

A Deep Dive into AvosLocker Ransomware

LIFARS

a  SecurityScorecard company

SecurityScorecard.com

info@securityscorecard.com

©2022 SecurityScorecard Inc.

244 Fifth Avenue, Suite 2035,

New York, NY 10001

1.212.222.7061

Table of Contents

Executive Summary	2
Analysis and Findings	2
Thread activity – sub_CBB930 function	10
Thread activity – sub_CBBAD0 function	13
Running with the -h (--help) parameter	21
Running with the -p (--path) parameter	21
Running with the -l (--disabledrives) parameter	21
Running with the --hide parameter	21
Running with the -t (--threads) parameter	22
Running with the -n (--enablesmb) parameter	22
Running with the -b (--brutesmb) -n (--enablesmb) parameters	23
Running with the --nomutex parameter	24
Indicators of Compromise	25

Executive Summary

AvosLocker is a ransomware-as-a-service (RaaS) group that appeared in 2021. The malware can run with one of the following parameters: "--help", "--path", "--disabledrives", "--hide", "--threads", "--enablesmb", "--brutesmb", and "--nomutex." The ransomware kills a list of targeted processes, deletes all Volume Shadow Copies using two commands, and clears all Windows event logs. The binary can target the logical drives as well as network shares by specifying proper arguments.

The encryption is done using multithreading with I/O completion ports. AvosLocker uses a combination of RSA and Salsa20 algorithms during the encryption process. Finally, the ransomware creates an image based on the ransom note text that is set as the Desktop Wallpaper.

Analysis and Findings

SHA256: EC955F589F25D0D28E55964A1AA79C27492026982994CD4CA1FAF7E8A78DB4BC

The malware performs a call to GetCurrentProcess and then opens the access token associated with the current process using the OpenProcessToken API (0xF01FF = **TOKEN_ALL_ACCESS**):

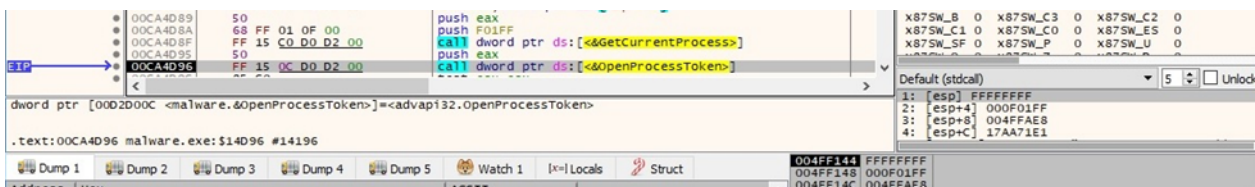


Figure 1

Most of the strings are encrypted using the XOR operator. An example of a decryption algorithm is displayed in figure 2:

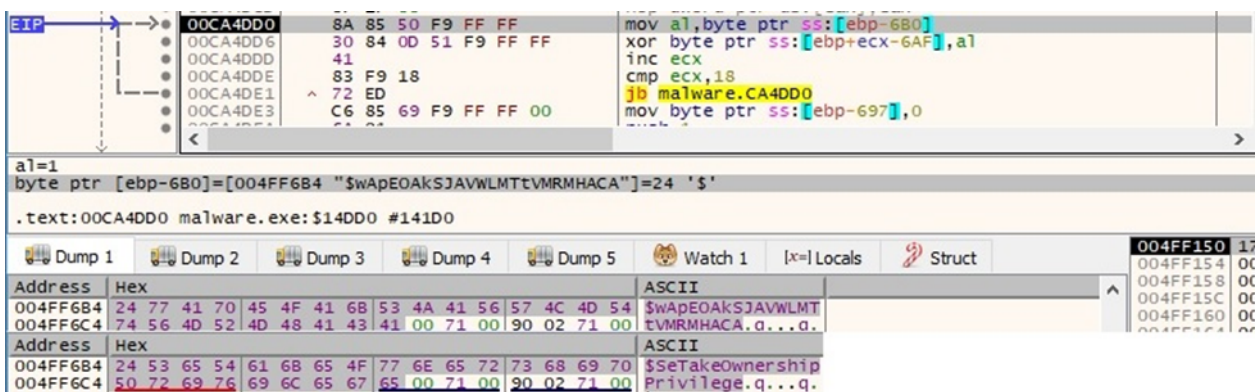


Figure 2

The LookupPrivilegeValueA function is utilized to retrieve the LUID (locally unique identifier) corresponding to the "SeTakeOwnershipPrivilege" privilege:

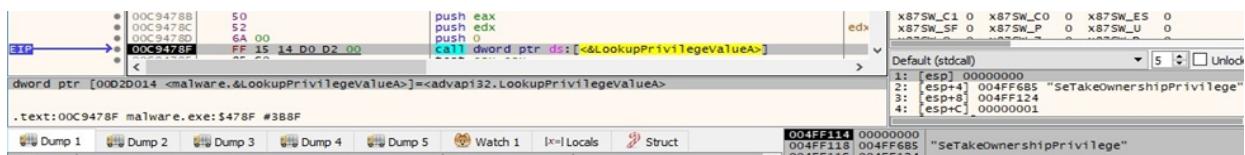


Figure 3

AvosLocker enables the above privilege in the access token via a function call to AdjustTokenPrivileges:



Figure 4

The binary decrypts a list of processes that will be killed (figure 5):

- "encsvc" "thebat" "mydesktopqos" "xfssvccon" "firefox" "infopath" "winword" "steam" "synctime" "notepad"
- "ocomm" "onenote" "mspub" "thunderbird" "agntsvc" "sql" "excel" "powerpnt" "outlook" "wordpad" "dbeng50"
- "isqlplussvc" "sqbcoreservice" "oracle" "ocautoupds" "dbsnmp" "msaccess" "tbirdconfig" "ocssd" "mydesktopservice" "visio"

Address	Hex	ASCII
004FEB18	27 65 6E 63 73 76 63 38 74 68 65 62 61 74 38 6D	'encsvc;thebat;m
004FEB28	79 64 65 73 68 74 6F 70 71 6F 73 38 78 66 73 73	ydesktopqos;xfss
004FEB38	76 63 63 6F 6E 38 66 69 72 65 66 6F 78 38 69 6E	vccon;firefox;in
004FEB48	66 6F 70 61 74 68 38 77 69 6E 77 6F 72 64 38 73	fopath;winword;s
004FEB58	74 65 61 6D 38 73 79 6E 63 74 69 6D 65 38 6E 6F	team;synctime;no
004FEB68	74 65 70 61 64 38 6F 63 6F 6D 6D 38 6F 6E 65 6E	tepad;ocomm;onen
004FEB78	6F 74 65 38 6D 73 70 75 62 38 74 68 75 6E 64 65	ote;mspub;thunde
004FEB88	72 62 69 72 64 38 61 67 6E 74 73 76 63 38 73 71	rbird;agntsvc;sq
004FEB98	6C 38 65 78 63 65 6C 38 70 6F 77 65 72 70 6E 74	l;excel;powerpnt
004FEBA8	38 6F 75 74 6C 6F 6F 68 38 77 6F 72 64 70 61 64	;outlook;wordpad
004FEBB8	38 64 62 65 6E 67 35 30 38 69 73 71 6C 70 6C 75	;dbeng50;isqlplu
004FEBC8	73 73 76 63 38 73 71 62 63 6F 72 65 73 65 72 76	ssvc;sqbcoreserv
004FEBD8	69 63 65 38 6F 72 61 63 6C 65 38 6F 63 61 75 74	ice;oracle;ocaut
004FEBE8	6F 75 70 64 73 38 64 62 73 6E 6D 70 38 6D 73 61	oupds;dbsnmp;msa
004FEBF8	63 63 65 73 73 38 74 62 69 72 64 63 6F 6E 66 69	ccess;tbirdconfi
004FEC08	67 38 6F 63 73 73 64 38 6D 79 64 65 73 68 74 6F	g;ocssd;mydesko
004FEC18	70 73 65 72 76 69 63 65 38 76 69 73 69 6F 00 00	pservice;visio..

Figure 5

CreateToolhelp32Snapshot is used to take a snapshot of all processes in the system (0x2 = TH32CS_SNAPPROCESS):

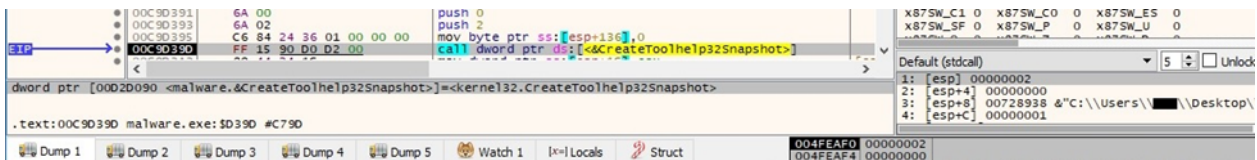


Figure 6

The ransomware extracts information about the first process from the snapshot using the Process32First routine:

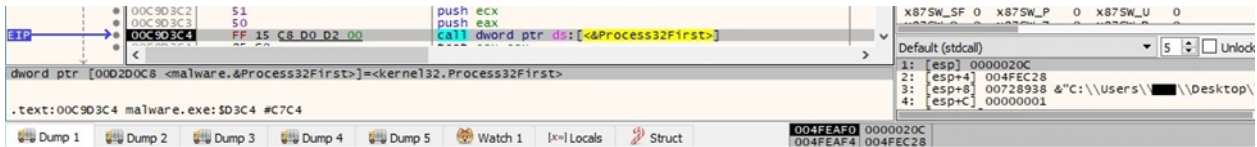


Figure 7

There is a comparison between the process name and the blacklisted processes:

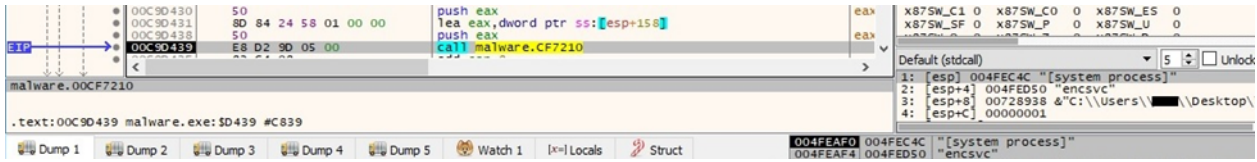


Figure 8

The malicious binary retrieves information about the next process from the snapshot via a call to Process32Next:

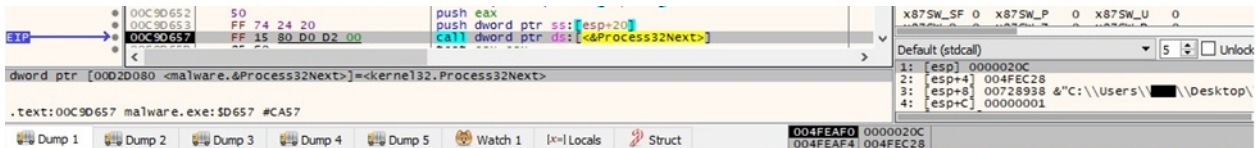


Figure 9

AvosLocker uses the FNV (Fowler-Noll-Vo) hashing algorithm to identify and call relevant APIs at runtime:

