A Deep Dive into AvosLocker Ransomware



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 -I (disabledrives) parameter	21
Running with thehide 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 thenomutex 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**):

00CA4089 00CA4084 00CA4084 00CA4085 00CA4085 00CA4095 00CA4095 00CA4095 C	50 FF 15 <u>CO DO D2 00</u> FF 15 <u>CO DO D2 00</u> FF 15 <u>OC DO D2 00</u>	push eax push FOIFF call dword ptr ds:[<&GetCu push eax call dword ptr ds:[<&OpenF	urrentProcess>] ProcessToken>]	>	x875W_B 0 x875W_C3 x875W_C1 0 x875W_C0 x875W_SF 0 x875W_P Default (stdcal) 1: [esp] FFFFFFF	0 x875W_C2 0 0 x875W_E5 0 0 x875W_U 0 0 x875W_U 0
dword ptr [00D2D00C <malu .text:00CA4D96 malware.e:</malu 	ware.&OpenProcessToken>]= <advi xe:\$14D96 #14196 #Dump 3 #Dump 4 #Dump</advi 	p132.OpenProcessToken> 5 🛞 Watch 1 Ix=I Locals	Ø04FF144 004FF148 004FF148	FFFFF 000F0 004FF	2: [esp+4] 000F01FF 3: [esp+8] 004FFAE8 4: [esp+C] 17AA71E1 FFFF 11FF AE8	.

Figure 1

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

		A4DD0 A4DD6 A4DDD A4DDE A4DDE A4DE1 A4DE1 A4DE3	84 30 41 83 ^ 72 C 6	85 84 F9 ED 85	50 F 0D 5 18 69 F	9 FF 1 F9 9 FF	FF	66		m x i c j m	ov al,byte p or byte ptr nc ecx mp ecx,18 b malware.CA ov byte ptr	tr sš: ss:[eb .4DD0 ss:[eb	ebp-6 p+ecx-	80] 6AF],al ,0			>
byte ptr [.text:00CA	ebp-6B0]= 4DD0 malw	[004FF are.ex	F6B4 "3	WAP	EOAKS	JAVW 0	LMTt	WRM	IHAC	A"]	=24 '\$'						
Dump 1	Ump Dump	2	Dump	3		ump 4	k 1	, D	ump	5	💮 Watch 1	[x=] Lo	cals	2 Struct		004FF150 004FF154	00
Address H	lex										ASCII				^	004FF158	00
004FF6B4 2	4 77 41 7	0 45 4	4F 41 (5B 5	3 4A	41 5	6 57	7 4C	4D	54	\$wApEOAkSJA	VWLMT				004FF15C	00
004FF6C4 7	4 56 4D 5	2 4D 4	48 41 4	13 4	1 00	71 0	0 90	0 02	71	00	tVMRMHACA.a	a.				00455464	00
Address H	lex										ASCII						
004FF6B4 2	4 53 65 5	4 61 (6B 65 4	1F 7	7 6E	65 7	2 73	68	69	70	\$SeTakeOwner	rship					
004FF6C4	0 72 69 7	6 69 (6C 65 (57 6	5 00	71 0	0 90	0 02	71	00	Privilege.q	q.					

Figure 2

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

EIP	00C 9478	8 50 52 6A 0 FF 1	0 5 <u>14 D0 D2 0</u>		sh eax sh edx sh 0 11 dword p	tr ds:[<mark><&Lo</mark>	okupPrivile	geValueA>]	ec	i× •	x87SW_C1 0 x87SW_SF 0	x875W_C0 x875W_P	0 x87 0 x87	SW_ES SW_U	Unlock
dword ptr	00D2D014 <ma< td=""><td>lware.&Loo exe:\$478F</td><td>kupPrivilege #388F</td><td>ValueA>]=<ad< td=""><td>api32.Look</td><td>upPrivilege</td><td>eValueA></td><td></td><td></td><td></td><td>1: [esp] 00 2: [esp+4] 3: [esp+8] 4: [esp+C]</td><td>0000000 004FF6B5 004FF124 00000001</td><td>"SeTakeOv</td><td>mership</td><td>pPrivilege"</td></ad<></td></ma<>	lware.&Loo exe:\$478F	kupPrivilege #388F	ValueA>]= <ad< td=""><td>api32.Look</td><td>upPrivilege</td><td>eValueA></td><td></td><td></td><td></td><td>1: [esp] 00 2: [esp+4] 3: [esp+8] 4: [esp+C]</td><td>0000000 004FF6B5 004FF124 00000001</td><td>"SeTakeOv</td><td>mership</td><td>pPrivilege"</td></ad<>	api32.Look	upPrivilege	eValueA>				1: [esp] 00 2: [esp+4] 3: [esp+8] 4: [esp+C]	0000000 004FF6B5 004FF124 00000001	"SeTakeOv	mership	pPrivilege"
Dump 1	Dump 2	Dump 3	🕮 Dump 4	Dump 5	Watch 1	[x=] Locals	2 Struct		004FF114 004FF118 004FF11C	00000 004F	0000 685 "SeTak	eOwnership	Privileg	e''	





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

	<pre>push 0 push 0 push 10 mov dword ptr ss:[esp+1C],eax lea eax,dword ptr ss:[esp+18] push eax push esi mov dword ptr ss: esp+24],1 mov dword ptr ss: esp+24],1 mov dword ptr ss: esp+30,2 call dword ptr ds:[c&AdjustTokenPrivileges>] </pre>		x87TW_0 3 (Empty) x87TW_1 3 (Empty) x87TW_2 3 (Empty) x87TW_3 3 (Empty) x87TW_4 3 (Empty) x87TW_5 3 (Empty) x87TW_6 3 (Empty) x87TW_7 3 (Empty) x87SW_6 0 x87SW_C3 0 x87SW_C2 0 x87SW_5 0 x87SW_C3 0 x87SW_5 0 x87SW_5 0 x87SW_C3 0 x87SW_5 0 x87SW_5 0 x87SW_5 0 x87SW_0 0 x87SW_5 0 x87SW_5 0 x87SW_0 0 x87SW_5 0 x87SW_5 0 x87SW_0 0 x87SW_5 0 x87SW_5 0 x87SW_5 0 x87SW_5 0 x87SW_5 0 x87SW_5 0 x87SW_5 0 x87SW_5 0 x87SW_5 0 x87SW_5 0 x87SW_5 0 x87SW_5 0 x87SW_5 0 x87SW_5 0 x87SW_5 0 x87SW_5
💭 Dump 1 💭 Dump 2 💭 Dump 3 💭 Dump 4 💭 Dump	5 👹 Watch 1 🛛 🕼 Locals 🖉 Struct	004FF108 0000 004FF10C 0000	01D4 0000
Address Hex	ASCII	004FF110 004F 004FF114 0000	F12C 0010
004FF12C 01 00 00 00 09 00 00 00 00 00 00 00 00 02 00 00 004FF13C A1 7D AA 17 9C 4D CA 00 64 FD 4F 00 FD 4D CA	00 i}=MÊ.dýO.ýMÊ.	004FF118 0000	0000

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	He	<							3								ASCII
004FEB18	27	65	6E	63	73	76	63	3B	74	68	65	62	61	74	3B	6D	<pre>encsvc;thebat;m</pre>
004FEB28	79	64	65	73	6B	74	6F	70	71	6F	73	3B	78	66	73	73	ydesktopqos; xfss
004FEB38	76	63	63	6F	6E	3B	66	69	72	65	66	6F	78	3B	69	6E	vccon;firefox;in
004FEB48	66	6F	70	61	74	68	3B	77	69	6E	77	6F	72	64	3B	73	fopath;winword;s
004FEB58	74	65	61	6D	3B	73	79	6E	63	74	69	GD	65	3B	6E	6F	team; synctime; no
004FEB68	74	65	70	61	64	3B	6F	63	6F	6D	GD	3B	6F	6E	65	6E	tepad; ocomm; onen
004FEB78	6F	74	65	3B	6D	73	70	75	62	3B	74	68	75	6E	64	65	ote;mspub;thunde
004FEB88	72	62	69	72	64	3B	61	67	6E	74	73	76	63	3B	73	71	rbird; agntsvc; sq
004FEB98	6C	3B	65	78	63	65	6C	3B	70	6F	77	65	72	70	6E	74	1; excel; powerpnt
004FEBA8	3B	6F	75	74	6C	6F	6F	6B	3B	77	6F	72	64	70	61	64	;outlook;wordpad
004FEBB8	3B	64	62	65	6E	67	35	30	3B	69	73	71	6C	70	6C	75	;dbeng50;isqlplu
004FEBC8	73	73	76	63	3B	73	71	62	63	6F	72	65	73	65	72	76	ssvc;sqbcoreserv
004FEBD8	69	63	65	3B	6F	72	61	63	6C	65	3B	6F	63	61	75	74	ice; or acle; oc aut
004FEBE8	6F	75	70	64	73	3B	64	62	73	6E	GD	70	3B	6D	73	61	oupds;dbsnmp;msa
004FEBF8	63	63	65	73	73	3B	74	62	69	72	64	63	6F	6E	66	69	ccess;tbirdconfi
004FEC08	67	3B	6F	63	73	73	64	3B	GD	79	64	65	73	6B	74	6F	g;ocssd;mydeskto
004FEC18	70	73	65	72	76	69	63	65	3B	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**):

