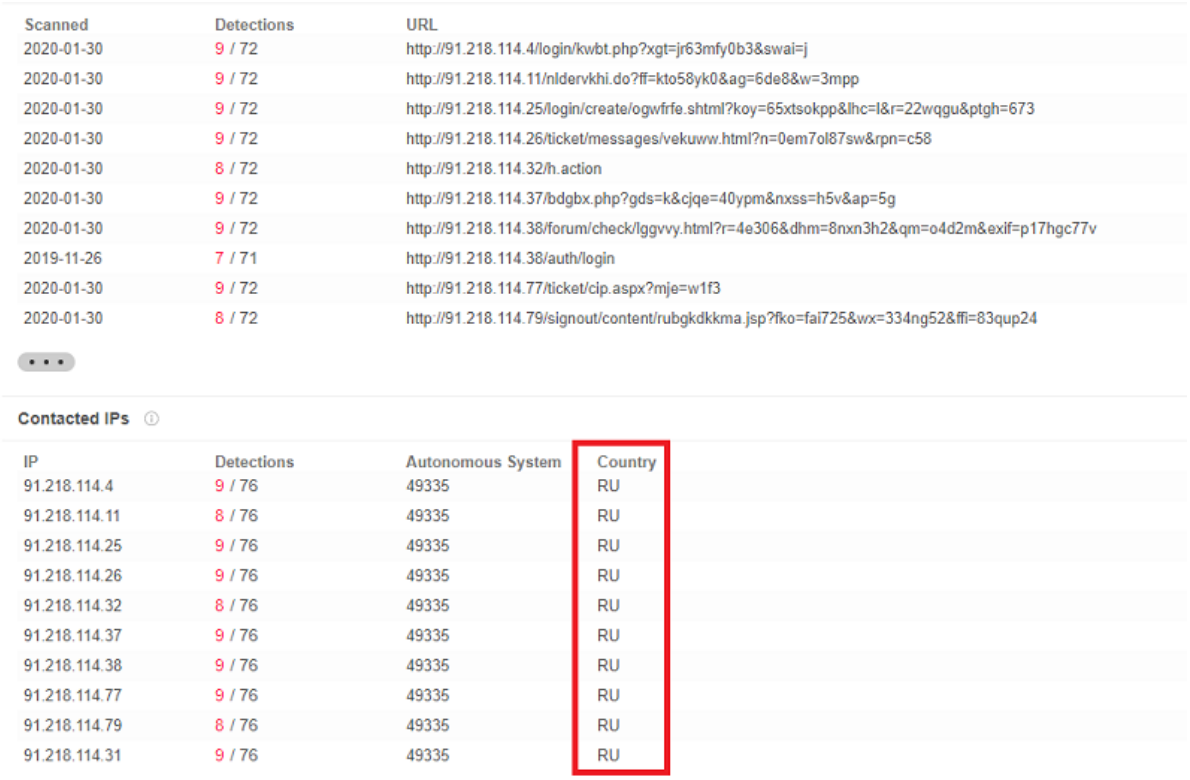
Another interesting fact from the later sample is that, besides it saying that the language code used to program it was Korean, the IPs where it connects belong to the Russian Federation as before, as can be seen in the next two images.



ExifTool File Metadata

CharacterSet	Unicode
CodeSize	73216
CompanyName	DjDjdfodgs
EntryPoint	0x6e2c
FileDescription	ergwt45y
FileFlagsMask	0x003f
FileOS	Windows NT 32-bit
FileSubtype	0
FileType	Win32 EXE
FileTypeExtension	exe
FileVersion	1.0.0.1
FileVersionNumber	1.0.0.1
ImageFileCharacteristics	Executable, 32-bit
ImageVersion	0
InitializedDataSize	1076224
InternalName	argfdg
LanguageCode	Korean
LegalCopyright	Copyright (C) 2024
LinkerVersion	12