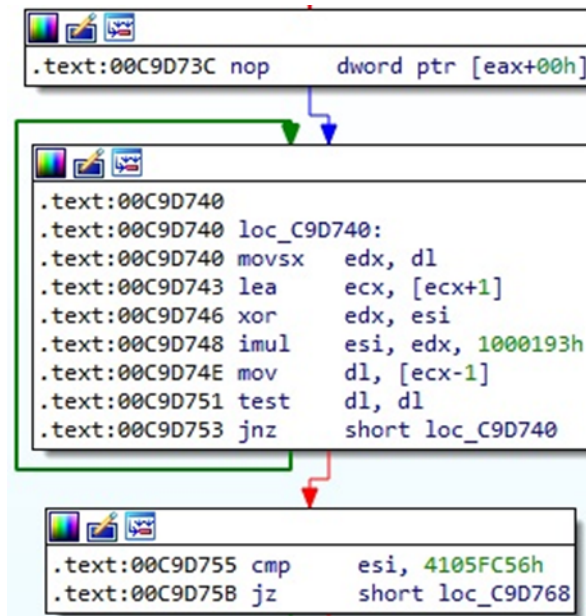


Figure 10

The executable opens a targeted process using OpenProcess (0x1 = **PROCESS_TERMINATE**):

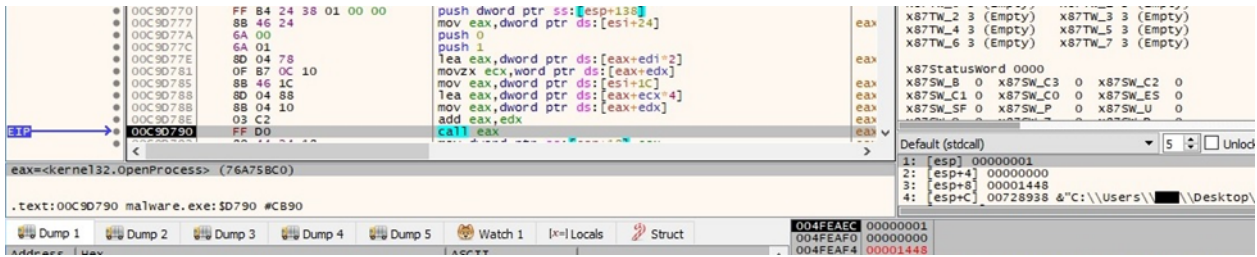


Figure 11

The `TerminateProcess` routine is utilized to kill the targeted process:

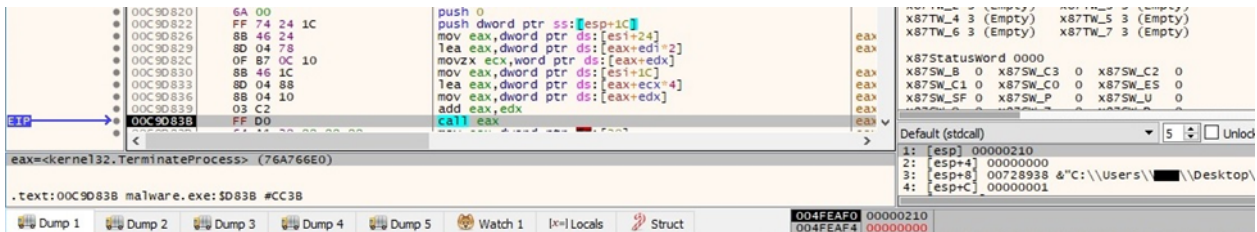


Figure 12

The ransomware writes the following data in the command line output:



Figure 13

We'll explain the purpose of each parameter in the following paragraphs.

The malware creates a mutex called "ZheicOWaWie6zeiy", which ensures that only one copy of the process is running at a single time:

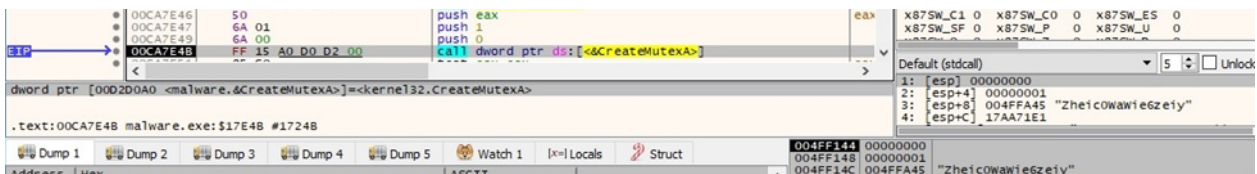


Figure 14

The process disables file system redirection by calling the `Wow64DisableWow64FsRedirection` API:

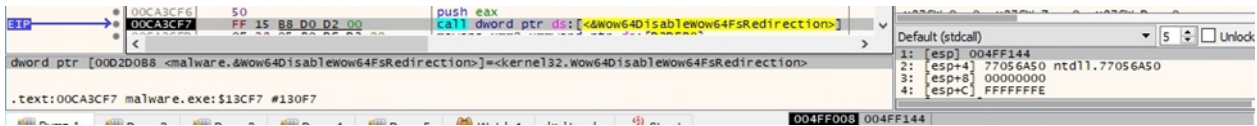


Figure 15

The binary calls the WinExec function in order to spawn multiple processes (figure 16):

- cmd /c wmic shadowcopy delete /nointeractive – delete volume shadow copies
- cmd /c vssadmin.exe Delete Shadows /All /Quiet – delete volume shadow copies
- cmd /c bcdedit /set {default} recoveryenabled No – disable automatic repair
- cmd /c bcdedit /set {default} bootstatuspolicy ignoreallfailures – ignore errors in the case of a failed boot / shutdown / checkpoint
- cmd /c powershell -command \"Get-EventLog -LogName * | ForEach { Clear-EventLog \$_.Log }\" - clear all entries from the event logs

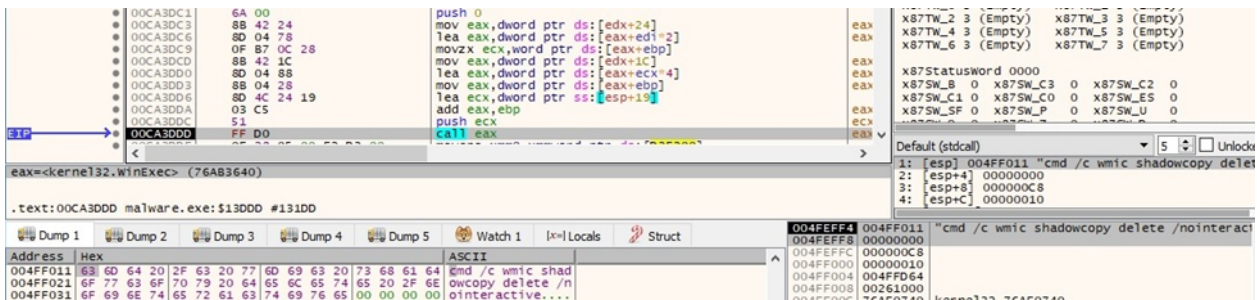


Figure 16

The file comes with a hard-coded RSA public key:

Address	Hex	ASCII
00D50389	2D 2D 2D 2D 42 45 47 49 4E 20 50 55 42 4C 49 43	-----BEGIN PUBLIC
00D503C9	20 48 45 59 2D 2D 2D 2D 2D 0A 4D 49 49 42 49 6A	KEY-----.MIIBIj
00D503D9	41 4E 42 67 6B 71 68 6B 69 47 39 77 30 42 41 51	ANBqkqhkiG9w0BAQ
00D503E9	45 46 41 41 4F 43 41 51 38 41 4D 49 49 42 43 67	EFAAOCAQ8AMIIBcG
00D503F9	48 43 41 51 45 41 69 6F 54 4B 50 4C 6E 72 61 63	KCAQEAIOTKPLnrAc
00D50409	30 43 4A 49 48 30 4D 77 48 69 0A 4A 6D 6C 43 78	OCJIKOmWhiJmICx
00D50419	70 66 35 6E 38 62 48 73 4F 6C 33 55 35 51 4F 59	pf5n8bHs013U5QOY
00D50429	46 48 33 52 62 4A 4C 5A 50 34 73 33 64 46 67 5A	FK3rbJLP4s3dfGZ
00D50439	6C 68 65 34 61 56 78 2F 6A 35 76 66 68 6B 77 6E	1ke4vx/j5vfhwkN
00D50449	78 56 30 58 74 43 6F 39 54 62 51 0A 38 45 31 4C	xV0xtcO9TbQ.8E1L
00D50459	7A 74 71 53 76 6D 59 78 35 4F 4D 32 54 53 49 61	Ztq5VmYx5OM2TSia
00D50469	56 36 39 6E 6D 48 64 30 5A 35 57 35 4F 53 48 63	V69nmkd0Z5W5OSHc
00D50479	75 74 73 56 56 65 41 62 68 39 73 54 55 56 36 45	utsVveAbk9sTUV6E
00D50489	31 39 42 54 77 59 6A 67 72 4C 68 2B 0A 50 6A 6A	I98TWyigrLk+.Pjj
00D50499	2B 46 52 64 43 4D 43 56 59 70 42 6C 33 4C 50 6F	+FRdCMCVvpB13LPo
00D504A9	68 50 53 75 6F 42 48 5A 68 31 58 77 34 53 47 6B	kPSuoBKZk1Xw45Gk
00D504B9	4F 36 55 61 2B 48 4E 79 78 69 67 42 69 4E 6D 73	O6Ua+KNyx1gB1Nms
00D504C9	56 53 65 76 58 48 65 48 51 6F 33 73 56 0A 77 4A	VSevXkEKQo3sv.wJ
00D504D9	4A 45 68 52 58 70 65 55 32 75 46 63 34 48 45 32	JEHRXpeU2uFc4KE2
00D504E9	6C 59 4D 79 72 55 71 66 58 41 72 31 71 61 46 36	1YMyrUqfXAr1qaF6
00D504F9	49 56 34 79 68 61 50 71 73 6C 5A 69 51 46 4E 70	Iv4ykaPqs1z1QFNp
00D50509	5A 42 57 6F 71 6A 37 4F 56 68 66 41 30 57 0A 45	ZBwojqj70vkFAOW.E
00D50519	69 42 2B 73 31 4A 6C 76 67 79 35 68 59 31 74 2F	1B+s1J1vgysHY1t/
00D50529	66 4E 31 51 5A 78 2F 62 37 49 70 68 68 6A 2F 50	fN1QZx/b7Iphkj/P
00D50539	61 36 6F 72 36 52 4E 39 44 6A 35 53 46 4D 63 6B	a6or6RN9dJ5SFmck
00D50549	6D 77 6C 44 77 67 52 4F 6B 6A 7A 30 37 74 73 0A	mw1DwgROKj207ts.
00D50559	46 51 49 44 41 51 41 42 0A 2D 2D 2D 2D 45 4E	FQIDAQAB.-----EN
00D50569	44 20 50 55 42 4C 49 43 20 4B 45 59 2D 2D 2D 2D	D PUBLIC KEY-----
00D50579	2D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Figure 17

AvosLocker creates an input/output (I/O) completion port that is not yet associated with a file handle (0xFFFFFFFF = INVALID_HANDLE_VALUE):

Figure 18

The malware creates multiple threads that will handle the files encryption. As we can see in figures 19 and 20, even if the starting address of the thread is StartAddress (sub_D0155F), the actual relevant function that will be called is sub_CBBAD0:

Figure 19

Figure 20

The thread's priority is set to 0x2 (**THREAD_PRIORITY_HIGHEST**) via a function call to SetThreadPriority:

Figure 21

The number of created threads is 200 (default value); however, it can be modified using the -t (or --threads) parameter.