00C 9D 391 00C 9D 393 00C 9D 393 00C 9D 395	6A 00 6A 02 C6 84 24 36 01 00 00 00	<pre>push 0 push 2 mov byte ptr ss:[esp+136],0</pre>		x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0
dword ptr [00020090 cma]war	FF 15 <u>90 D0 D2 00</u>	call dword ptr ds: [<&CreateToolhelp32Snapsh	ot>] / v	Default (stdcall)
.text:00C9D39D malware.exe:	\$D39D #C79D			2: [esp+8] 00000000 3: [esp+8] 00728938 &"C:\\Users\\\\Desktop\ 4: [esp+C] 00000001
Dump 1 Dump 2	Dump 3 👹 Dump 4 👹 Dump	5 🧶 Watch 1 🛛 🕸 I Locals 🖉 Struct	004FEAF0 0000 004FEAF4 0000	00002

Figure 6

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



00C903C2 51 00C903C3 50 00C903C3 50 00C903C4 FF 15 C8 D0 D2 00 00C903C4 FF 15 C8 D0 D2 00	push ecx push eax call dword ptr ds:[<&Process32First>]	× 875W_5F 0 x875W_P 0 x875W_U 0
dword ptr [0002D0C8 <malware.&process32first>]=<kerne< td=""><td>132.Process32First></td><td>1: [esp] 0000020C 2: [esp+4] 004FEC28 3: [esp+8] 00728338 &"C:\\Users\\\\Desktop\ 4: [esp+C] 0000001</td></kerne<></malware.&process32first>	132.Process32First>	1: [esp] 0000020C 2: [esp+4] 004FEC28 3: [esp+8] 00728338 &"C:\\Users\\\\Desktop\ 4: [esp+C] 0000001
Itext:000505C4 Imativare: exe: s05C4 Im	o 5 👹 Watch 1 🕼 Locals 🖉 Struct	04FEAF4 004FEC28

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

	00C9D43 00C9D43 00C9D43 00C9D43 00C9D43	0 50 1 8D 8- 50 9 E8 D2	24 58 01 0 9D 05 00	00 00	push eax lea eax,dword push eax call malware.	ptr ss: [e	sp+158		eax eax	×	x875W_C1 0 x8 x875W_SF 0 x8 Default (stdcall)	7 SW_CO C 7 SW_P C	x87SW_ES x87SW_U	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
.text:00C9	0439 malware.	exe: \$D439	C 839								2: [esp+4] 004 3: [esp+8] 007 4: [esp+C] 000	EDS0 "en 28938 &"C	csvc" :\\Users\\	\\Desktop\
Dump 1	Ump 2	Ump 3	🕮 Dump 4	Dump 5	🛞 Watch 1	[x=] Locals	2 Struct	004FEA	F0 00 F4 00	04FE	C4C "[system p D50 "encsvc"	rocess]"		

Figure 8

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

EIP	• 00C 9D 65 00C 9D 65 00C 9D 65	2 50 3 FF 74 7 FF 15	24 20 80 D0 D2 0	<u>10</u>	oush eax oush dword pt all dword pt	r ss: esp- r ds: (<⪻	20] ocess32Next>]	>	× D	efault (stdcal	x875W_P	0 x87SW_U	5 🗘 Unlock
dword ptr	[00D2D080 <ma< td=""><td>alware.&Pro</td><td>cess32Next>]</td><td>=<kernel32.< td=""><td>Process32Nex</td><td>Þ</td><td></td><td></td><td>2</td><td>: [esp+4]</td><td>004FEC28</td><td></td><td></td></kernel32.<></td></ma<>	alware.&Pro	cess32Next>]	= <kernel32.< td=""><td>Process32Nex</td><td>Þ</td><td></td><td></td><td>2</td><td>: [esp+4]</td><td>004FEC28</td><td></td><td></td></kernel32.<>	Process32Nex	Þ			2	: [esp+4]	004FEC28		
.text:00090	0657 malware.	exe: \$D657	CAS7						3	: [esp+8] : [esp+C]	00728938	&"C:\\Users\	\\Desktop\
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	🛞 Watch 1	[x=] Locals	2 Struct	004FEAF0 00 004FEAF4 00	00002 04FEC	0C 28			



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**):



	D770 FF B D777 8B 4 D777 8B 6A D772 6A 0 D774 6F B D772 6A 0 D774 6F B D775 5B 4 D785 8B 4 D785 8B 0 D788 8B 0 D789 FF 0 D780 FF 0 D780	4 24 38 01 00 5 24 1 4 78 7 0C 10 5 1C 4 88 4 10 2 0 8 8 8 0 8 8 0 8 8 0 8 8 0 8 8 1 1 1 1 1 1 1 1 1 1 1 1 1	00	Such dword pt Nov eax, dword Jush 0 Jush 1 lea eax, dword Nov eax, dword lea eax, dword add eax, edx call eax	r ss: esp i ptr ds: [d d ptr ds: [i ptr ds]]]]]]]]]]]]]]]]]]]	+130] +130] +130] +130] +231 +2 +2 +2 +2 +2 +2 +2 +2 +2 +2		eax eax eax eax eax eax eax eax	x87Tw_2 3 (Empty) x87Tw_3 3 (Empty) x87Tw_4 3 (Empty) x87Tw_5 3 (Empty) x87Tw_6 3 (Empty) x87Tw_5 3 (Empty) x87Tw_8 0 x875w_23 0 x875w_2 0
Dump 1 Dump	2 💭 Dump 3	Dump 4	Dump 5	🛞 Watch 1	[x=] Locals	Struct	004FEA	EC 0000	000001 000000 001448

The TerminateProcess routine is utilized to kill the targeted process:

	NO.829 SD 0.4 7.8 NO.82C OF B7 OC 10 NO.833 SD 0.4 88 0.0 NO.833 SD 0.4 88 0.0	<pre>lea eax,dword ptr ds:[eax+ed1*2] movz eax,dword ptr ds:[eax+edx] mov eax,dword ptr ds:[eax+edx] lea eax,dword ptr ds:[eax+ecx*4] mov eax,dword ptr ds:[eax+edx] add eax,edx call eax</pre>	eax eax eax eax eax	x875tatusWord 0000 x875w_E8 0 x875w_C3 0 x875W_C2 0 x875w_C1 0 x875w_C0 0 x875W_ES 0 x875w_SF 0 x875w_P 0 x875W_U 0 Default (stdcall) ▼ 5 . Unloc
eax= <kernel32.termi .text:00C9D83B malw</kernel32.termi 	nateProcess> (76A766E0) are.exe:\$D83B #CC3B			1: [esp] 00000210 2: [esp+4] 0000000 3: [esp+8] 00728938 &"C:\\Users\\ \ \Desktop\ 4: [esp+c] 0000001
Dump 1 Dump	2 💭 Dump 3 💭 Dump 4	Dump 5 😸 Watch 1 🛛 Ix=I Locals 🖉 Struct	004FEAF0 0000 004FEAF4 0000	00210 00000

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:

EIP	00CA7E46 00CA7E47 00CA7E49 00CA7E49 00CA7E48	50 6A 01 6A 00 FF 15 A0 0	00 D2 00	push eax push 1 push 0 call dword p	otr ds:[<mark><&Cr</mark>	eateMutexA>]	eax	V Defa	SW_C1 0 SW_SF 0	x875W_C0 x875W_P	0 x875W_ES 0 0 x875W_U 0 ▼ 5 ♀ Unlock
dword ptr [00D2D0A0 <malwa< td=""><td>e:\$17E4B #172</td><td>texA>]=<kernel32 4B</kernel32 </td><td>2.CreateMutexA</td><td>></td><td></td><td></td><td>1: 2: 3: 4:</td><td>[esp+4] [esp+8] [esp+C]</td><td>000000 00000001 004FFA45 " L7AA71E1</td><td>'ZheicOwawie6zeiy"</td></malwa<>	e:\$17E4B #172	texA>]= <kernel32 4B</kernel32 	2.CreateMutexA	>			1: 2: 3: 4:	[esp+4] [esp+8] [esp+C]	000000 00000001 004FFA45 " L7AA71E1	'ZheicOwawie6zeiy"
Dump 1	Ump 2	Dump 3 🕮 I	Dump 4 🛛 💭 Dump	5 🛞 Watch 1	[x=] Locals	Struct	004FF144 00 004FF148 00 004FF14C 00	000000 000001 4FFA45	"Zheic	WaWie6zei	У"

Figure 14

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



COCA3CF6 50 OCCA3CF7 FF 15 BS	DO D2 00 call dword ptr ds:	<pre><kwow64disablewow64fsredirection>]</kwow64disablewow64fsredirection></pre>	V De	fault (stdcall)	 ▼ 5 \$ 	Unlock
dword ptr [0002D0B8 <malware.&wow64dis< td=""><td>sablewow64FsRedirection>]=<kernel32.w< td=""><td>ow64Disablewow64FsRedirection></td><td>1: 2: 3: 4:</td><td>[esp] 004FF144 [esp+4] 77056A50 [esp+8] 00000000 [esp+C] FFFFFFE</td><td>ntdll.77056A50</td><td></td></kernel32.w<></td></malware.&wow64dis<>	sablewow64FsRedirection>]= <kernel32.w< td=""><td>ow64Disablewow64FsRedirection></td><td>1: 2: 3: 4:</td><td>[esp] 004FF144 [esp+4] 77056A50 [esp+8] 00000000 [esp+C] FFFFFFE</td><td>ntdll.77056A50</td><td></td></kernel32.w<>	ow64Disablewow64FsRedirection>	1: 2: 3: 4:	[esp] 004FF144 [esp+4] 77056A50 [esp+8] 00000000 [esp+C] FFFFFFE	ntdll.77056A50	
		004FF008 (004FF14	14		