FIGURE 35. LANGUAGE CODE “USED” IN THE PACKER SAMPLE, NOT THE MALWARE



Scanned	Detections	URL
2020-01-30	9 / 72	http://91.218.114.4/login/kwbt.php?xgt=jr63mfy0b3&swai=j
2020-01-30	9 / 72	http://91.218.114.11/nldervkhi.do?ff=kto58yk0&ag=6de8&w=3mpp
2020-01-30	9 / 72	http://91.218.114.25/login/create/ogwfrfe.shtml?koy=65xtsokpp&lhc=l&r=22wqgu&ptgh=673
2020-01-30	9 / 72	http://91.218.114.26/ticket/messages/vekuww.html?n=0em7ol87sw&rpn=c58
2020-01-30	8 / 72	http://91.218.114.32/h.action
2020-01-30	9 / 72	http://91.218.114.37/bdgbx.php?gds=k&cjqe=40ypm&nxss=h5v&ap=5g
2020-01-30	9 / 72	http://91.218.114.38/forum/check/lggvvy.html?r=4e306&dhm=8nxn3h2&qm=o4d2m&exif=p17hgc77v
2019-11-26	7 / 71	http://91.218.114.38/auth/login
2020-01-30	9 / 72	http://91.218.114.77/ticket/cip.aspx?mje=w1f3
2020-01-30	8 / 72	http://91.218.114.79/signout/content/rubgkdkkma.jsp?fko=fai725&wx=334ng52&ffi=83qup24

• • •

Contacted IPs

IP	Detections	Autonomous System	Country
91.218.114.4	9 / 76	49335	RU
91.218.114.11	8 / 76	49335	RU
91.218.114.25	9 / 76	49335	RU
91.218.114.26	9 / 76	49335	RU
91.218.114.32	8 / 76	49335	RU
91.218.114.37	9 / 76	49335	RU
91.218.114.38	9 / 76	49335	RU
91.218.114.77	9 / 76	49335	RU
91.218.114.79	8 / 76	49335	RU
91.218.114.31	9 / 76	49335	RU

FIGURE 36. ALL C2 DOMAINS BELONG TO THE RUSSIAN FEDERATION

It is impossible to know the truth, but this could be a trick to try misleading researchers into thinking that the malware comes from some country when in truth it originates in another. It is known that malware developers often check the language on potential victim's machines to avoid the CIS countries, so we can guess that the check for the "Korean" language was a trick designed to mislead, but it is impossible to know that for sure. Of course, the "Korean" language can be changed manually, or it could be a Korean packer, but it is impossible to say with certainty.

CONCLUSION

Maze is a ransomware created by skilled developers. It uses a lot of tricks to make analysis very complex by disabling disassemblers and using pseudocode plugins.

It poses a big problem to individuals and enterprises that do not pay as the developers threaten to release the information if they do not receive payment and they do indeed keep their word on that. More and more ransomwares are exhibiting the same behavior and we expect to see more of it this year and perhaps further into the future too.

The malware developers are active on social media sites, such as Twitter, and they are familiar with the work of malware researchers. They also know how to provoke them perfectly and they like to play cat and the mouse with them.

We recommend making periodic backups of files and keeping them isolated off the network and having an always updated antivirus in place. The latest software patch should also be applied. Remote Desktop Connections that are not needed should be avoided.

Avoid suspicious emails and do not open attachments that come from anyone that you do not know. The same goes for links in emails and, even if they come from a known source, check with the sender if you have any doubts. Also, disable macros in Office programs and never enable them unless it is essential to do so.

COVERAGE

McAfee protects against this threat in all its products, including personal antivirus, endpoint and gateway.

The names that it can have are:

Ransom-Maze!<hash>

YARA RULE

rule maze_unpacked {

```
  meta:
    description = “Rule to detect unpacked Maze samples”
    author = “Marc Rivero | McAfee ATR Team”
strings:
    $opcode_sequence = { 5589e583ec208b450c8b4d08c745fc00 }
        $opcode_sequence_2 = { 5589e553575683e4f883ec28c7042400 }
        $opcode_sequence_3 = { 5589e55dc3662e0f1f84000000000090 }
        $opcode_sequence_4 = { 5589e553575683e4f081ec600200008b }
        $opcode_sequence_5 = { 5589e553575683e4f081ecc00000000f }
        $opcode_sequence_6 = { 5589e583ec208b45108b4d0c8b550883 }
        $opcode_sequence_7 = { 5589e5575683ec388b45108b4d0c8b55 }
        $opcode_sequence_8 = { 5589e5575683e4f883ec088b45088b48 }
        $opcode_sequence_9 = { 558b6c241468997a41000f84bdc50000 }
        $opcode_sequence_10 = { 5589e553575683e4f883ec588b5d088b }
        $opcode_sequence_11 = { 5589e553575683e4f083ec408a42048b }
        $opcode_sequence_12 = { 5589e583ec188b4508837d08008945fc }
        $opcode_sequence_13 = { 5589e553575683e4f8b8d05b0000687f }
        $opcode_sequence_14 = { 5589e5508b450831c98945fc89c883c4 }
        $opcode_sequence_15 = { 5589e553575683e4f883ec708b5d0889 }
        $opcode_sequence_16 = { 5589e583ec308b45088b4d08894df883 }
        $opcode_sequence_17 = { 5589e553575683e4f881ec18030000f2 }
        $opcode_sequence_18 = { 5589e583ec188b45088b4d08894df48b }
        $opcode_sequence_19 = { 5589e583ec2056be74c14400566a0068 }
        $opcode_sequence_20 = { 5589e553575683e4f081ec900000008b }
        $opcode_sequence_21 = { 5589e583e4f083ec208b4d108b450c0f }
```

$opcode_sequence_22 = { 5589e55383e4f883ec108b4d0c8b4508 }

$opcode_sequence_23 = { 558b8e150409133f03fd08f81b0c4f22 }

$opcode_sequence_24 = { 5589e553575683e4f883ec7031f68379 }

$opcode_sequence_25 = { 5589e553575683e4f881ec3001000089 }

$opcode_sequence_26 = { 5589e553575683e4f881ece00000000f }

$opcode_sequence_27 = { 558b589608361d1943a57d0ba6492beb }

$opcode_sequence_28 = { 5589e553575683e4f883ec1089ce6a00 }

$opcode_sequence_29 = { 5589e5575683e4f883ec688b75088b7d }

$opcode_sequence_30 = { 5589e553575683e4f883ec386a006a00 }

$opcode_sequence_31 = { 558b7c240868dca8440057683d484300 }

$opcode_sequence_32 = { 5589e55683e4f881ec2801000089ce8d }

$opcode_sequence_33 = { 5589e583ec188b450831c98b5508c704 }

$opcode_sequence_34 = { 5589e583ec308b450c8b4d088b55088b }

$opcode_sequence_35 = { 5589e583ec348b450831c983c1188b55 }

$opcode_sequence_36 = { 5589e553575683e4f881ec78040000f2 }

$opcode_sequence_37 = { 5589e583ec108b4508837d08008945f8 }

$opcode_sequence_38 = { 5589e583ec348b4508837d08008945dc }

$opcode_sequence_39 = { 5589e55683ec548b45088b4d08894df0 }

$opcode_sequence_40 = { 558bec5de9a48efeffe9ef8efeffcccc }

$opcode_sequence_41 = { 5589e553575683ec108b45108b4d0c8b }

$opcode_sequence_42 = { 5589e5575683ec348b4508c745f40100 }

$opcode_sequence_43 = { 558bec8325a0c345000083ec1c5333db }

$opcode_sequence_44 = { 5589e553575683e4f083ec208b750c0f }

$opcode_sequence_45 = { 5589e583ec348b450c8b4d088b55088b }

$opcode_sequence_46 = { 558b6fd8d843ef516154e2526781aecd }

condition:

(uint16(0) == 0x5a4d) and 38 of them

}

IOCs

SHA256	dee863ffa251717b8e56a96e2f9f0b41b09897d3c7cb2e8159fcb0ac0783611b
SHA1	31c3f7b523e1e406d330958e28882227765c3c5e
SHA256	b345697c16f84d3775924dc17847fa3ff61579ee793a95248e9c4964da586dd1
SHA1	c5938ec75e5b655be84eb94d73adec0f63fbce16
SHA256	5a900fd26a4ece38de5ca319b5893f96c7e9e2450dbac796c12f85b99238ec18
SHA1	1e994b5ac039a1c7612bab93248532bf3ed7e6de