FindFirstVolumeW is utilized to retrieve the first volume of the local machine:

Figure 22

The ransomware extracts a list of drive letters and mounted folder paths for a volume using the GetVolumePathNamesForVolumeNameW function:

```

00CA4C8B 50          push eax
00CA4C8C 57          push 78
00CA4C8D 8D 44 24 28 lea eax,dword ptr ss:[esp+28]
00CA4CAF 50          push eax
00CA4CB0 57          push esi
00CA4CB1 FF 15 5C D0 D2 00 call dword ptr ds:[<&GetVolumePathNamesForVolumeNameW>]

```

Register pane (Default (stdcall)):

```

1: [esp] 0073A0E8 L"\\\\?\\Volume{d7e47829-0000-0000-0000-100000000000}
2: [esp+4] 004FEF38
3: [esp+8] 00000078
4: [esp+C] 004FEF2C

```

Figure 23

The volume enumeration continues by calling FindNextVolumeW:

```

00CA4D05 68 00 80 00 00 push 8000
00CA4D0A 57          push edi
00CA4D0B 56          push esi
00CA4D0C FF 15 50 D0 D2 00 call dword ptr ds:[<&FindNextVolumeW>]

```

Register pane (Default (stdcall)):

```

1: [esp] 00738DD0
2: [esp+4] 0073A0E8 L"\\\\?\\Volume{d7e47829-0000-0000-0000-100000000000}
3: [esp+8] 00008000
4: [esp+C] 000000C8

```

Figure 24

The malware is looking for volumes that aren't mounted using the GetDriveTypeW routine (0x1 = DRIVE_NO_ROOT_DIR):

```

00CA4CDB 50          push eax
00CA4CDC FF D3       call ebx
00CA4CDE 83 F8 01   cmp eax,1
00CA4CE1 74 12     jle malware.CA4CF5

```

Register pane (Default (stdcall)):

```

1: [esp] 004FEF30 L"Z:\\\\"
2: [esp+4] 000000C8
3: [esp+8] 00000010
4: [esp+C] 00261000

```

Figure 25

The binary associates an unmounted volume with a drive letter using SetVolumeMountPointW:

```

00CA4CF5 57          push edi
00CA4CF6 8D 44 24 1C lea eax,dword ptr ss:[esp+1C]
00CA4CFA 50          push eax
00CA4CFB FF 15 78 D0 D2 00 call dword ptr ds:[<&SetVolumeMountPointW>]

```

Register pane (Default (stdcall)):

```

1: [esp] 004FEF30 L"Z:\\\\"
2: [esp+4] 0073A0E8 L"\\\\?\\Volume{d7e47829-0000-0000-0000-b10000000000}
3: [esp+8] 000000C8
4: [esp+C] 00000010

```

Figure 26

AvosLocker obtains a bitmask representing the available disk drives:

```

00CA2962 FF 15 84 65 D5 00 call dword ptr ds:[<&GetLogicalDrives>]

```

Register pane (Default (stdcall)):

```

1: [esp] 17AA70CD
2: [esp+4] 000000C8
3: [esp+8] 00000010
4: [esp+C] 00261000

```

Figure 27

The process creates a thread for each drive that is found. The same method as above is utilized here, i.e., the starting address of the thread is StartAddress (sub_D0155F); however, the important function is sub_CBB930:

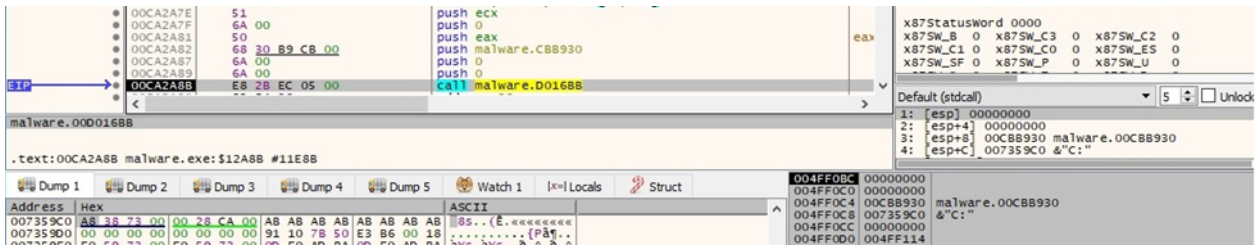


Figure 28

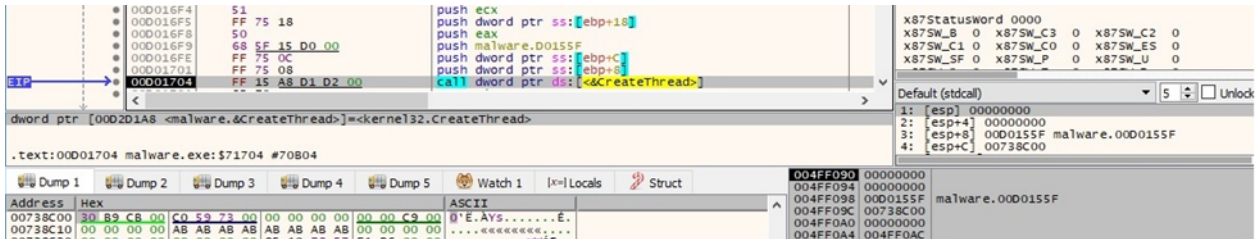


Figure 29

All identified drives are written to the command line, as highlighted in figure 30.

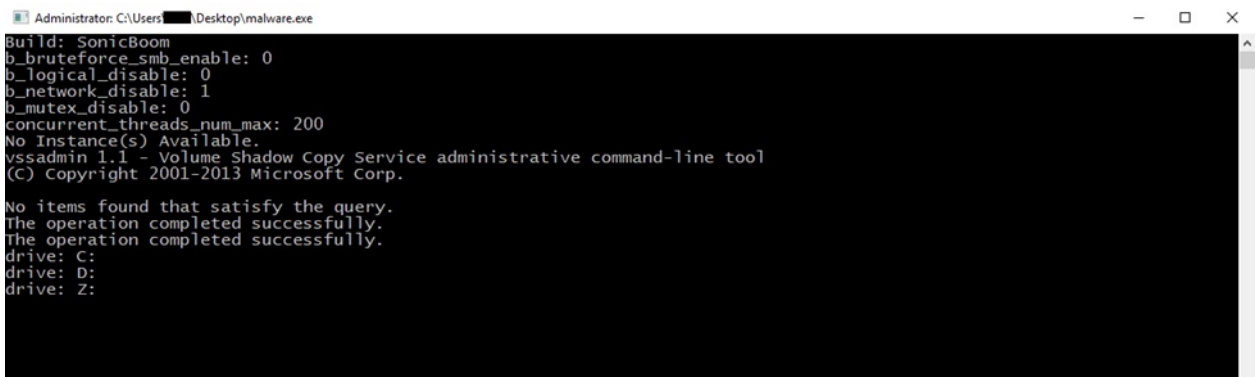


Figure 30

The new threads' priority is also set to **THREAD_PRIORITY_HIGHEST** by the malicious binary.

GetConsoleWindow is used to retrieve the window handle used by the console associated with the process:

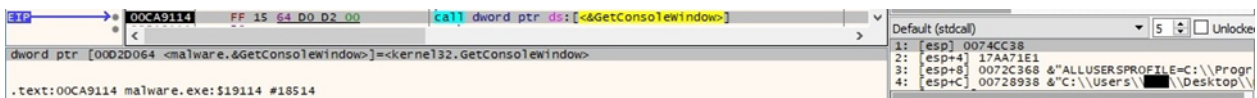


Figure 31

The malware calls the ShutdownBlockReasonCreate API and indicates that the machine should not be shut down during the encryption process:

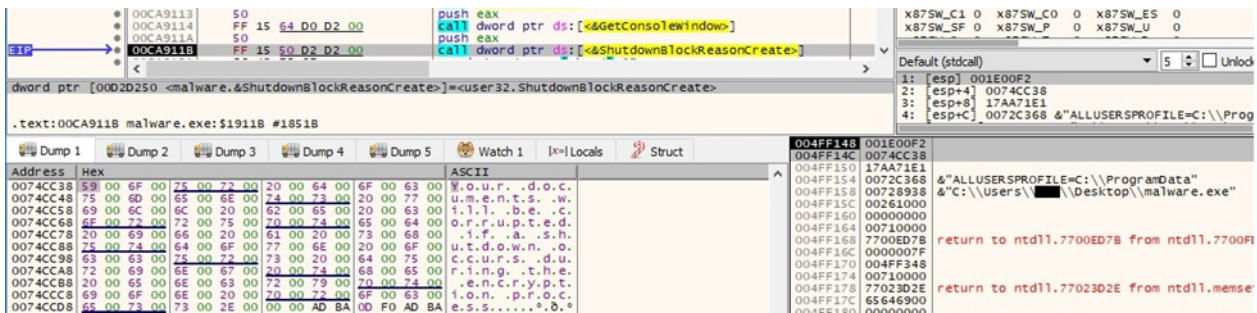


Figure 32

The ransomware extracts the identifier of the calling thread:

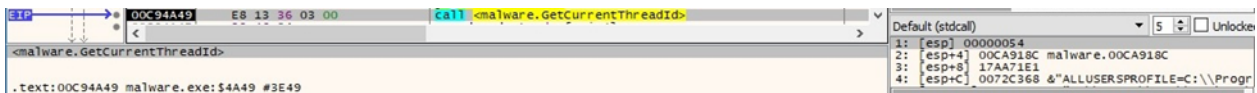


Figure 33

AvosLocker blocks the calling thread until all created threads will terminate their work using the Join method:

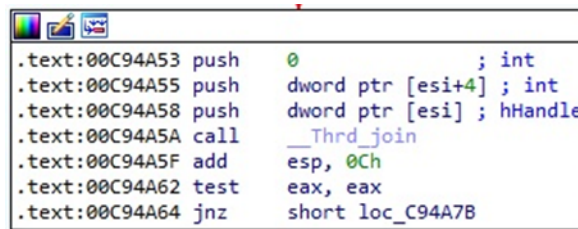


Figure 34

Thread activity – sub_CBB930 function

The ransomware decrypts a list of extensions that will be skipped (figure 35):

- "avos" "avoslinux" "avos2" "avos2j" "themepack" "nls" "diagpkg" "msi" "lnk" "exe" "cab" "scr" "bat" "drv" "rtp" "msp"
- "prf" "msc" "ico" "key" "ocx" "diagcab" "diagcfg" "pdb" "wpx" "hlp" "icns" "rom" "dll" "msstyles" "mod" "ps1" "ics" "hta"
- "bin" "cmd" "ani" "386" "lock" "cur" "idx" "sys" "com" "deskthemepack" "shs" "ldf" "theme" "mpa" "nomedia" "spl" "cpl" "adv" "icl" "msu"

Address	Hex	ASCII
049BE5ED	61 76 6F 73 20 61 76 6F 73 6C 69 6E 75 78 20 61	avos avoslinux a
049BE5FD	76 6F 73 32 20 61 76 6F 73 32 6A 20 74 68 65 6D	vos2 avos2j them
049BE600	65 70 61 63 68 20 6E 6C 68 20 64 69 61 67 70 68	epack nls diagpk
049BE61D	67 20 6D 73 69 20 6C 6E 68 20 65 78 65 20 63 61	g msi lnk exe ca
049BE62D	62 20 73 63 72 20 62 61 74 20 64 72 76 20 72 74	b scr bat drv rt
049BE63D	70 20 6D 73 70 20 70 72 66 20 6D 73 63 20 69 63	p msp prf msc ic
049BE64D	6F 20 68 65 79 20 6F 63 78 20 64 69 61 67 63 61	o key ocx diagca
049BE65D	62 20 64 69 61 67 63 66 67 20 70 64 62 20 77 70	b diagcfg pdb wp
049BE66D	78 20 68 6C 70 20 69 63 6E 73 20 72 6F 6D 20 64	x hlp icns rom d
049BE67D	6C 6C 20 6D 73 73 74 79 6C 65 73 20 6D 6F 64 20	ll msstyles mod
049BE68D	70 73 31 20 69 63 73 20 68 74 61 20 62 69 6E 20	psi ics hta bin
049BE69D	63 6D 64 20 61 6E 69 20 33 38 36 20 6C 6F 63 68	cmd ani 386 lock
049BE6AD	20 63 75 72 20 69 64 78 20 73 79 73 20 63 6F 6D	cur idx sys com
049BE6BD	20 64 65 73 68 74 68 65 6D 65 70 61 63 68 20 73	deskthemepack s
049BE6CD	68 73 20 6C 64 66 20 74 68 65 6D 65 20 6D 70 61	hs ldf theme mpa
049BE6DD	20 6E 6F 6D 65 64 69 61 20 73 70 6C 20 63 70 6C	nomedia spl cpl
049BE6ED	20 61 64 76 20 69 63 6C 20 6D 73 75 00 00 00 00	adv icl msu....