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

Will Dump 1 Will Dump 2 Will Dump 3 Will Dump 5 Weath 1 Ix= Locals Struct OO11EFF43 OO47E011 "cmid /c wmic shadowcopy delete /nointerac Address Hex ASCII Address ASCII OO47EFF43 OO47E011 "cmid /c wmic shadowcopy delete /nointerac		push 0 mov eax,dword ptr ds:[edx+24] lea eax,dword ptr ds:[eax+ed1*2] movz eax,dword ptr ds:[edx+edp] lea eax,dword ptr ds:[eax+ecx+4] mov eax,dword ptr ds:[eax+ebp] lea eax,dword ptr	eax x87TW_2 3 (Empty) x87TW_3 3 (Empty) eax x87TW_4 3 (Empty) x87TW_5 3 (Empty) x87TW_4 3 (Empty) x87TW_5 3 (Empty) eax x87TW_5 3 (Empty) eax x87TW_5 3 (Empty) eax x875W_5 3 (Empty) eax x875W_5 3 (Empty) eax x875W_5 0 0000 eax x875W_5 0 x875W_5 0 0 x875W_5 0 x875W_5 0 x875W_9 0 x875W_9 0 x875W_0 0 eax x875W_5 0 0x875W_9 0 x875W_0 0 eax x855W_5 0 0x875W_9 0 x875W_0 0 x855W_5 0 0x875W_9 0 x875W_0 0 x875W_0 0 x855W_5 0 0x875W_9 0 x875W_0 0 x855W_5 0 0x875W_9 0 x875W_0 0 x855W_5 0 0x875W_9 0 x875W_0 0 x855W_5 0 0x855W_0 0 x875W_0 0 x855W_5 0 0x855W_0 0 x855W_0 0 x855W_0 0 x855W_5 0 0x855W_0 0 x855
erecomp 2 erecomp 2 erecomp 3 erecomp 4 erecomp 3 erecomp 4 erecomp 5 erecomp 4 erecom		5 Mustch 1 [vallegale 9 Struct	EFF4 004FF011 "cmd /c wmic shadowcopy delete /nointeract
Address Hex ASCII	and comb z and comb z and comb 2 and comb 4 and comb	J Water I Prototo g Subtl 004FE	EFF8 00000000
	Address Hex	ASCII	EPPC 00000008
004FF011 [63] 60 64 20 2F 63 20 77 60 69 63 20 73 68 61 64 [m/and /c wmic shad	004FF011 63 6D 64 20 2F 63 20 77 6D 69 63 20 73 68 61	64 gmd /c wmic shad 004FF	F004 004FFD64
004FF021 6F 7/ 65 6F 7/ 65 2 61 63 7/ 69 2 6 6 00 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	004FF031 6F 69 6E 74 65 72 61 63 74 69 76 65 00 00 00	00 ointeractive	F008 00261000

Figure 16

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

Address	He	(ASCII
00D503B9	2D	2D	2D	2D	42	45	47	49	4E	20	50	55	42	4C	49	43	BEGIN PUBLIC
00D503C9	20	4B	45	59	2D	2D	2D	2D	2D	0A	4D	49	49	42	49	6A	KEYMIIBIj
00D503D9	41	4E	42	67	6B	71	68	6B	69	47	39	77	30	42	41	51	ANBgkqhkiG9w0BAQ
00D503E9	45	46	41	41	4F	43	41	51	38	41	4D	49	49	42	43	67	EFAAOCAQ8AMIIBCg
00D503F9	4B	43	41	51	45	41	69	6F	54	4B	50	4C	6E	72	61	63	KCAQEAioTKPLnrac
00D50409	30	43	4A	49	4B	30	4D	77	48	69	0A	4A	GD	6C	43	78	OCJIKOMwHi.JmlCx
00D50419	70	66	35	6E	38	62	48	73	4F	6C	33	55	35	51	4F	59	pf5n8bHs013U5QOY
00D50429	46	4B	33	52	62	4A	4C	5A	50	34	73	33	64	46	67	5A	FK3RbJLZP4s3dFgZ
00D50439	6C	6B	65	34	61	56	78	2F	6A	35	76	66	68	6B	77	6E	lke4aVx/j5vfhkwn
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	GD	59	78	35	4F	4D	32	54	53	49	61	ztqSvmYx50M2TSIa
00D50469	56	36	39	6E	6D	4B	64	30	5A	35	57	35	4F	53	48	63	V69nmKd0Z5W50SHc
00D50479	75	74	73	56	56	65	41	62	6B	39	73	54	55	56	36	45	utsVVeAbk9sTUV6E
00D50489	31	39	42	54	77	59	6A	67	72	4C	6B	2B	OA	50	6A	6A	19BTwYjgrLk+.Pjj
00D50499	2B	46	52	64	43	4D	43	56	59	70	42	6C	33	4C	50	6F	+FRdCMCVYpB13LP0
00D504A9	68	50	53	75	6F	42	4B	5A	6B	31	58	77	34	53	47	6B	kPSuoBKZk1Xw4SGk
00D504B9	4F	36	55	61	2B	4B	4E	79	78	69	67	42	69	4E	GD	73	OGUa+KNyxigBiNms
00D504C9	56	53	65	76	58	4B	65	4B	51	6F	33	73	56	0A	77	4A	VSevXKeKQo3sV.wJ
00D504D9	4A	45	68	52	58	70	65	55	32	75	46	63	34	4B	45	32	JEhRXpeU2uFc4KE2
00D504E9	6C	59	4D	79	72	55	71	66	58	41	72	31	71	61	46	36	lyMyrUqfXAr1qaF6
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	6B	66	41	30	57	0A	45	ZBWoqj70VkfA0W.E
00D50519	69	42	2B	73	31	4A	6C	76	67	79	35	68	59	31	74	2F	iB+s1Jlvgy5hY1t/
00D50529	66	4E	31	51	5A	78	2F	62	37	49	70	68	6B	6A	2F	50	fN1QZx/b7Iphkj/P
00D50539	61	36	6F	72	36	52	4E	39	44	6A	35	53	46	4D	63	6B	aGor GRN9D j5 SFMck
00D50549	GD	77	6C	44	77	67	52	4F	6B	6A	7A	30	37	74	73	0A	mwlDwgROkjz07ts.
00D50559	46	51	49	44	41	51	41	42	OA	2D	2D	2D	2D	2D	45	4E	FQIDAQABEN
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 (0xFFFFFFF = **INVALID_HANDLE_VALUE**):



 00CA7BA8 00CA7BAA 00CA7BAA 00CA7BAA 00CA7BAA 	6A 00 6A 00 6A 00 6A FF	push 0 push 0 push FFFFFFF		x875W_B 0 x875W_C3 x875W_C1 0 x875W_C0 x875W_SF 0 x875W_P	0 x87SW_C2 0 0 x87SW_ES 0 0 x87SW_U 0
dword ptr [00D2D054 <mal< td=""><td>FF 15 54 D0 D2 00</td><td><pre>call dword ptr ds:[<&CreateIoComple >]=<kernel32.createiocompletionport></kernel32.createiocompletionport></pre></td><td>tionPort>] V</td><td>Default (stdcall) 1: [esp] FFFFFFF 2: [esp+4] 00000000</td><td>🔻 💈 🗋 Unlod</td></mal<>	FF 15 54 D0 D2 00	<pre>call dword ptr ds:[<&CreateIoComple >]=<kernel32.createiocompletionport></kernel32.createiocompletionport></pre>	tionPort>] V	Default (stdcall) 1: [esp] FFFFFFF 2: [esp+4] 00000000	🔻 💈 🗋 Unlod
.text:00CA7BB0 malware.ex	e:\$17880 #16F80			3: [esp+8] 00000000 4: [esp+C] 00000000	
Dump 1 Dump 2	Dump 3 💭 Dump 4 💭 Dum	np 5 👹 Watch 1 🛛 🕸 Struct	004FF140 FFFF 004FF144 0000	FFFF 0000	
Address Hex		ASCII	^ 004FF148 0000 004FF14C 0000	0000	

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:

	push edi push o push ecx push malware.CSBADO push 0 push 0 call malware.D016BB	x875tatusword 0000 x875w_B 0 x875w_C3 0 x875w_C2 0 x875w_C1 0 x875w_C0 0 x875w_E5 0 x875w_SF 0 x875w_P 0 x875w_U 0 > Default(stdcal) S ♀ Unlod
malware.00D01688 .text:00C82814 malware.exe:\$22814 #21F14		2: [esp+4] 00000000 3: [esp+8] 00C8BAD0 malware.00C8BAD0 4: [esp+C] 0071D150
Image: Second state	5 🛞 Watch 1 K= Locals 🖉 Struct 004FF0F ASCII 004FF10F 1004FF10F 000FF10F 000FF	00000000 00008AD0 malware.00CBBAD0 007D150 0000000 0000000
	Figure 19	
	push ecx push dword ptr ss:[ebp+18] push eax push dware.D0155F push dword ptr ss:[ebp+6] push dword ptr ss:[ebp+6] call dword ptr ds:[ebp+8]	x875tatusWord 0000 x875W_B 0 x875W_C3 0 x875W_C2 0 x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_S5 0 x875W_P 0 x875W_U 0
dword ptr [00D2D1A8 <malware.&createthread>]=<kernel32 .text:00D01704 malware.exe:\$71704 #70804</kernel32 </malware.&createthread>	.CreateThread>	> Denour(gutan)
Hex Dump 1 Hex Dump 2 Hex Dump 3 Hex Dump 4 Hex Dump 3 Hex Dump 4 Hex Dump 3 Hex Dump 4 Hex Dump 4 <t< td=""><td>5 6001FFCCC 0004FFC0C 004FFC0C 004FFC0C</td><td>00000000 00000000 0000155F malware.00D0155F 00738980 00000000 0004FF0E8</td></t<>	5 6001FFCCC 0004FFC0C 004FFC0C	00000000 00000000 0000155F malware.00D0155F 00738980 00000000 0004FF0E8