Network

Domain	mazedecrypt.top
IP	91.218.114.11
IP	91.218.114.25
IP	91.218.114.26
IP	91.218.114.31
IP	91.218.114.32
IP	91.218.114.37
IP	91.218.114.38
IP	91.218.114.4
IP	91.218.114.77
IP	91.218.114.79

MITRE ATT&CK COVERAGE

- CommonlyUsedPort
- StandardApplicationLayerProtocol
- SecuritySoftwareDiscovery

- SystemTimeDiscovery
- CommandLineInterface
- DataEncrypted
- DataEncryptedForImpact
- Query registry
- Hooking

[1] https://twitter.com/jeromesegura/status/1133767240686288896

[2] https://www.bleepingcomputer.com/news/security/maze-ransomware-demands-6-million-ransom-from-southwire/

[3] https://www.bleepingcomputer.com/news/security/nemty-ransomware-to-start-leaking-non-paying-victims-data/

[4] https://twitter.com/McAfee_Labs/status/1206651980086685696

[5] https://www.bleepingcomputer.com/news/security/new-threat-actor-impersonates-govt-agencies-to-deliver-malware/

[6] https://securityintelligence.com/news/spelevo-ek-exploits-flash-player-vulnerability-to-deliver-maze-ransomware/

[7] https://github.com/revsic/AntiDebugging

[8] https://ss64.com/locale.html

[9] https://twitter.com/malwrhunterteam/status/1222253947332841472

[10] https://twitter.com/luca_nagy_/status/1222819371644522500

Alexandre Mundo
Alexandre Mundo, Senior Malware Analyst is part of Mcafee's Advanced Threat Research team. He reverses the new threads in advanced attacks and make research of them in a daily basis....

More from McAfee Labs

Crypto Scammers Exploit: Elon Musk Speaks on Cryptocurrency

By Oliver Devane Update: In the past 24 hours (from time of publication) McAfee has identified 15...

May 05, 2022 | 4 MIN READ

Instagram Credentials Stealer: Disguised as Mod App

Authored by Dexter Shin McAfee's Mobile Research Team introduced a new Android malware targeting Instagram users who...

May 03, 2022 | 4 MIN READ

Instagram Credentials Stealers: Free Followers or Free Likes

Authored by Dexter Shin Instagram has become a platform with over a billion monthly active users. Many...

May 03, 2022 | 6 MIN READ

Scammers are Exploiting Ukraine Donations

Authored by Vallabh Chole and Oliver Devane Scammers are very quick at reacting to current events, so...

Apr 01, 2022 | 7 MIN READ

Imposter Netflix Chrome Extension Dupes 100k Users

Authored by Oliver Devane, Vallabh Chole, and Aayush Tyagi McAfee has recently observed several malicious Chrome Extensions...

Mar 10, 2022 | 8 MIN READ

Why Am I Getting All These Notifications on my Phone?

Authored by Oliver Devane and Vallabh Chole Notifications on Chrome and Edge, both desktop browsers, are commonplace,...

Feb 25, 2022 | 5 MIN READ

Emotet’s Uncommon Approach of Masking IP Addresses

In a recent campaign of Emotet, McAfee Researchers observed a change in techniques. The Emotet maldoc was...

Feb 04, 2022 | 4 MIN READ

HANCITOR DOC drops via CLIPBOARD

Hancitor, a loader that provides Malware as a Service, has been observed distributing malware such as FickerStealer,...

Dec 13, 2021 | 6 MIN READ

‘Tis the Season for Scams

‘Tis the Season for Scams

Nov 29, 2021 | 18 MIN READ

The Newest Malicious Actor: “Squirrelwaffle” Malicious Doc.

Authored By Kiran Raj Due to their widespread use, Office Documents are commonly used by Malicious actors...

Nov 10, 2021 | 4 MIN READ

Social Network Account Stealers Hidden in Android Gaming Hacking Tool

Authored by: Wenfeng Yu McAfee Mobile Research team recently discovered a new piece of malware that specifically...

Oct 19, 2021 | 6 MIN READ

Malicious PowerPoint Documents on the Rise

Authored by Anuradha M McAfee Labs have observed a new phishing campaign that utilizes macro capabilities available...

Sep 21, 2021 | 6 MIN READ