Figure 35

The malicious executable starts enumerating the drive by calling the FindFirstFileW function:

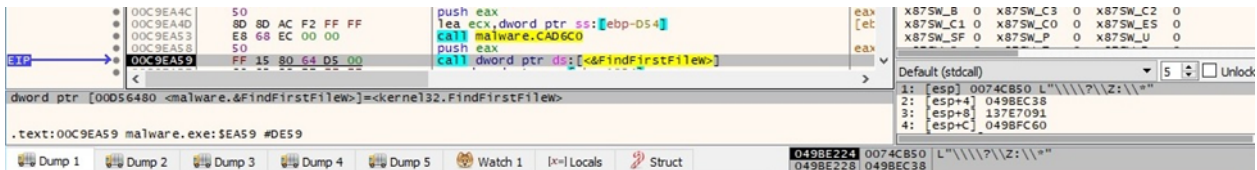


Figure 36

The ransomware creates a ransom note called "GET_YOUR_FILES_BACK.txt" in every directory that will be encrypted (0x4000000 = **GENERIC_WRITE**, 0x2 = **CREATE_ALWAYS**, 0x80 = **FILE_ATTRIBUTE_NORMAL**):

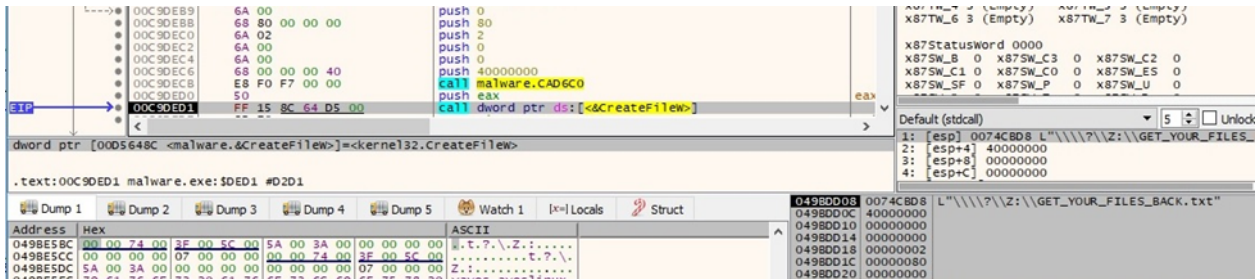


Figure 37

The WriteFile routine is used to populate the ransom note:

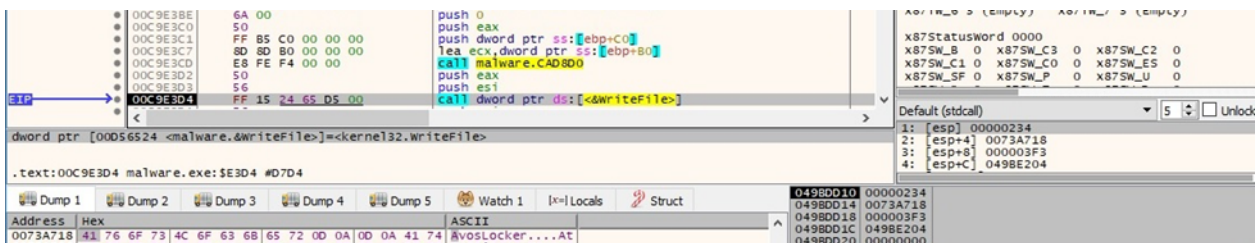


Figure 38

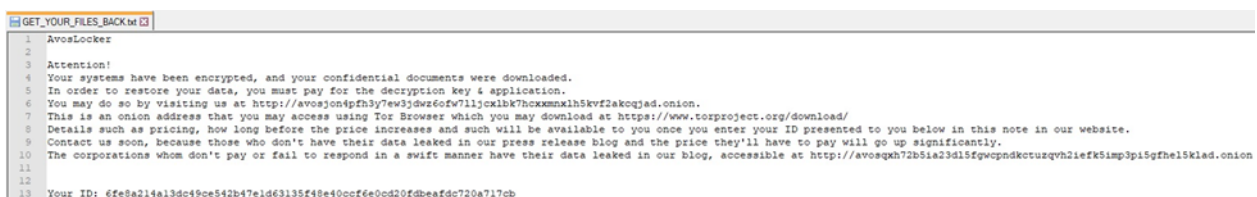


Figure 39

The process decrypts a list of folders that will not be encrypted (figure 40):

- "Program Files" "Windows" "Windows.old" "bootmgr" "ProgramData" "System Volume Information"
- "AppData" "Public" "All Users" "boot" "Intel" "WinNT" "Sophos" "Microsoft." "Games" "config.msi"

Address	Hex	ASCII
049BEA68	17 50 72 6F 67 72 61 6D 20 46 69 6C 65 73 00 00	.Program Files..
049BEA78	3E 50 72 6F 67 72 61 6D 44 61 74 61 00 00 00 00	>ProgramData....
049BEA88	60 57 69 6E 64 6F 77 73 2E 6F 6C 64 00 00 00 00	`windows.old....
049BEA98	28 4D 69 63 72 6F 73 6F 66 74 2E 00 0D 63 6F 6E	(Microsoft...con
049BEAA8	66 69 67 2E 6D 73 69 00 3D 41 6C 6C 20 55 73 65	fig.msi.=All Use
049BEAB8	72 73 00 00 48 62 6F 6F 74 6D 67 72 00 00 00 00	rs..Hbootmgr....
049BEAC8	68 41 70 70 44 61 74 61 00 00 00 00 58 57 69 6E	kAppData...[Win
049BEAD8	64 6F 77 73 00 00 00 00 09 4D 69 63 72 6F 73 6F	dows...Microso
049BEAE8	66 74 2E 00 73 50 75 62 6C 69 63 00 3C 53 6F 70	ft..sPublic.<Sop
049BEAF8	68 6F 73 00 27 49 6E 74 65 6C 00 00 75 57 69 6E	hos.'Intel..uWin
049BEB08	4E 54 00 00 1F 62 6F 6F 74 00 00 00 00 00 00 00	NT...boot.....

Figure 40

AvosLocker continues the file enumeration using FindNextFileW:

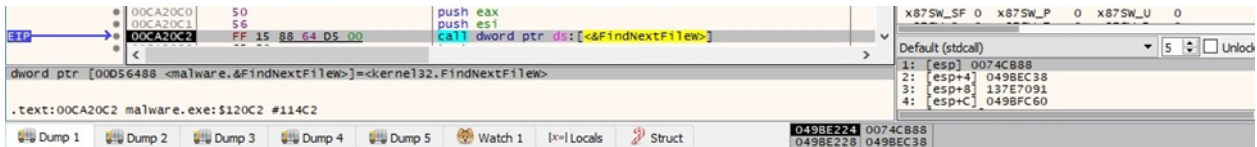


Figure 41

A list of files that will be skipped is decrypted using the XOR operator (figure 42):

- "GET_YOUR_FILES_BACK.txt" "desktop.ini" "autorun.inf" "ntldr" "bootsect.bak" "thumbs.db"
- "boot.ini" "ntuser.dat" "iconcache.db" "bootfont.bin" "ntuser.ini" "ntuser.dat.log" "Thumbs.db"

Address	Hex	ASCII
049BE859	47 45 54 5F 59 4F 55 52 5F 46 49 4C 45 53 5F 42	GET_YOUR_FILES_B
049BE869	41 43 48 2E 74 78 74 00 00 00 00 40 6E 74 75 73	ACK.txt...@ntus
049BE879	65 72 2E 64 61 74 2E 6C 6F 67 00 5D 62 6F 6F 74	er.dat.log.]boot
049BE889	66 6F 6E 74 2E 62 69 6E 00 00 00 78 69 63 6F 6E	font.bin...{icon
049BE899	63 61 63 68 65 2E 64 62 00 00 00 1B 62 6F 6F 74	cache.db...boot
049BE8A9	73 65 63 74 2E 62 61 68 00 00 00 28 61 75 74 6F	sect.bak...+auto
049BE8B9	72 75 6E 2E 69 6E 66 00 00 00 11 64 65 73 68	run.inf....desk
049BE8C9	74 6F 70 2E 69 6E 69 00 00 00 62 6E 74 75 73	top.ini...bntus
049BE8D9	65 72 2E 64 61 74 00 88 CB 74 00 3D 6E 74 75 73	er.dat..Èt.=ntus
049BE8E9	65 72 2E 69 6E 69 00 47 54 68 75 6D 62 73 2E 64	er.ini.GThumbs.d
049BE8F9	62 00 00 63 74 68 75 6D 62 73 2E 64 62 00 00 16	b..cthumbs.db...
049BEC09	62 6F 6F 74 2E 69 6E 69 00 00 00 04 47 61 6D 65	boot.ini....Game

Figure 42

An example of a comparison between the file extension and one that is whitelisted is shown below:

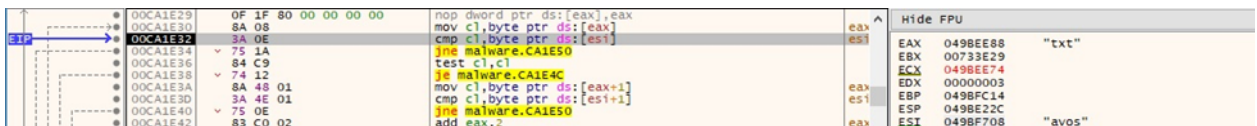


Figure 43

The ransomware sends an I/O completion packet that contains the targeted file path to the IOCP created earlier:

```

00CA2086 FF B5 F8 ED FF FF push dword ptr ss:[ebp-1208]
00CA208C 6A 77 push 77
00CA208E 68 FE FF 00 00 push FFFE
00CA2093 FF 35 7C 64 D5 00 push dword ptr ds:[05647C]
00CA2099 FF 15 98 D0 D2 00 call dword ptr ds:[<&PostQueuedCompletionStatus>]

```

Stack dump (stdcall):

```

1: [esp] 00000214
2: [esp+4] 0000FFFE
3: [esp+8] 00000077
4: [esp+C] 0074CC78 L"\\\\?\\Z:\\test.txt"