Figure 20

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

	push 2 push dword ptr ss:[ebp-276] lea ecx,dword ptr ss:[ebp-126] coll mainare.CAAE40 mov ecx,eax call mainare.C94A90 push eax call dword ptr ds:[c&setThreadPriority>]	~	x875tatusword 0000 x875w.E0 x875w.E2 0 x875w.E1 0 x875w.E2 0 x875w.E1 0 x875w.E2 0 x875w.E1 0 x875w.E2 0 x875w.E5 0 x875w.E9 0 x875w.U 0 Defaul(stdcal) ▼ 5 ↓ Unlock
dword ptr [000200A8 <malware.&setthreadpriority>]=<ker .text:00CA7C42 malware.exe:\$17C42 #17042</ker </malware.&setthreadpriority>	nel32.SetThreadPriority>	148 0000	1: [esp+d] 000000218 2: [esp+d] 00000002 3: [esp+d] 17A471E1 4: [esp+d] 0072C368 &"ALLUSERSPROFILE=C:\\Prog
Dump 1 Dump 2 Dump 3 Dump 4 Dump	5 🛞 Watch 1 [x=] Locals 🖉 Struct 004FF	14C 0000	0002

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:

00CA4C8C 68 00 80 00 00 00CA4C91 57 00CA4692 FF 15 C4 D0 D2 00	push 8000 push edi call dword ptr ds:[<&FindFirstVolumew>]	X875W_SF 0 X875W_P	x875₩_U 0 ▼ 5 € Unlock
dword ptr [0002D0C4 <malware.&findfirstvolumew>]=<kerne .text:00CA4C92 malware.exe:\$14C92 #14092</kerne </malware.&findfirstvolumew>	132.FindFirstVolumew>	2: [esp14] 00008000 3: [esp+8] 00000008 4: [esp+C] 00000008	
Dump 1 Dump 2 Dump 3 Dump 4 Dump 5	🛞 Watch 1 🛛 [x=] Locals 🖉 Struct	004FEF10 0073A0E8 004FEF14 00008000	





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

<u>616</u>	00CA4CA3 00CA4CA3 00CA4CA3 00CA4CA3 00CA4CA3 00CA4CA3 00CA4CB3 00CA4CB3 00CA4CB3 €	50 6A 78 80 44 50 57 FF 15	24 28	20	push eax push 78 lea eax,dword push eax push edi call dword pt	l ptr ss: [e r ds:[<&Ge	sp+28] tVolumePathN	amesForVolumeNam	edi ew>	x87 x87 x87 x87 Defa	StatusWo SW_B 0 SW_C1 0 SW_SF 0	x87SW_C3 x87SW_C0 x87SW_P	0 x87 0 x87 0 x87	SW_C2 SW_ES SW_U	0 0 5 🗢 🗌 Unlock
dword ptr	[00D2D05C <ma< td=""><td>lware.&Get\</td><td>olumePathNa</td><td>amesForVolum</td><td>eNameW>]=<ke< td=""><td>rnel32.GetV</td><td>/olumePathNa</td><td>nesForVolumeNamew</td><td>Þ</td><td>1: 2: 3: 4:</td><td>esp] 00 [esp+4] [esp+8] [esp+C]</td><td>73A0E8 L"\ 004FEF38 00000078 004FEF2C</td><td>\\\?\\V0</td><td>olume{</td><td>17e47829-000</td></ke<></td></ma<>	lware.&Get\	olumePathNa	amesForVolum	eNameW>]= <ke< td=""><td>rnel32.GetV</td><td>/olumePathNa</td><td>nesForVolumeNamew</td><td>Þ</td><td>1: 2: 3: 4:</td><td>esp] 00 [esp+4] [esp+8] [esp+C]</td><td>73A0E8 L"\ 004FEF38 00000078 004FEF2C</td><td>\\\?\\V0</td><td>olume{</td><td>17e47829-000</td></ke<>	rnel32.GetV	/olumePathNa	nesForVolumeNamew	Þ	1: 2: 3: 4:	esp] 00 [esp+4] [esp+8] [esp+C]	73A0E8 L"\ 004FEF38 00000078 004FEF2C	\\\?\\V0	olume{	17e47829-000
.text:00CA4	4CB1 malware.	exe: \$14CB1	#140B1												
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	💮 Watch 1	[x=] Locals	2 Struct	004F8 004F8	F0S 00	73A0E8 4FEF38	r./////	?\	d7e47829	-0000-	0000-0000-1
Address H	ex	45 00 00 0	e ee 10 00		ASCII			^ 004FE	F10 00	000078 4FEF2C					

Figure 23

The volume enumeration continues by calling FindNextVolumeW:

51 P		68 00 57 56 FF 15	80 00 00		sh 8000 sh edi sh esi 11 dword p	tr ds:[<mark><&Fi</mark>	ndNextVolumew>	ec	11 • •	x87SW_C1 0 x87SW_SF 0 Default (stdcall)	x875W_C0 x875W_P	0 x87SW_ES 0 x87SW_U	0 0 5 🗣 🗌 Unlock
dword ptr	00D2D050 <ma< td=""><td>lware.&Find</td><td>iNextVolumew #1410C</td><td>>]=<kernel32< td=""><td>FindNextVo</td><td>lumeW></td><td></td><td></td><td></td><td>1: [esp] 0 2: [esp+4] 3: [esp+8] 4: [esp+C]</td><td>0738DD0 0073A0E8 00008000 000000C8</td><td>L"\\\\?\\Volur</td><td>ne{d7e47829-0</td></kernel32<></td></ma<>	lware.&Find	iNextVolumew #1410C	>]= <kernel32< td=""><td>FindNextVo</td><td>lumeW></td><td></td><td></td><td></td><td>1: [esp] 0 2: [esp+4] 3: [esp+8] 4: [esp+C]</td><td>0738DD0 0073A0E8 00008000 000000C8</td><td>L"\\\\?\\Volur</td><td>ne{d7e47829-0</td></kernel32<>	FindNextVo	lumeW>				1: [esp] 0 2: [esp+4] 3: [esp+8] 4: [esp+C]	0738DD0 0073A0E8 00008000 000000C8	L"\\\\?\\Volur	ne{d7e47829-0
		-	-		20	1	d	00455500	0072	8000			
Dump 1	Dump 2	Dump 3	Ump 4	Ump 5	🐨 Watch 1	[x=] Locals	2 Struct	004FEF10	0073	AOES L"\\\\	?\	[d7e47829-0000	-0000-0000-10
Addesses 110				1	ACC TT	1		004FEF14	0000	8000			

Figure 24

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

OOCAACDB 50 OOCAACDC FF D3 OOCAACDC 83 F8 01 OOCAACE1 74 12 OOCAACE1 74 12	oush eax call ebx cmp eax,1 je malwa <u>re.CA4CFS</u>	eax ebx eax	x875w_CL 0 x875w_CO 0 x875w_ES 0 x875w_SF 0 x875w_P 0 x875w_U 0 Default (stdcall)
ebx= <kernel32.getdrivetypew> (76ACDFA0) .text:00CA4CDC malware.exe:\$14CDC #140DC</kernel32.getdrivetypew>	. 1 🙈 1	FEF14 00/	1: [csp] 004FF30 L 2:\\ 2: [csp+4] 00000010 4: [csp+c] 00261000 FFEF30 L"Z:\\"

Figure 25

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

		57 8D 44 50 FF 15	24 1C <u>78 D0 D2 0</u>		ea eax,dword ush eax all dword p	d ptr ss:[e tr ds:[<&se	sp+1C	Pointw>]	edn eax	V Def	87 SW_C1 0 87 SW_SF 0 fault (stdcall)	x87SW_C0 (x87SW_P (x87SW_ES x87SW_U	5 🗣 🗌 Unlock
dword ptr	[OOD2D078 <mal< td=""><td>ware.&SetV</td><td>olumeMountP #140FB</td><td>ointW>]=<ker< td=""><td>rnel32.SetVo</td><td>lumeMountPo</td><td>intW></td><td></td><td></td><td>2: 3: 4:</td><td>[esp] 00 [esp+4] [esp+8] [esp+C]</td><td>0073A0E8 L"\ 000000C8 00000010</td><td>///?//Volum</td><td>e{d7e47829-0</td></ker<></td></mal<>	ware.&SetV	olumeMountP #140FB	ointW>]= <ker< td=""><td>rnel32.SetVo</td><td>lumeMountPo</td><td>intW></td><td></td><td></td><td>2: 3: 4:</td><td>[esp] 00 [esp+4] [esp+8] [esp+C]</td><td>0073A0E8 L"\ 000000C8 00000010</td><td>///?//Volum</td><td>e{d7e47829-0</td></ker<>	rnel32.SetVo	lumeMountPo	intW>			2: 3: 4:	[esp] 00 [esp+4] [esp+8] [esp+C]	0073A0E8 L"\ 000000C8 00000010	///?//Volum	e{d7e47829-0
Dump 1	Dump 2	Dump 3	🚛 Dump 4	🕮 Dump 5	🛞 Watch 1	[x=] Locals	Struct		004FEF10 00 004FEF14 00	04FEF3 073A0E	0 L"Z:\\' 8 L"\\\\	" ?\d7	e47829-0000	-0000-0000-b(

Figure 26

AvosLocker obtains a bitmask representing the available disk drives:

ODC#19262 FF 15 84 65 D5 00 call dword ptr ds[{<&GetLogicalDrives>]	>	Default (stdcall)	▼ 5 🗘 Unlock
dword ptr [00D56584 <malware.6getlogicaldrives>]=<kernel32.getlogicaldrives></kernel32.getlogicaldrives></malware.6getlogicaldrives>		1: [esp+4] 000000C8 3: [esp+8] 00000010 4: [esp+C] 00261000	
	17AA	7DCD	

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:



	push ecx push eax push malware.C88930 push o push o call malware.D01688	eax x875tatusword 0000 x875W_B 0 x875W_C3 0 x875W_C2 0 x875W_E1 0 x875W_C 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0 Default (stdcall) Cefault (stdcall)
malware.00D0168B .text:00CA2A8B malware.exe:\$12A8B #11E8B		2: [esp+4] 00000000 3: [esp+4] 0008930 malware.00C88930 4: [esp+C]_007359C0 &"C:"
Image: Second	p 5 😻 Watch 1 [x=] Locals 🎾 Struct ASCII 8 AB [SS(Ê.eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee	001F20BC 0000000 004FF0C0 0000000 004FF0C4 00C8B930 malware.00CBB930 004FF0C5 007359C0 &"C:" 004FF0CC 0000000 004FF0CC 00000000 004FF0C0 004FF114
	Figure 28	
	push ecx push dword ptr ss:[ebp+18] push malware.D0155F push dword ptr ss:[ebp+C] push dword ptr ss:[ebp+C] call dword ptr ds:[edp+8]	x875tatusword 0000 x875w_E0 0 x875w_C3 0 x875w_C2 0 x875w_E5 0 x875w_C 0 0 x875w_E5 0 x875w_SF 0 x875w_P 0 x875w_U 0 Default (stdcall) 5 Unio
dword ptr [00D2D1A8 <malware.&createthread>]=<kernel: .text:00D01704 malware.exe:\$71704 #70B04</kernel: </malware.&createthread>	32.CreateThread>	1: [esp] 00000000 2: [esp+4] 0000000 3: [esp+8] 00D0155F malware.00D0155F 4: [esp+C]_00738C00
Image: Constraint of the state of	p 5	00155000 004FF094 0000000 004FF098 000155F malware.00D0155F 004FF09C 00738C00 004FF0A0 00000000 004FF0A4 004FF0AC

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:

EIP 00CA9114 FF 15 64 D0 D2 00	call dword ptr ds: [<&GetConsoleWindow>]	> ×	Default (stdcall)
dword ptr [0002D064 <malware.&getconsolewindow>]=<ker .text:00CA9114 malware.exe:\$19114 #18514</ker </malware.&getconsolewindow>	nel32.GetConsoleWindow>		1: [esp+4] 17A471E1 2: [esp+4] 17A471E1 3: [esp+8] 0072C368 &"ALLUSERSPROFILE=C:\\Progr 4: [esp+C] 00728938 &"C:\\Users\\\\\\Desktop\\

Figure 31

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



• 00CA9113 50 • 00CA9114 FF 15 <u>64 D0 D2 00</u> • 00CA911A 50 50 • 00CA911A FF 15 <u>50 D2 D2 00</u> • • • • • • • • • • • • • • • • • • • • • • • • </th <th>push eax call dword ptr ds:[<&GetConsolewindow>] push eax call dword ptr d:[<&ShutdownBlockReasonCreate:]=<user32.shutdownblockreasoncreate></user32.shutdownblockreasoncreate></th> <th>>]</th> <th>x87 x87 Defau 1: [2:] 3: [4:]</th> <th>SW_C1 0 X87SW_C0 0 X87SW_E5 0 SW_SF 0 X87SW_P 0 X87SW_U 0 it (stdcall) ▼ 5 0 Unlod esp1 001600F2 esp+1 0074C288 esp+6] 10A74E1 esp+6] 10A74E1 esp+6] 10A74E1</th>	push eax call dword ptr ds:[<&GetConsolewindow>] push eax call dword ptr d:[<&ShutdownBlockReasonCreate:]= <user32.shutdownblockreasoncreate></user32.shutdownblockreasoncreate>	>]	x87 x87 Defau 1: [2:] 3: [4:]	SW_C1 0 X87SW_C0 0 X87SW_E5 0 SW_SF 0 X87SW_P 0 X87SW_U 0 it (stdcall) ▼ 5 0 Unlod esp1 001600F2 esp+1 0074C288 esp+6] 10A74E1 esp+6] 10A74E1 esp+6] 10A74E1
Image: Dump 1 Image: Dump 2 Image: Dump 3 Image: Dump 4 Image: Dump 5 Address Hex 0074cC38 S80 00 6F 00 75 00 72 00 20 00 64 00 6F 00 63 00 0074cC38 S80 00 6F 00 65 00 66 00 74 00 20 00 77 00 20 00 77 00 0074cC38 6F 00 72 00 70 70 72 00 20 00 77 00 20 00 77 00 20 00 77 00 0074cC38 6F 00 72 00 72 00 72 00 70 06 62 00 70 07 70 06 70 07 70 06 70 06 75 00 0074cC38 65 00 66 00 75 00 72 00 70 06 62 00 74 00 64 00 67 00 75 00 72 00 70 06 64 00 64 00 67 00 75 00 72 00 70 07 74 00 64 00 65 00 70 07 74 00 64 00 67 00 72 00 72 00 74 00 64 00 65 00 72 00 72 00 74 00 64 00 65 00 72 00 72 00 64 00 65 00 72 00 72 00 74 00 64 00 67 00 72 00 72 00 74 00 64 00 67 00 72 00 72 00 74 00 64 00 67 00 72 00 72 00 74 00 64 00 67 00 72 00 72 00 72 00 72 00 74 00 74 00 64	Image: Struct Image: Struct ASCII Image: Struct Mound: Struct Image: Struct Image: Struct Image: Struct	OdfF145 O01 004F14C 007 004F150 17A 004F150 17A 004F150 17A 004F150 007 004F160 000 004F164 007 004FF165 000 004FF164 007 004FF164 007 004FF164 007 004FF164 007 004FF164 007 004FF174 007 004FF174 007 004FF174 007 004FF174 007 004FF174 007 004FF174 007	E00F2 4CC38 A71E1 2C368 28938 61000 00000 10000 0ED78 0007F FF348 10000 23D2E 46900	&"ALLUSERSPROFILE=C:\\ProgramData" &"C:\\Users\\\\\Desktop\\malware.exe" return to ntdll.7700ED78 from ntdll.7700F return to ntdll.77023D2E from ntdll.memse

The ransomware extracts the identifier of the calling thread:

EIP 000094A49 E8 13 36 03 00	call <malware.getcurrentthreadid></malware.getcurrentthreadid>	>	Default (stdcall)
<malware.getcurrentthreadid> .text:00C94A49 malware.exe:\$4A49 #3E49</malware.getcurrentthreadid>			1: [csp+4] 00CA918C malware.00CA918C 3: [csp+4] 17AA71E1 4: [csp+6] 17AA71E1 4: [csp+c] 0072C368 &"ALLUSERSPROFILE=C:\\Progr

Figure 33

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

	1
🚺 🛃 🖼	State and the second se
.text:00C94A53 push	0 ; int
.text:00C94A55 push	dword ptr [esi+4] ; int
.text:00C94A58 push	dword ptr [esi] ; hHandle
.text:00C94A5A call	Thrd_join
.text:00C94A5F add	esp, 0Ch
.text:00C94A62 test	eax, eax
.text:00C94A64 jnz	short loc C94A7B

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	He	ĸ															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
049BE60D	65	70	61	63	6B	20	6E	6C	73	20	64	69	61	67	70	6B	epack nls diagpk
049BE61D	67	20	6D	73	69	20	6C	6E	6B	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	6B	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	11 msstyles mod
049BE68D	70	73	31	20	69	63	73	20	68	74	61	20	62	69	6E	20	ps1 ics hta bin
049BE69D	63	6D	64	20	61	6E	69	20	33	38	36	20	6C	6F	63	6B	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	6B	74	68	65	6D	65	70	61	63	6B	20	73	deskthemepack s
049BE6CD	68	73	20	6C	64	66	20	74	68	65	GD	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
0498E6ED	20	61	64	76	20	69	63	60	20	60	73	75	00	00	00	00	adv icl msu





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 (0x40000000 = **GENERIC_WRITE**, 0x2 = **CREATE_ALWAYS**, 0x80 = **FILE_ATTRIBUTE_NORMAL**):

dword .text:	ptr [000	OOC 9D E89 OOC 9D E88 OOC 9D EC0 OOC 9D EC2 OOC 9D EC4 O	6A 68 6A 6A 68 50 FF vare.&Cr	0 00 00 00 0 00 00 0 00 00 0 00 00 0 00 00 0 F7 00 15 8C 64 eateFile #D2D1	00 40 00 D5 00 W>]=<	2 (erne]32.Cr	push (push (push (push (push (call (call (eateF	0 80 2 0 40000000 malware eax dword p	0 .CAD6CO tr ds:[<&Crea	teFilew>]	 _	e	ax >	x875 x875 x875 x875 x875 x875 x875 x875	W_6 3 tatusW W_B 0 W_C1 0 W_C1 0 W_SF 0 t(stdcall esp] 0 tsp+4] esp+8] esp+c]	(Empty (empty x875 x875 x875 x875 x875 x875 x875 x875	00 W_C3 W_C0 W_P S_L"\\ 000 000	0 x 0 x 0 x	7 3 (EI 875W_C 875W_E 875W_U 875W_U	2 0 5 0 5) DUR_FI	Jnlod LES_
📖 Du	mp 1 🚦	Dump 2	Dump 3	💭 Du	mp 4	Dump 5	۲	Watch 1	[x=] Lo	cals	2 Struct	0	04960008 0498000C	0074	CBD 8 0000	r.////	\?\\Z:\	GET_	YOUR_	FILES_	BAC	.txt"	
Addres	S Hex	0 74 00 25 0	00 50 00	EA 00.1	A 00 0	00.00.00.00	ASCI					 ^ 0	0498DD10 0498DD14	0000	0000								
049BE5 049BE5	CC 00 0 DC 5A 0	0 00 00 07 0 0 3A 00 00 07		00 00 7	4 00	3F 00 5C 00 07 00 00 00	z.:.		.?.\.			000	0498DD18 0498DD1C 0498DD20	0000	0002 0080 0000								