```

Figure 44

Thread activity – sub_CBBAD0 function

GetQueuedCompletionStatus is utilized to dequeue an I/O completion packet from the IOCP:

```

00C9CE0E 6A FF push FFFFFFFF
00C9CE10 8D 4D 18 lea ecx,dword ptr ss:[ebp+18]
00C9CE13 51 push ecx
00C9CE14 8D 4D 50 lea ecx,dword ptr ss:[ebp+50]
00C9CE17 51 push ecx
00C9CE18 8D 4D 54 lea ecx,dword ptr ss:[ebp+54]
00C9CE1B 51 push ecx
00C9CE1C 50 push eax
00C9CE1D FF 15 A4 D0 D2 00 call dword ptr ds:[<&GetQueuedCompletionStatus>]

```

Stack dump (stdcall):

```

1: [esp] 00000214
2: [esp+4] 045FF7F4
3: [esp+8] 045FF7F0
4: [esp+C] 045FF7B8

```

Figure 45

AvosLocker retrieves file system attributes for a file or directory and avoids the **FILE_ATTRIBUTE_SYSTEM** (0x4) attribute:

```

00C9CECC 50 push eax
00C9CED0 FF 15 9C D0 D2 00 call dword ptr ds:[<&GetFileAttributesW>]
00C9CED3 A8 04 test al,4
00C9CED5 74 28 je malware.C9CEFF
00C9CED7 F0 FF 00 50 64 D5 00 lock dec dword ptr ds:[056450]

```

Stack dump (stdcall):

```

1: [esp] 0074C888 L"\\\\?\\Z:\\test.txt"
2: [esp+4] 138A7825
3: [esp+8] 00738980
4: [esp+C] 0071D150

```

Figure 46

Based on the assembly code we analyzed, the ransomware uses a free C++ library of cryptographic schemes called Cryptopp (<https://github.com/weidai11/cryptopp>):

```

.text:00CDB234 loc_CDB234:
.text:00CDB234 push offset aCryptographicA ; "Cryptographic algorithms are disabled b"...
.text:00CDB239 lea ecx, [ebp+var_28]
.text:00CDB23C call sub_CADA40
.text:00CDB241 lea eax, [ebp+var_28]
.text:00CDB244 mov [ebp+var_4], 0
.text:00CDB24B push eax ; void *
.text:00CDB24C lea ecx, [ebp+var_50]
.text:00CDB24F call sub_CDB7D0
.text:00CDB254 push offset __TI3?AVSelfTestFailure@CryptoPP@@ ; throw info for 'class CryptoPP::SelfTestFailure'
.text:00CDB254 ; throw info for 'class CryptoPP::SelfTestFailure'
.text:00CDB259 lea eax, [ebp+var_50]
.text:00CDB25C push eax
.text:00CDB25D call sub_CF782E

```

Figure 47

The malicious process acquires a handle to a key container within a particular cryptographic service provider (0x1 = **PROV_RSA_FULL**, 0xF0000000 = **CRYPT_VERIFYCONTEXT**):



Figure 48

The ransomware generates 32 random bytes via a function call to CryptGenRandom. These bytes will be used to derive a Salsa20 key and a nonce:

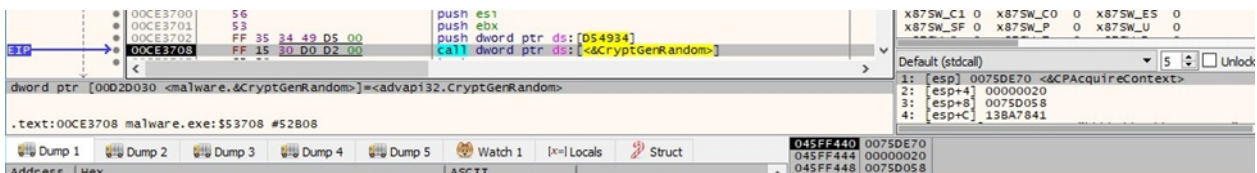


Figure 49

Address	Hex	ASCII
0075D058	3C E0 06 45 0E 94 36 54 B5 B9 F8 C4 8E 5D 5C C4	<ã.E..6Tµ'øA.]\A
0075D068	69 11 83 89 D8 C4 B0 FF 7C C0 F5 C3 B9 DD ED 8A	i...øA'ÿ ÀöA'Yi.

Figure 50

The RSA public key is converted into an array of bytes using CryptStringToBinaryA:

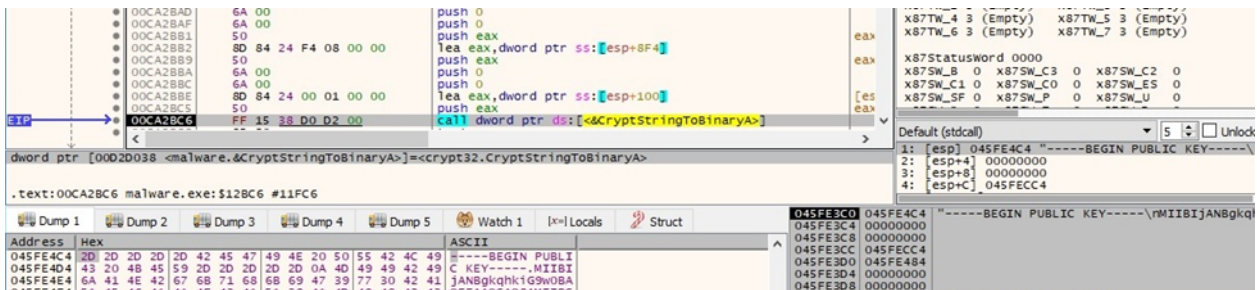


Figure 51

The CryptDecodeObjectEx API is utilized to decode a structure of a particular type (0x1 = X509_ASN_ENCODING, 0x8 = X509_PUBLIC_KEY_INFO, 0x8000 = CRYPT_DECODE_ALLOC_FLAG):

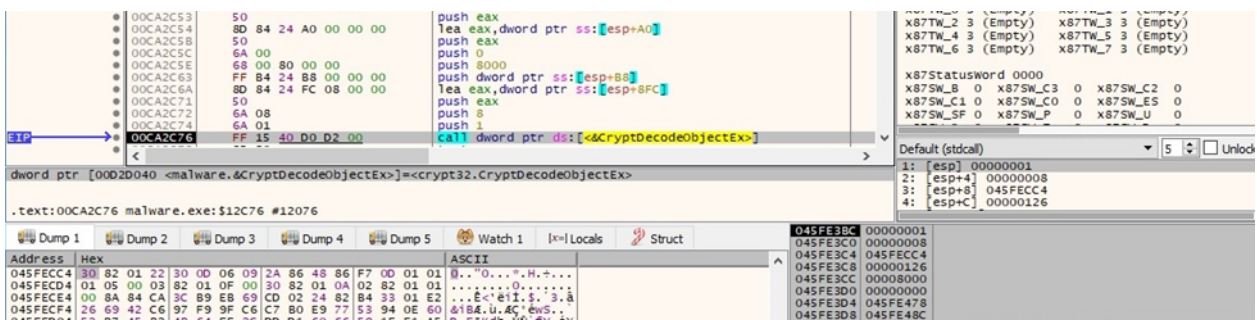


Figure 52

The process converts and imports the RSA public key information into the provider and returns a handle (0x1 = **X509_ASN_ENCODING**):

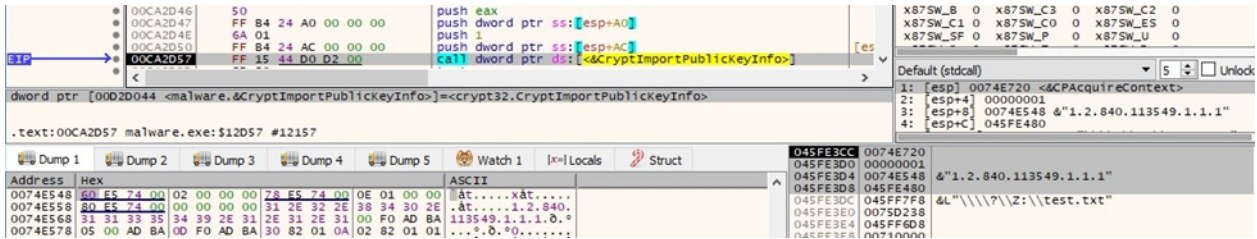


Figure 53

The Salsa20 key (32 bytes) and a nonce (8 bytes) that were derived from the randomly generated buffer are encrypted using the RSA public key:

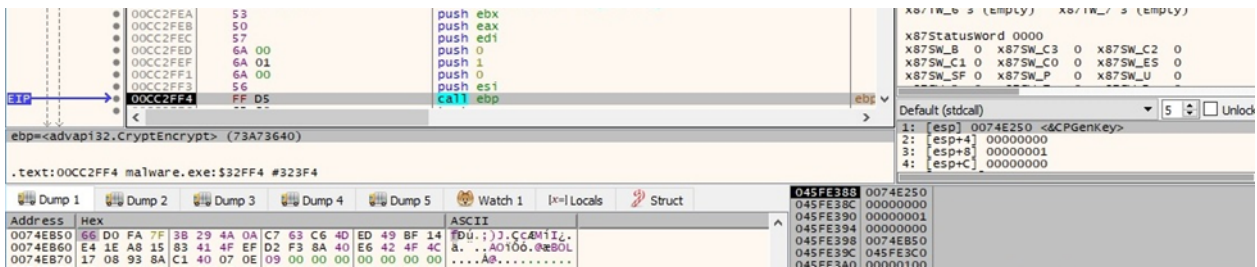


Figure 54

Address	Hex	ASCII
0074EB50	08 31 AD 2B 4A 37 51 DA 02 76 4D 83 2C 74 C6 36	.1.+j7QÜ.VM.,t46
0074EB54	9D D0 F2 BF FE 13 64 79 9E 79 C9 A2 5D 2B CD AB	.Dö2ö.dy.yE\$]~i&
0074EB58	D7 59 F3 15 BE AF 81 C3 B3 6A F5 3B CD B7 69 B5	Xy0.%~A~]0:~1~i~
0074EB5C	9C A1 5F EE 4C C3 88 CE 3E D8 78 38 48 43 C6 0F	.i~i~l.A~i~0\$HCA.
0074EB60	03 51 08 BF EC 8E 72 DE 02 48 8B 4C 11 C4 BE 17	.Q~i~i~r.p.HvL.A&
0074EB64	08 CD 9C 27 12 90 11 FE 82 22 14 54 DE F2 EC 6F	.I~i~i~r.p."Tp0i0
0074EB68	D3 8C A2 C1 01 91 C1 F2 A9 10 2E 6D 87 7C 47 32	O.e.A..Aöb..m.lG2
0074EB6C	17 23 D8 35 C5 2A D8 54 15 2D D7 34 62 86 20 56	.#05Ä*0T.-x4b] V
0074EB70	3D D0 50 39 66 8A 1E C9 21 04 48 BD 0A 7F 9D CB	=0P9f.É!..H5..E
0074EB74	D5 FB EC C3 77 29 92 25 2D 03 60 13 1E DE 33 8B	0üiAw].%~..D.B3.
0074EB78	72 AF 4F 74 48 5A 32 83 51 23 1C 4C 48 66 87 A5	f~0tH2Z*Q#.LkF.¥
0074EB7C	5E 07 FD FD BC 35 FA 56 2C 11 29 9C E3 08 5D 81	^,yy45Üv.,.)&.]
0074EB80	57 37 5D F8 AB 97 70 4F A5 98 BB A7 FD D9 57 EE	W7]ö.k.pöW.¥5yüW
0074EB84	01 BC 30 7F 01 F8 70 D5 EC 69 CE 6E EA 9C 7A 8F	.%0..öPöiInë.z.
0074EB88	32 CF FF 34 93 4A E4 63 AF E8 1A D9 A2 3A AD A3	2Iy4.Jäc'e.Ue:;f
0074EB8C	29 DE B2 38 00 34 2D 2A 5B A2 3F EB 01 86 2E 50]D*:.4~*[e7ë...P