Figure 37

The WriteFile routine is used to populate the ransom note:

OCC9E3BE 6A 00 OCC9E3C0 50 OCC9E3C1 FF B5 C0 00 00 00 OCC9E3C1 FF B5 C0 00 00 00 OCC9E3C2 E8 FE F4 00 00 OCC9E3C2 50 OCC9E3C3 OCC9E3C3	push o push eax push eax lea ecx.dword ptr ss:[ebp+C0] lea ecx.dword ptr ss:[ebp+B0] call malware.CADSOO push esx push esx push esx call dword ptr ds:[<dwritefile>] iteFile></dwritefile>	>	xe/rm_o ≥ (cmp/cy) xe/rm_o ≥ (cmp/cy) x875tatusword 0000 x875w_C3 x875w_C2 0 x875w_E1 0 x875w_C3 0 x875w_C5 0 x875w_E1 0 x875w_C0 0 x875w_C5 0 x875w_C5 0 x875w_E5 0 x875w_C0 0 x875w_C1 0 x875w_C1 0 befault(stdcal) v 5 0 x875w_C1 0 x875w_C1 0 1: (sep) 00000234 : 5 : Unlock 1: (sep+0) 0003873 : : spep-4) 0498E204
.text:00C9E3D4 malware.exe:\$E3D4 #D7D4		1	
💭 Dump 1 👹 Dump 2 👹 Dump 3 👹 Dump 4 👹 Dump	5 🛞 Watch 1 🛛 🕸 Locals 🖉 Struct 049	BDD10 00000	234
Address Hex	ASCII ^ 049 74 Byost ocker AT	BDD18 00000 BDD1C 04988	3F3 204
	049	BDD20 00000	000



1	AvosLocker
2	
3	Attention!
- 4	Your systems have been encrypted, and your confidential documents were downloaded.
5	In order to restore your data, you must pay for the decryption key & application.
6	You may do so by visiting us at http://avosjon4pfh3y7ew3jdwz6ofw7lljcxlbk7hcxxmnxlh5kvf2akcqjad.onion.
7	This is an onion address that you may access using Tor Browser which you may download at https://www.torproject.org/download/
8	Details such as pricing, how long before the price increases and such will be available to you once you enter your ID presented to you below in this note in our website.
9	Contact us soon, because those who don't have their data leaked in our press release blog and the price they'll have to pay will go up significantly.
10	The corporations whom don't pay or fail to respond in a swift manner have their data leaked in our blog, accessible at http://avosqxh72b5ia23d15fgwcpndkctuzqvh2iefk5imp3pi5gfhe15klad.onior
11	
12	
13	Your ID: 6fe8a214a13dc49ce542b47e1d63135f48e40ccf6e0cd20fdbeafdc720a717cb

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	He	<			_												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	(Microsoftcon
049BEAA8	66	69	67	2E	GD	73	69	00	3D	41	6C	6C	20	55	73	65	fig.msi.=All Use
0498EAB8	72	73	00	00	48	62	6F	6F	74	6D	67	72	00	00	00	00	rsHbootmgr
049BEAC8	6B	41	70	70	44	61	74	61	00	00	00	00	5 B	57	69	6E	kAppData[Win
0498EAD8	64	6F	77	73	00	00	00	00	09	4D	69	63	72	6F	73	6F	dowsMicroso
049BEAE8	66	74	2E	00	73	50	75	62	6C	69	63	00	3C	53	6F	70	ftsPublic. <sop< th=""></sop<>
049BEAF8	68	6F	73	00	27	49	6E	74	65	6C	00	00	75	57	69	6E	hos.'InteluWin
049BEB08	4E	54	00	00	1F	62	6F	6F	74	00	00	00	00	00	00	00	NTboot

AvosLocker continues the file enumeration using FindNextFileW:

EIP	00CA20C0 00CA20C1 00CA20C2	50 56 FF 15	88 64 D5 0	0	ush eax ush esi all dword pt	tr ds:[<mark><&Fi</mark>	ndNextFileWa		~	x87SW_SF 0 x87SW_P Default (stdcall)	0 x875W_U	Unlock
dword ptr [.text:00CA2	00056488 <mai< td=""><td>lware.&Find</td><td>NextFileW>] #114C2</td><td>=<kernel32.f< td=""><td>indNextFile</td><td>W></td><td></td><td></td><td></td><td>1: [esp] 0074CB88 2: [esp+4] 049BEC38 3: [esp+8] 137E7091 4: [esp+C] 049BFC60</td><td></td><td></td></kernel32.f<></td></mai<>	lware.&Find	NextFileW>] #114C2	= <kernel32.f< td=""><td>indNextFile</td><td>W></td><td></td><td></td><td></td><td>1: [esp] 0074CB88 2: [esp+4] 049BEC38 3: [esp+8] 137E7091 4: [esp+C] 049BFC60</td><td></td><td></td></kernel32.f<>	indNextFile	W>				1: [esp] 0074CB88 2: [esp+4] 049BEC38 3: [esp+8] 137E7091 4: [esp+C] 049BFC60		
Dump 1	Ump 2	Dump 3	📖 Dump 4	Dump 5	🛞 Watch 1	[x=] Locals	2 Struct	049BE224 049BE228	0074 0498	CB88 EC38		

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	He	ĸ			5				500				9				ASCII
049BEB59	47	45	54	5F	59	4F	55	52	5F	46	49	4C	45	53	5F	42	GET_YOUR_FILES_B
049BEB69	41	43	4B	2E	74	78	74	00	00	00	00	40	6E	74	75	73	ACK.txt@ntus
049BEB79	65	72	2E	64	61	74	2E	6C	6F	67	00	5D	62	6F	6F	74	er.dat.log.]boot
049BEB89	66	6F	6E	74	2E	62	69	6E	00	00	00	7B	69	63	6F	6E	font.bin{icon
049BEB99	63	61	63	68	65	2E	64	62	00	00	00	18	62	6F	6F	74	cache.dbboot
049BEBA9	73	65	63	74	2E	62	61	6 B	00	00	00	2B	61	75	74	6F	sect.bak+auto
049BEBB9	72	75	6E	2E	69	6E	66	00	00	00	00	11	64	65	73	6B	run.infdesk
049BEBC9	74	6F	70	2E	69	6E	69	00	00	00	00	62	6E	74	75	73	top.inibntus
049BEBD9	65	72	2E	64	61	74	00	88	CB	74	00	3D	6E	74	75	73	er.datEt.=ntus
049BEBE9	65	72	2E	69	6E	69	00	47	54	68	75	6D	62	73	2E	64	er.ini.GThumbs.d
049BEBF9	62	00	00	63	74	68	75	6D	62	73	2E	64	62	00	00	16	bcthumbs.db
049BEC09	62	6F	6F	74	2E	69	6E	69	00	00	00	04	47	61	6D	65	boot.iniGame

Figure 42

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

OOCA1E29 OOCA1E30	OF 1F 80 00 00 00 00 8A 08	<pre>nop dword ptr ds:[eax],eax mov cl.byte ptr ds:[eax]</pre>	eax ^	Hide FPU
ET2 → 00CALE92 00CALE35 ↓ 00CALE35 ↓ 00CALE38 ↓ 00CALE38 ↓ 00CALE30 ↓ 00CALE40 ↓ 00CALE40 ↓	3A 0E 75 1A 84 C9 74 12 8A 48 01 3A 4E 01 75 0E 83 C0 02	<pre>cmp cl_byte ptr d::[ess] ime maiware.CAIES0 test cl.cl jme maiware.CAIE40 mov cl.byte ptr d::[ess+1] ime maiware.CAIE50 add eax.2</pre>	eax esi eax	EAX 0498EE88 "txt" EBX 00738E29 ECX 0498EE74 EDX 00000003 EBP 0498FC14 ESI 0498F708 "avos"

Figure 43

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





Thread activity – sub_CBBAD0 function

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

• 00C •	CEDE 6A CCE10 8D CCE13 51 CCE14 8D CCE17 51 CCE18 51 CCE18 51 CCE10 FF CCE10 FF CCE10 FF	4D 18 4D 50 4D 54 1 <u>5 A4 D0 D2 0</u>	0 tionStatus>]	ush ecx, dword ush ecx ea ecx, dword ush ecx ea ecx, dword ush ecx ush eax all dword pt = <kernel32.4< th=""><th>l ptr ss:[el l ptr ss:[el l ptr ss:[el r ds:[<&Get GetQueuedCo</th><th>bp+18] bp+50] bp+54] tQueuedCompli mpletionStat</th><th>etionStatus>] :us></th><th>></th><th>×87 ×87 ×87 ×87 ×87 ×87 ×87 ×87 ×87</th><th>TW_6 3 (Empty) 'StatusWord 0000 'SW_B 0 x875W_ 'SW_SF 0 x875W_ 'SW_SF 0 x875W_ 'SW_SF 0 x875W_ ult(stdcall) [esp+4] 045FFF7F (esp+4) 045FF7F7</th><th>x87Tv C3 0 C0 0 P 0</th><th>L_7 3 (Empt x875W_C2 x875W_E5 x875W_U ▼ 5</th><th>y) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</th></kernel32.4<>	l ptr ss:[el l ptr ss:[el l ptr ss:[el r ds:[<&Get GetQueuedCo	bp+18] bp+50] bp+54] tQueuedCompli mpletionStat	etionStatus>] :us>	>	×87 ×87 ×87 ×87 ×87 ×87 ×87 ×87 ×87	TW_6 3 (Empty) 'StatusWord 0000 'SW_B 0 x875W_ 'SW_SF 0 x875W_ 'SW_SF 0 x875W_ 'SW_SF 0 x875W_ ult(stdcall) [esp+4] 045FFF7F (esp+4) 045FF7F7	x87Tv C3 0 C0 0 P 0	L_7 3 (Empt x875W_C2 x875W_E5 x875W_U ▼ 5	y) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
taxt1000000510 mal		#6310							3: 4:	[esp+8] 045FF7F [esp+C] 045FF7B	D 8		
.text.oocsceib many	are.exe.sceib	#C210											
Dump 1 Dump	2 💭 Dump 3	Dump 4	Dump 5	💮 Watch 1	[x=] Locals	2 Struct	049	FF6D0 0	0000214 45FF7F4				
Address Hex				ASCII			A 045	FF6D4 0	45FF7F0				
045FF7F0 64 F8 SF 0	4 D0 76 CF 00	ED 47 31 17	FE FF FF FF	doDvI.iG1	. Þ <u>ý</u> ÿÿ		049	SFF6DC FI	FFFFFFF				