Figure 55

The above buffer is reversed and converted to Base64 format (0x40000001 = **CRYPT_STRING_NOCTRL | CRYPT_STRING_BASE64**):

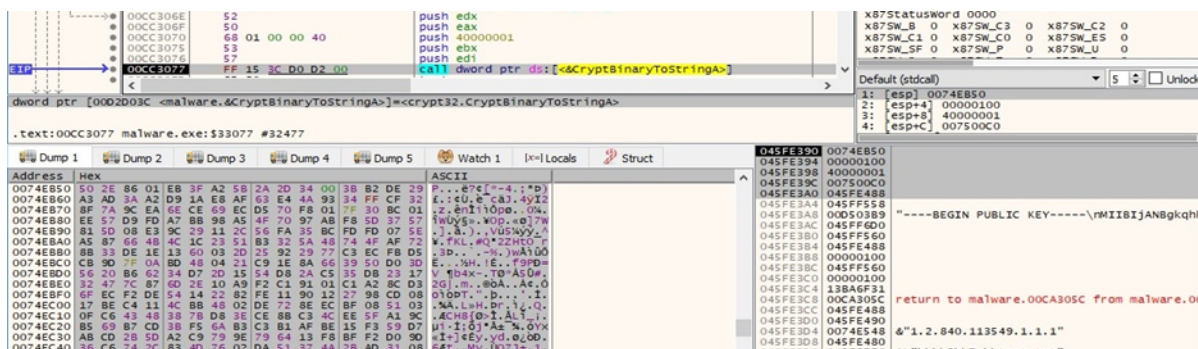


Figure 56

Address	Hex	ASCII
007500C0	55 43 36 47 41 65 73 2F 6F 6C 73 71 4C 54 51 41	UCG6Aes/01sqlTQA
007500D0	4F 37 4C 65 48 61 4F 74 4F 71 4C 5A 47 75 69 76	07LeKa0toqLZGuiv
007500E0	59 2B 52 48 68 7A 54 2F 7A 7A 4B 50 65 70 7A 71	Y+rKkzT/zZkPepzq
007500F0	62 73 35 70 37 4E 56 77 2B 41 46 2F 4D 4C 77 42	bs5p7NVw+Af/MLwB
00750100	37 6C 66 5A 2F 61 65 37 6D 4B 56 50 63 4A 65 72	71Fz/ae7mkVPC3er
00750110	2B 46 30 33 56 34 46 64 43 4F 4F 63 48 52 45 73	+F03v4FdCO0CKRES
00750120	56 76 6F 31 76 50 33 39 42 31 36 6C 68 32 5A 4C	Vv01vP39B161h2ZL
00750130	54 42 77 6A 55 62 4D 79 57 68 68 30 54 36 39 79	TBwjUbmYkwhOT69y
00750140	69 7A 50 65 48 68 4E 67 41 79 30 6C 68 69 6C 33	izPeHNgAy01k1i3
00750150	77 2B 7A 37 31 63 75 64 66 77 71 39 53 41 51 68	w+z71cudfwq9SAqh
00750160	79 52 36 48 5A 6A 6C 51 30 44 31 57 49 4C 5A 69	yRKzJlQ0D1WLZ1
00750170	4E 4E 63 74 46 56 54 59 4B 73 55 31 32 79 4D 58	NnctFVTYksU12yMx
00750180	4D 6B 64 38 68 32 30 75 45 4B 6E 79 77 5A 45 42	Mkd8h20uEKnywZEB
00750190	77 61 4B 4D 30 32 2F 73 38 74 35 55 46 43 4B 43	wakM02/s8t5UFCKK
007501A0	2F 68 47 51 45 69 65 59 7A 51 67 58 76 73 51 52	/hQE1eyZQgXvsQR
007501B0	54 4C 74 49 41 74 35 79 6A 75 79 2F 43 46 45 44	TLtIAt5yJuy/CFED
007501C0	44 38 5A 44 53 44 68 37 32 44 37 4F 69 38 4E 4D	D82SDh72D018NM
007501D0	37 6C 2B 68 6E 4C 56 70 74 38 30 37 39 57 71 7A	71+hnLvpt8079WzD
007501E0	77 37 47 76 76 68 58 7A 57 64 65 72 7A 53 74 64	w7GvvhXzWderZStD

Figure 57

The AllocateAndInitializeSid function is utilized to allocate and initialize a security identifier (SID) with 2 subauthorities:

Figure 58

The malware creates a new ACL by calling the SetEntriesInAcl routine:

Figure 59

SetNamedSecurityInfoW is utilized to modify the DACL of the targeted file (0x1 = SE_FILE_OBJECT, 0x4 = DACL_SECURITY_INFORMATION):

Figure 60

The ransomware opens the file via a call to CreateFileW (0xc0000000 = **GENERIC_READ** | **GENERIC_WRITE**, 0x3 = **OPEN_EXISTING**, 0x80 = **FILE_ATTRIBUTE_NORMAL**):

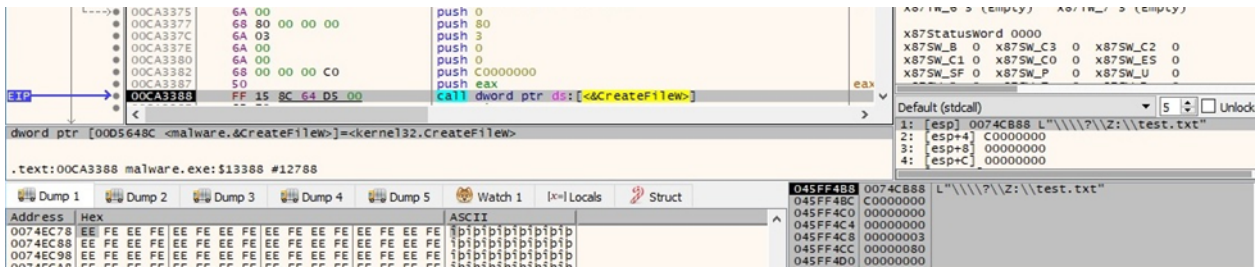


Figure 61

The file size is retrieved using the GetFileSizeEx API:

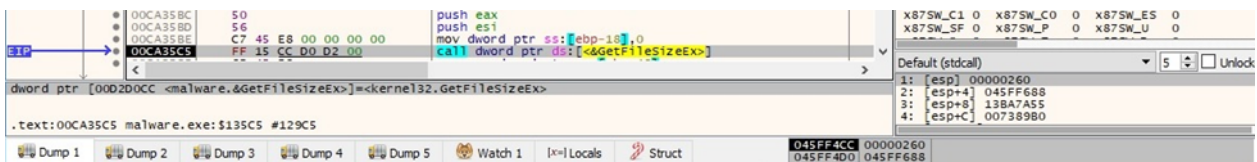


Figure 62

AvosLocker reads 1MB at a time:

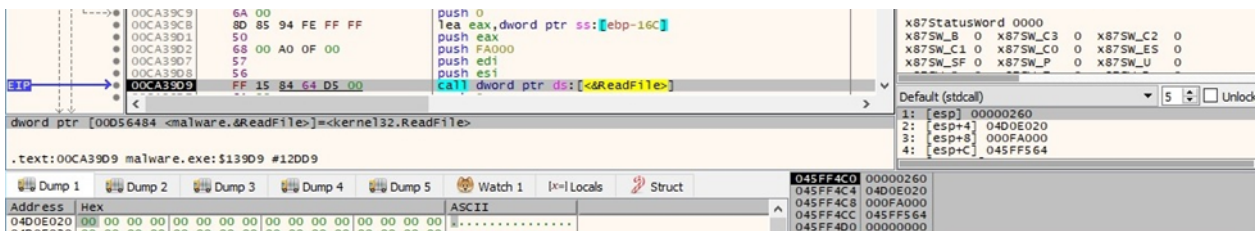


Figure 63

The process moves the file pointer to the beginning of the file by calling SetFilePointer (0x0 = **FILE_BEGIN**):

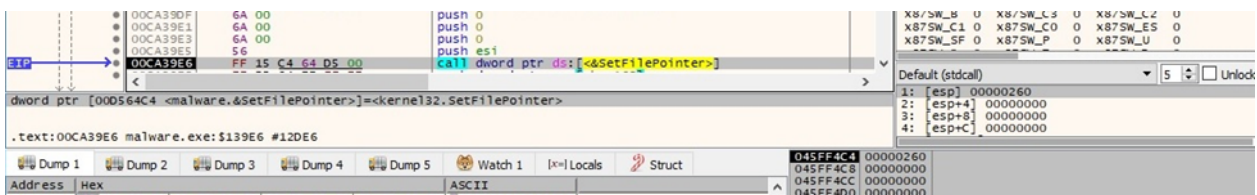


Figure 64

The binary passes a pointer to the file content to the encryption function:

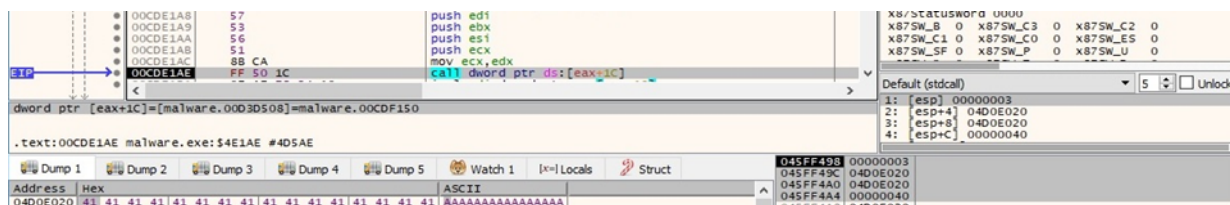


Figure 65

The file content is encrypted using the Salsa20 algorithm. The implementation below is very similar to the one presented at <https://github.com/weidai11/cryptopp/blob/master/salsa.cpp>:

```
.text:00CDF672 loc_CDF672:
.text:00CDF672 movdqa xmm0, [esp+3F0h+var_30C]
.text:00CDF67B movdqa xmm4, [esp+3F0h+var_2FC]
.text:00CDF684 movdqa xmm2, [esp+3F0h+var_3CC]
.text:00CDF68A movdqa xmm6, [esp+3F0h+var_3BC]
.text:00CDF690 paddb xmm0, xmm2
.text:00CDF694 paddb xmm4, xmm6
.text:00CDF698 movdqa xmm1, xmm0
.text:00CDF69C movdqa xmm5, xmm4
.text:00CDF6A0 pslld xmm0, 7
.text:00CDF6A5 pslld xmm4, 7
.text:00CDF6AA psrld xmm1, 19h
.text:00CDF6AF psrld xmm5, 19h
.text:00CDF6B4 pxor xmm0, [esp+3F0h+var_38C]
.text:00CDF6BA pxor xmm4, [esp+3F0h+var_37C]
.text:00CDF6C0 pxor xmm0, xmm1
.text:00CDF6C4 pxor xmm4, xmm5
.text:00CDF6C8 movdqa [esp+3F0h+var_38C], xmm0
.text:00CDF6CE movdqa [esp+3F0h+var_37C], xmm4
.text:00CDF6D4 movdqa xmm1, xmm0
.text:00CDF6D8 movdqa xmm5, xmm4
.text:00CDF6DC paddb xmm0, xmm2
.text:00CDF6E0 paddb xmm4, xmm6
.text:00CDF6E4 movdqa xmm3, xmm0
.text:00CDF6E8 movdqa xmm7, xmm4
.text:00CDF6EC pslld xmm0, 9
.text:00CDF6F1 pslld xmm4, 9
.text:00CDF6F6 psrld xmm3, 17h
.text:00CDF6FB psrld xmm7, 17h
.text:00CDF700 pxor xmm0, [esp+3F0h+var_34C]
.text:00CDF709 pxor xmm4, [esp+3F0h+var_33C]
.text:00CDF712 pxor xmm0, xmm3
.text:00CDF716 pxor xmm4, xmm7
.text:00CDF71A movdqa [esp+3F0h+var_34C], xmm0
.text:00CDF723 movdqa [esp+3F0h+var_33C], xmm4
.text:00CDF72C movdqa xmm3, xmm0
.text:00CDF730 movdqa xmm7, xmm4
.text:00CDF734 paddb xmm0, xmm1
.text:00CDF738 paddb xmm4, xmm5
.text:00CDF73C movdqa xmm1, xmm0
.text:00CDF740 movdqa xmm5, xmm4
.text:00CDF744 pslld xmm0, 0Dh
.text:00CDF749 pslld xmm4, 0Dh
.text:00CDF74E psrld xmm1, 13h
```