Figure 45

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

OOC SECECT 50 OOC SECECT 50 OOC SECECT 45 92 D0 D2 00 OOC SECECT 45 04 04 OOC SECECT 45 04 05 00 OOC SECECT 7 74 28 OOC SECECT 7 60 FF 00 50 64 D5 00	push eax call dword ptr ds:[<&GetFileAttributesw>] test al.4 js malware.CSCEFE lock dec dword ptr ds:[D56450]	eax v	x875W_B 0 x875W_C3 x875W_C1 0 x875W_C0 x875W_SF 0 x875W_P	0 x875W_C2 0 0 x875W_ES 0 0 x875W_U 0
<pre>dword ptr [0002D09C <malware.&getfileattributesw>]=< .text:00C9CECD malware.exe:\$CECD #C2CD</malware.&getfileattributesw></pre>	kernel32.GetFileAttributesW>	>	1: [esp] 0074CB88 L"\' 2: [esp+4] 13BA7B25 3: [esp+8] 00738980 4: [esp+C] 0071D150	\\\?\\Z:\\test.txt"
	(A)	45FF60C 0074	CB88 L"\\\\?\\Z:\\tes!	t.txt"

Figure 46

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

📕 🛃 🖼	
.text:00CDB234	
.text:00CDB234 loc_C	DB234:
.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	offsetTI3?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**):



	push F0000000 push 0 push 0 push 0 push si mov dword ptr ds:[esi],0 call edi	x87Statusword 0000 x87Sw_B 0 x87Sw_C2 0 x87Sw_S 0 x87Sw_C2 0 x87Sw_S 0 x87Sw_C2 0 x87Sw_S 0 x87Sw_C 0 x87Sw_S 0 x87Sw_U 0 x87Sw_S 0 x87Sw_S 0 x87Sw_S 0 x87Sw_S 0 x87Sw_S 0 x87Sw_S 0 x87Sw_
.text:00CE30C5 malware.exe:\$530C5 #524C5		3: [esp+8] 00000000 4: [esp+C] 00000001
💭 Dump 1 💭 Dump 2 💭 Dump 3 💭 Dump 4 💭 Dump 5	∰ Watch 1 [x= Locals	0 045FF470 4 00000000
Address Hex 045FF5A0 00 00 00 00 00 00 00 00 00 00 00 00 0	ASCII 045FF3D 00	8 0000000 C 00000001 0 F000000

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:

	00CE3700 00CE3701 00CE3702	56 53 FF 35	34 49 D5 0	0	oush esi oush ebx oush dword p	tr ds:[<mark>D549</mark>	34]			x87SW_C1 0 x87SW_C0 0 x87SW_E5 0 x87SW_SF 0 x87SW_P 0 x87SW_U 0
EIP		FF 15	<u>30 D0 D2 0</u>	<u>0</u>	call dword p	tr ds:[<&Cr	yptGenRandom>]		, `	Default (stdcall)
.text:00CE	3708 malware.	exe: \$53708	#52808	j= <auvap132< th=""><th>.cryptGerkar</th><th>idom></th><th></th><th></th><th></th><th>2: [esp+4] 00000020 3: [esp+8] 0075D058 4: [esp+C] 13BA7841</th></auvap132<>	.cryptGerkar	idom>				2: [esp+4] 00000020 3: [esp+8] 0075D058 4: [esp+C] 13BA7841
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	💮 Watch 1	[x=] Locals	2 Struct	045FF440 045FF444	00750	DE70 0020
Address He	AV				ASCTT	1		045FF448	00750	D058

Figure 49

Address	He	ĸ															ASCII
0075D058	3C	EO	06	45	0E	94	36	54	B5	B9	F8	C4	8E	5D	5C	C4	<`a.E6Tµ`øÄ.]\Ä
0075D068	69	11	83	89	D8	C4	BO	FF	7C	CO	F5	C3	B9	DD	ED	8A	iØÄ°ÿ ÀÕÅ'Ŷí.

Figure 50

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

<pre></pre>	push 0 push 0 push 0 lea eax,dword ptr ss:[esp+8F4] push 0 push 0 lea eax,dword ptr ss:[esp+100] push eax call dword ptr ds:[e&CryptStringToBinaryA>] crypt32.CryptStringToBinaryA>	eax eax eax >	x87TW_4 3 (Empty) x87TW_5 3 (Empty) x87TW_6 3 (Empty) x87TW_7 3 (Empty) x87Stw_6 3 (Empty) x87TW_7 3 (Empty) x87Stw_6 0 x87SW_C3 0 x87SW_C2 0 x87SW_5 0 x87SW_C3 0 x87SW_E5 0 x87SW_5 0 x87SW_P 0 x87SW_U 0 Default(stdcall) 1 [esp1 045FE4C4 "BEGIN PUBLIC KEY\ 2: [esp+4] 00000000 4: [esp+c] 045FECC4
🕮 Dump 1 👹 Dump 2 👹 Dump 3 👹 Dump 4 👹 Dump 5	👹 Watch 1 🛛 🗐 Locals 🖉 Struct	045FE3C0 045F 045FE3C4 0000	E4C4 "BEGIN PUBLIC KEY\nMIIBIJANBgkq
Address Hex	ASCII	045FE3C8 0000	0000
045FE4C4 2D 2D 2D 2D 2D 2D 42 45 47 49 4E 20 50 55 42 4C	49BEGIN PUBLI	045FE3D0 045F	E484
045FE4E4 6A 41 4E 42 67 6B 71 68 6B 69 47 39 77 30 42	1 jANBgkqhkiG9w0BA	045FE3D4 0000	0000
Concentration as we asilan as an anion on an apilan an ap		04572508 0000	5000

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**):

	push dax lea eax,dword ptr ss:[esp+A0] push dax push dword ptr ss:[esp+88] lea eax,dword ptr ss:[esp+8FC] push dax push dax push dax push eax push dword ptr ds:[<4CryptDecodeObjectEx>] > >push dword ptr ds:[<4CryptDecodeObjectEx>]	x87Tw_2 3 (Empty) x87Tw_3 3 (Empty) x87Tw_4 3 (Empty) x87Tw_5 3 (Empty) x87Tw_6 3 (Empty) x87Tw_5 3 (Empty) x87StatusWord 0000 x87Sw_5 0 x87Sw_6 2 3 (Empty) x87Ssw_5 0 x87Sw_6 0 x87Sw_6 2 0 x87Sw_5 0 x87Sw_5 0 x87Sw_6
Ump 1 Ump 2 Ump 3 Ump 4 Ump 5	Watch 1 x= Locals Struct	00000001
Address Hex 045FECC4 303 82 01 22 30 00 06 09 2A 86 45 86 F7 00 01 01 045FECC4 303 82 01 02 00 03 82 01 01 02 02 01 01 02 02 01 03 02 02 01 03 02 02 01 03 02 01 03 02 02 01 03 02 02 01 03 02 02 01 03 02 02 01 03 02 02 01 03 02 02 01 03 02 02 01 03 02 <td>ASCII 045FE3C4 00.1 045FE3C8 00.1 045FE3C8 045FE3C9 045FE3C8 045FE3C9 045FE3C9 045FE3C9 045FE3C9 045FE3C9 045FE3C9 045FE3C9 045FE3C9 045FE3C9 045FE3C9</td> <td>145FEC24 00000126 0000000 0000000 045FE478 045FE478 045FE48C</td>	ASCII 045FE3C4 00.1 045FE3C8 00.1 045FE3C8 045FE3C9 045FE3C8 045FE3C9 045FE3C9 045FE3C9 045FE3C9 045FE3C9 045FE3C9 045FE3C9 045FE3C9 045FE3C9 045FE3C9	145FEC24 00000126 0000000 0000000 045FE478 045FE478 045FE48C



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

	push dword ptr ss:[esp+A0] push dword ptr ss:[esp+A0] push dword ptr ss:[esp+A0] call dword ptr ds:[csCryptImportPublicKeyInfo>] = <crypt32.cryptimportpublickeyinfo></crypt32.cryptimportpublickeyinfo>][es >	x875W_E0 0 x875W_C2 0 x875W_C10 x875W_C2 0 x875W_E5 0 x875W_F0 x875W_P 0 x875W_U 0 Default (stdcall) ▼ 5 □ Unlock 1: [esp1 0074E720 <&CPAcquireContext> 2: esp+400000001 0 2: esp+400000001 04554846 a*1.2.840.113549.1.1.1" 4:
Ump 1 Ump 2 Ump 3 Ump 4 Ump Dump 4	5 🛞 Watch 1 🛛 🕸 Locals 🖉 Struct	045FE3CC 0074 045FE3D0 0000	4E720 00001
Address Hex	ASCII	045FE3D4 0074	4E548 &"1.2.840.113549.1.1.1"
0074E548 ISOI E5 74 00 02 00 00 02 85 74 00 02 00 00 01 12 32 26 38 34 30 0074E558 80 E5 74 00 00 00 00 31 12 32 26 38 34 30 0074E558 31 31 33 35 34 39 22 31 22 38 34 30 0074E558 31 31 33 35 34 39 22 31 22 12 21 10 10 10 40 10 <	00 a tx a t 2E . a t 8A 113549.1.1.1. 0 .° 01°. 0 .°0	045FE3DC 045F 045FE3E0 0075 045FE3E4 045F 045FE3E8 0075	F7F8 &L"\\\\?\\Z:\\test.txt" 5D238 FF6D8