Figure 66



Figure 67

The encrypted file content and the encrypted Salsa20 key and nonce are written to the file using WriteFile:

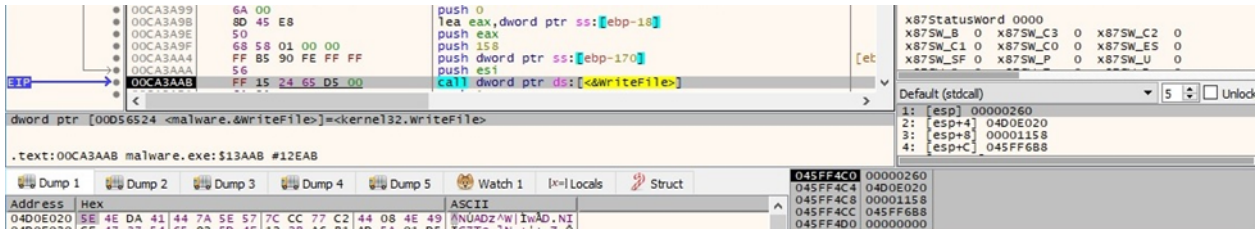


Figure 68

The “.avos2” extension is appended to the encrypted files:

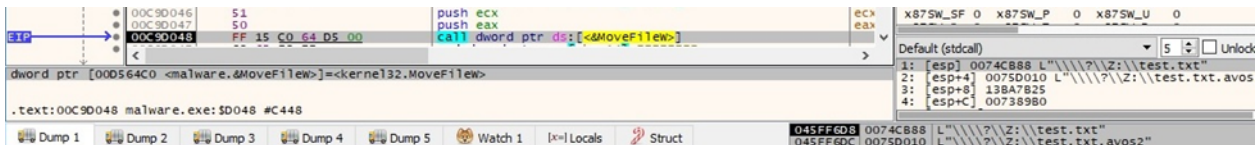


Figure 69

An example of an encrypted file is displayed in figure 70.

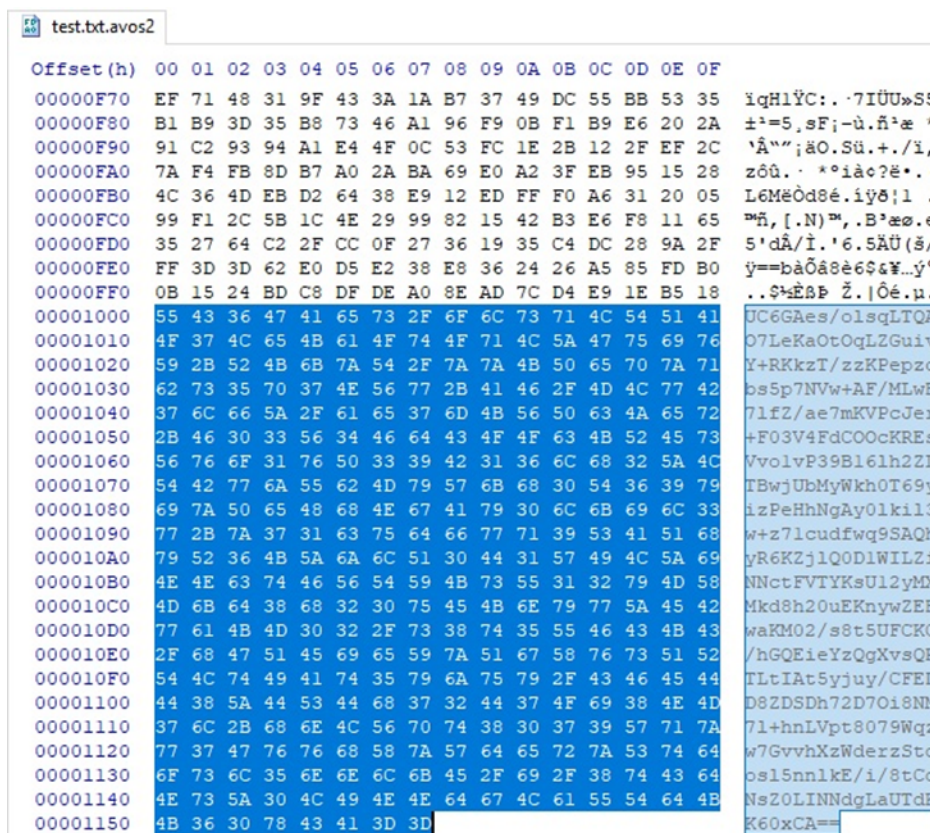


Figure 70

We continue with the analysis of the main thread.

AvosLocker displays some statistics regarding encryption in the command line window:

```

Administrator: C:\Users\...\Desktop\malware.exe
Build: SonicBoom
b_bruteforce_smb_enable: 0
b_logical_disable: 0
b_network_disable: 1
b_mutex_disable: 0
concurrent_threads_num_max: 200
No Instance(s) Available.
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2013 Microsoft Corp.

No items found that satisfy the query.
The operation completed successfully.
The operation completed successfully.
drive: C:
drive: D:
drive: Z:
Waiting for encryption threads to finish ... [OK]
Encryption Stats
[+] Locked objects: 1
asym time: clocks 693521
total time: clocks 16254009 | 16254.009000 seconds

```

Figure 71

The ransomware decrypts and runs a PowerShell script (see figure 72):

- powershell -Command "\$a = [System.IO.File]::ReadAllText(\"Z:\GET_YOUR_FILES_BACK.txt\");Add-Type -AssemblyName System.Drawing;\$filename = \"\\$env:temp\$(Get-Random).png\";\$bmp = new-object System.Drawing.Bitmap 1920,1080;\$font = new-object System.Drawing.Font Consolas,10;\$brushBg = [System.Drawing.Brushes]::Black;\$brushFg = [System.Drawing.Brushes]::White;\$format = [System.Drawing.StringFormat]::GenericDefault;\$format.Alignment = [System.Drawing.StringAlignment]::Center;\$format.LineAlignment = [System.Drawing.StringAlignment]::Center;\$graphics = [System.Drawing.Graphics]::FromImage(\$bmp);\$graphics.FillRectangle(\$brushBg,0,0,\$bmp.Width,\$bmp.Height);\$graphics.DrawString(\$a,\$font,\$brushFg,[System.Drawing.RectangleF]::FromLTRB(0,0,1920,1080),\$format);\$graphics.Dispose();\$bmp.Save(\$filename);reg add \"HKEY_CURRENT_USER\Control Panel\Desktop\" /v Wallpaper /t REG_SZ /d \$filename /f;Start-Sleep 1;rundll32.exe user32.dll, UpdatePerUserSystemParameters, 0, \$false;"

Figure 72

The script's purpose is to create an image that contains the ransom note and set that as the Desktop Wallpaper.

The final image that will be set is highlighted below:

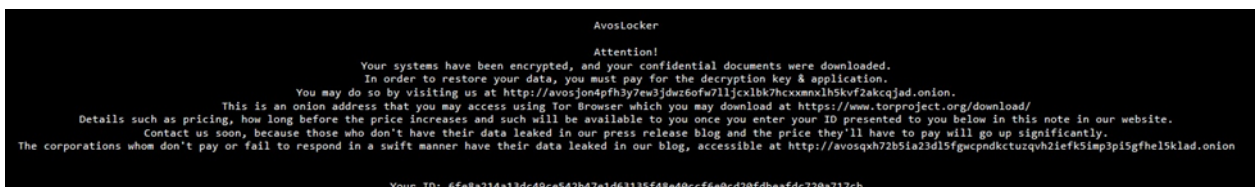
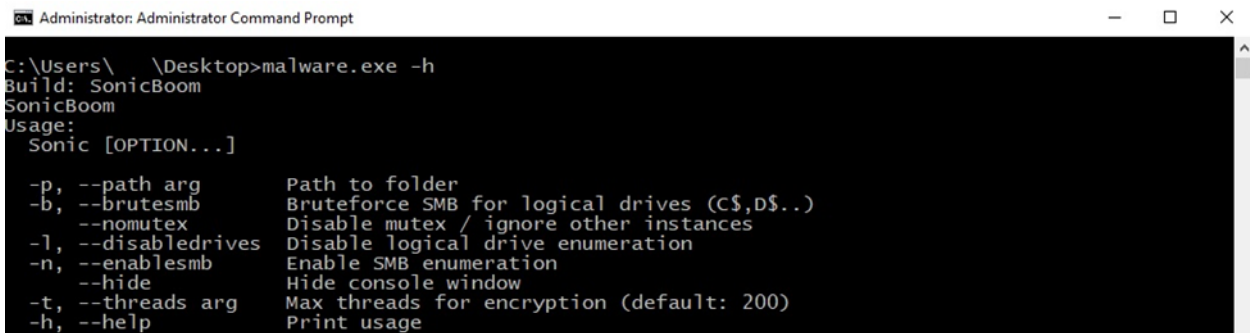


Figure 73

Running with the -h (--help) parameter

AvosLocker displays the help menu:



```
Administrator: Administrator Command Prompt
C:\Users\  \Desktop>malware.exe -h
Build: SonicBoom
SonicBoom
Usage:
Sonic [OPTION...]
-p, --path arg      Path to folder
-b, --brutesmb      Bruteforce SMB for logical drives (C$,D$..)
--nomutex           Disable mutex / ignore other instances
-l, --disabledrives Disable logical drive enumeration
-n, --enablesmb     Enable SMB enumeration
--hide             Hide console window
-t, --threads arg   Max threads for encryption (default: 200)
-h, --help         Print usage
```

Figure 74

Running with the -p (--path) parameter

The ransomware only encrypts this specific directory:



```
Administrator: Administrator Command Prompt
C:\Users\  \Desktop>malware.exe -p test2
Build: SonicBoom
b_bruteforce_smb_enable: 0
b_logical_disable: 0
b_network_disable: 1
b_mutex_disable: 0
concurrent_threads_num_max: 200
mutex: disabled!
path: test2
```

Figure 75

Running with the -l (--disabledrives) parameter

AvosLocker doesn't encrypt the logical drives:

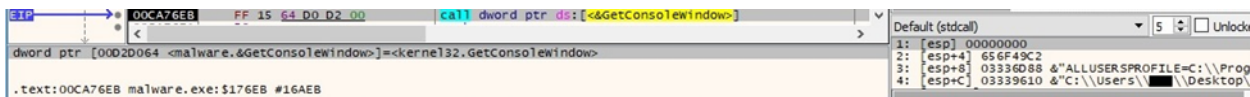


```
Administrator: Administrator Command Prompt
C:\Users\  \Desktop>malware.exe -l
Build: SonicBoom
b_bruteforce_smb_enable: 0
b_logical_disable: 1
b_network_disable: 1
b_mutex_disable: 0
concurrent_threads_num_max: 200
```

Figure 76

Running with the --hide parameter

The malicious executable retrieves the window handle used by the console:



```
EIP 00CA76EB FF 15 64 D0 D2 00 call dword ptr ds:[<&GetConsoleWindow>]
dword ptr [00020064 <malware.&GetConsoleWindow>]=<kernel32.GetConsoleWindow>
.text:00CA76EB malware.exe:$176EB #16AEB
```

Figure 77

The console window is hidden by calling the ShowWindow function (0x0 = **SW_HIDE**):

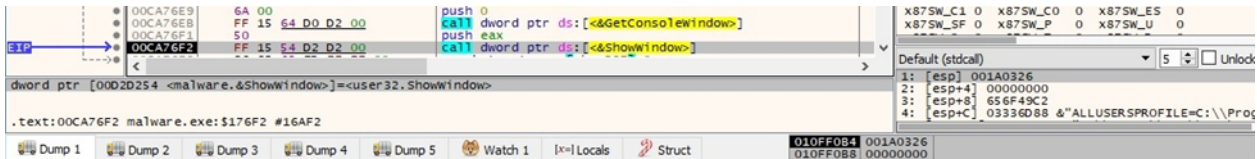


Figure 78

Running with the -t (--threads) parameter

This parameter represents the number of threads that will concurrently encrypt the files:



Figure 79

Running with the -n (--enablesmb) parameter

The ransomware starts enumerating all resources on the network via a function call to WNetOpenEnumA (0x2 = **RESOURCE_GLOBALNET**, 0x0 = **RESOURCETYPE_ANY**):



Figure 80

WNetEnumResourceA is utilized to continue the enumeration of network resources:

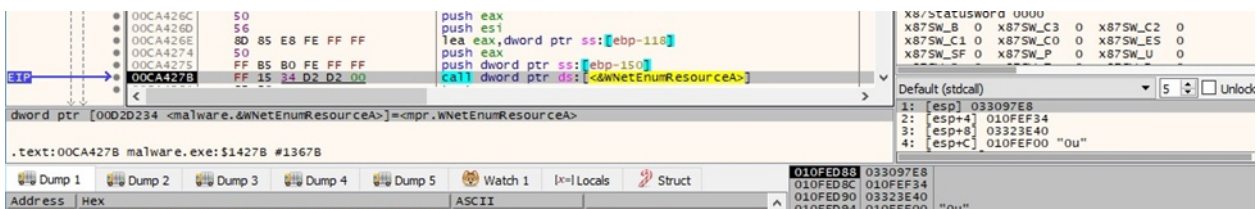


Figure 81

AvosLocker connects to a network share by calling the WNetAddConnection2A routine (0x4 = **CONNECT_TEMPORARY**):

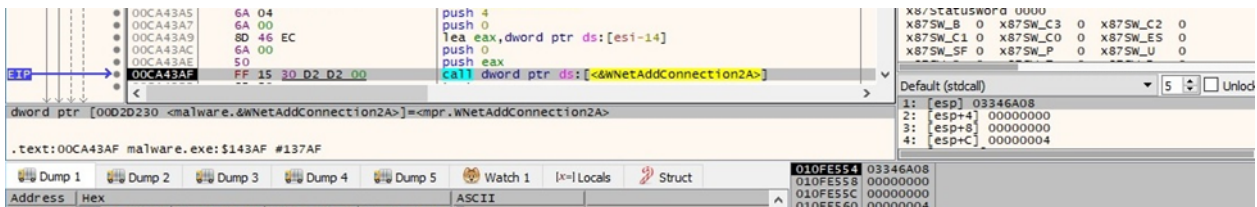


Figure 82

The network share name that will be encrypted is written to the command line:

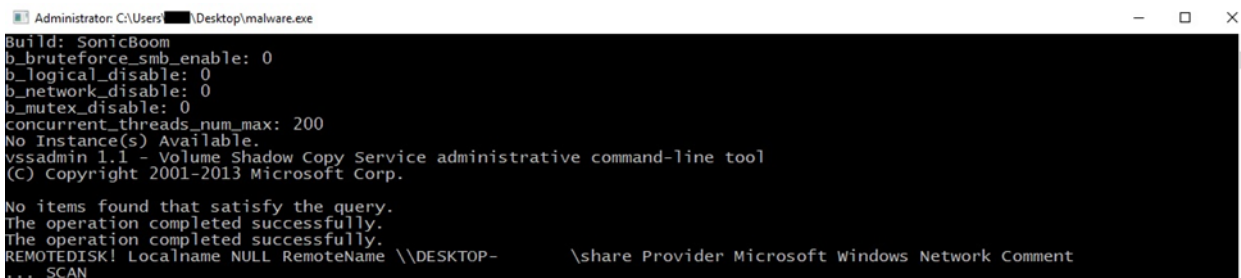


Figure 83

A similar thread that has enumerated logical drives is also created in order to traverse the network shares:

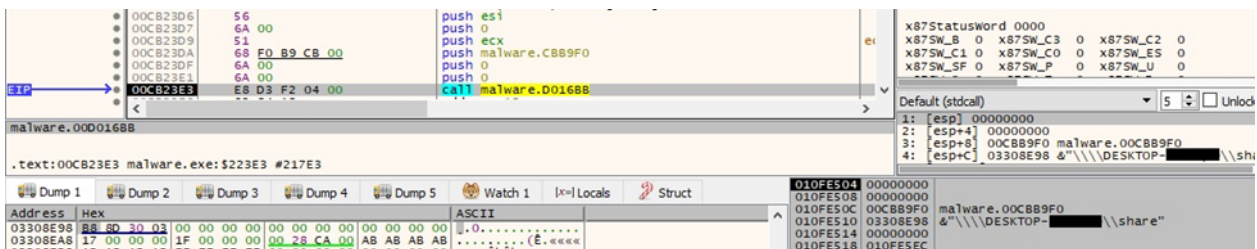


Figure 84

Running with the -b (--brutesmb) -n (--enablesmb) parameters

The process of discovering all network shares is identical to the above. The main idea, in this case, is that the malware is looking to extract the hostname/IP address from an available network share and trying to find logical drives based on it.

The binary makes a connection to a potential logical drive using the WNetAddConnection2A function (0x4 = **CONNECT_TEMPORARY**):

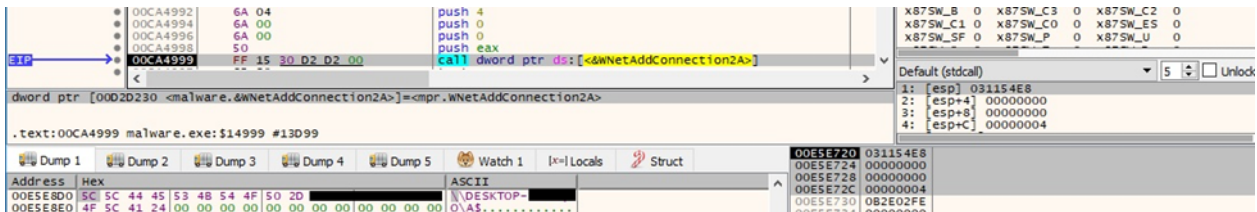


Figure 85

For all logical drives that can be found using the above method, the process creates a new thread that will enumerate them:

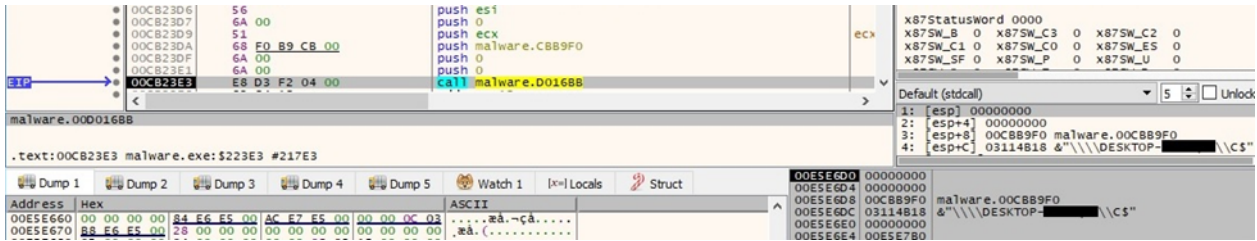


Figure 86

Finally, AvosLocker writes the logical drives that were found to the command line:

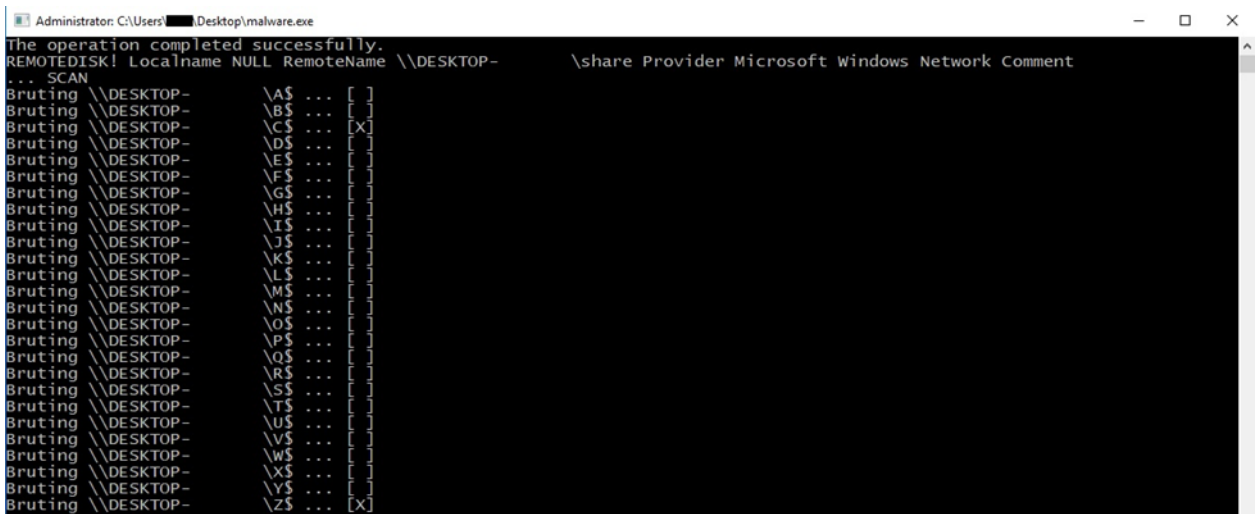


Figure 87

Running with the --nomutex parameter

The ransomware doesn't create the mutex in this case.

Indicators of Compromise

Mutex

Zheic0WaWie6zeiy

AvosLocker Ransom Note

GET_YOUR_FILES_BACK.txt

Processes spawned

cmd /c wmic shadowcopy delete /nointeractive

cmd /c vssadmin.exe Delete Shadows /All /Quiet

cmd /c bcdedit /set {default} recoveryenabled No

cmd /c bcdedit /set {default} bootstatuspolicy ignoreallfailures

cmd /c powershell -command \"Get-EventLog -LogName * | ForEach { Clear-EventLog \$_.Log }\"