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	He	x															ASCII
0074EB50	08	31	AD	2B	4A	37	51	DA	02	76	4D	83	2C	74	C6	36	.1.+J7QU.VM., tA6
0074EB60	9D	D0	F2	BF	F8	13	64	79	9E	79	C9	A2	5D	2B	CD	AB	.Dò¿ø.dy.yÉ¢]+Í«
0074EB70	D7	59	F3	15	BE	AF	B1	C3	B 3	6A	F5	3B	CD	B 7	69	B 5	×Yó.☆ ±A•jõ;İ·iμ
0074EB80	9C	A1	5 F	EE	4C	C3	8B	CE	3E	D8	7B	38	48	43	C6	OF	.i_îLĂ.Î>Ø{8HCÆ.
0074EB90	03	51	08	BF	EC	8E	72	DE	02	48	BB	4C	11	C4	BE	17	.Q.21.rb.H»L.A%.
0074EBA0	08	CD	98	27	12	90	11	FE	82	22	14	54	DE	F2	EC	6F	.1.'þ.".TÞòìo
0074EBB0	D3	8C	A2	C1	01	91	C1	F2	A9	10	2E	6D	87	7C	47	32	0.¢AAo@m. G2
0074EBC0	17	23	DB	35	C5	2A	D8	54	15	2D	D7	34	62	B6	20	56	.#05Å*ØTx4b¶ V
0074EBD0	3D	DO	50	39	66	8A	1E	C9	21	04	48	BD	0A	7F	9D	CB	=DP9fE!.H%E
0074EBE0	D5	FB	EC	C3	77	29	92	25	2D	03	60	13	1E	DE	33	8B	Ô <u>û</u> ìĂw).%`Þ3.
0074EBF0	72	AF	4F	74	48	5A	32	B 3	51	23	1C	4C	4B	66	87	A5	r OtHZ2 Q# LKf ¥
0074EC00	5E	07	FD	FD	BC	35	FA	56	2C	11	29	90	E3	08	5D	81	^.ýý¼5ú∨,.).ā.].
0074EC10	57	37	5D	F8	AB	97	70	4F	A5	98	BB	A7	FD	D9	57	EE	W7]Ø«.pO¥.»§ýÙWî
0074EC20	01	BC	30	7F	01	F8	70	D5	EC	69	CE	6E	EA	90	7A	8F	.%0opOiiînê.z.
0074EC30	32	CF	FF	34	93	4A	E4	63	AF	E8	1A	D9	A2	3A	AD	A3	2Iÿ4.Jäc e.Ù¢∶.£
0074EC40	29	DE	B2	3B	00	34	2D	2A	5 B	A2	3F	EB	01	86	2E	50)Þ⁼;.4-*[¢?ëP

Figure 55

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





1		_																
	Address	He	ĸ															ASCII
	007500C0	55	43	36	47	41	65	73	2F	6F	6C	73	71	4C	54	51	41	UC6GAes/olsqLTQA
	007500D0	4F	37	4C	65	4B	61	4F	74	4F	71	4C	5A	47	75	69	76	07LeKaOtOqLZGuiv
	007500E0	59	2B	52	4B	6B	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	7lfZ/ae7mKVPcJer
	00750110	2B	46	30	33	56	34	46	64	43	4F	4F	63	4B	52	45	73	+F03V4FdC00cKREs
ł	00750120	56	76	6F	31	76	50	33	39	42	31	36	6C	68	32	5A	4C	Vvo1vP39B161h2ZL
1	00750130	54	42	77	6A	55	62	4D	79	57	6B	68	30	54	36	39	79	TBwjUbMyWkh0T69y
	00750140	69	7A	50	65	48	68	4E	67	41	79	30	6C	6B	69	6C	33	izPeHhNgAy01ki13
ł	00750150	77	2B	7A	37	31	63	75	64	66	77	71	39	53	41	51	68	w+z71cudfwq9SAQh
	00750160	79	52	36	4B	5A	6A	6C	51	30	44	31	57	49	4C	5A	69	yR6KZjlQ0D1WILZi
	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/s8t5UFCKC
	007501A0	2F	68	47	51	45	69	65	59	7A	51	67	58	76	73	51	52	/hGQEieYzQgXvsQR
	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	D8ZDSDh72D70i8NM
	007501D0	37	6C	2B	68	6E	4C	56	70	74	38	30	37	39	57	71	7A	71+hnLVpt8079Wqz
	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 SetEntriesInAclA routine:

	<pre>push eax push eax movlpd qword ptr ss:[esp+30] movlpd qword ptr ss:[esp+30],xmm0 push ax push 1 mov dword ptr ss:[esp+30],FFFFFFFF mov dword ptr ss:[esp+30],esi mov dword ptr ss:[esp+40],esi mov dword ptr ss:[esp+40],esi mov dword ptr ss:[esp+40],si mov dword ptr ss:[esp+</pre>	x87Tw_0 3 (Empty) x87Tw_1 3 (Empty) x87Tw_2 3 (Empty) x87Tw_3 3 (Empty) x87Tw_4 3 (Empty) x87Tw_5 3 (Empty) x87Tw_6 3 (Empty) x87Tw_5 3 (Empty) x87Tw_6 3 (Empty) x87Tw_7 3 (Empty) x87Sw_5 0 x87Sw_C 0 x87Sw_C 0 x87Sw_5 0 x87Sw_5 0 x87Sw_5 0 x87Sw_C 0 x87Sw_E 0 x87Sw_5 0 x87Sw_2 0 x87Sw_5 0 x87Sw_P 0 x87Sw_U 0 0 Default (stdcal) ▼ 5 € Unlock 2: [esp+4] 045FF4AC 3: [esp+6] 045FF490
🚛 Dump 1 🚛 Dump 2 💭 Dump 3 🚛 Dump 4 🚛 Dump	5 🛞 Watch 1 [X=] Locals 🖉 Struct 045FF47	00000001 8 045FF4AC
Address Hex	ASCII 045FF470 045FF48	00000000

Figure 59

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





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

OCA3375 OCA3377 OCA3377 OCA3377 OCA3372 OCA3372 OCA3382 OCA3382 OCA3882 OCA3882 OCA3882 OCA3882 OCA388 OCA388 OCA388 OCA388 OCA388 OCA388 OCA388 OCA388 OCA388 OCA388	6A 00 68 80 00 00 00 6A 03 6A 00 6A 00 68 00 00 00 C0 50 FF 15 8C 64 D5 00 FF 15 8C 64 D5 00 E.&CreateF11ew>]= <kernel32 \$13388 #12788</kernel32 	push 0 push 80 push 30 push 0 push CO000000 push cax call dword ptr ds:[<&CreateFilew>] .createFilew>	eas	x875tatusword 0000 x875w_B 0 x875w_ x875w_C1 0 x875w_ x875w_SF 0 x875w_ x875w_SF 0 x875w_ z875w_SF 0 x875w_ z875w_z875w_ z875w_ z875w_ z875w_ z875w_z75w_ z875w_ z875w_ z875w_z75w_ z875w_	x875W_C2 0 0 0 x875W_C2 0 0 0 x875W_E5 0 0 x875W_U 0 ▼ [5] □ Unlock "\\\\?\\z:\\test.txt"
💭 Dump 1 💭 Dump 2 💭 I	Dump 3 💭 Dump 4 💭 Dum	o 5 🛞 Watch 1 🛛 🕸 Struct	045FF4B8 0 045FF4BC C	074CB88 L"\\\\?\\Z:\\t 0000000	est.txt"
Address Hex 0074EC78 EE FE EE FE EE FE 0074EC88 EE FE EE FE EE FE 0074EC98 EE FE EE FE EE FE 0074EC98 EF FE EE FE EE FE	EE FE EE FE EE FE EE FE EE EE FE EE FE EE FE EE FE EE EE FE EE FE EE FE EE FE EE	ASCII FE [pipipipipipipipi FE [pipipipipipipipi FE [pipipipipipipipipi FE [pipipipipipipipipi	045FF4C0 0 045FF4C4 0 045FF4C8 0 045FF4CC 0 045FF4CC 0 045FF4D0 0	0000000 0000000 0000003 0000080 0000080	

Figure 61

The file size is retrieved using the GetFileSizeEx API:

EIP	00CA35BC 00CA35BC 00CA35BC 00CA35BC 00CA35BC 00CA35CS 4	50 56 C7 45 FF 15	E8 00 00 0 CC D0 D2 0		oush eax oush esi ov dword ptr all dword pt	ss:[ebp-1 r ds:[<mark><&Ge</mark>	s,0 tFileSizeEx:	•]	> De	875W_C1 0 x875W_C0 875W_SF 0 x875W_P	0 x875W_ES 0 x875W_U • 5	Unlock
dword ptr	[OOD2DOCC <mail< td=""><td>lware.&GetF exe:\$135C5</td><td>ileSizeEx>] #129C5</td><td>=<kernel32.0< td=""><td>GetFileSizeE</td><td>x></td><td></td><td></td><td>1 2 3 4</td><td>: [esp] 00000260 : [esp+4] 045F688 : [esp+8] 13BA7A55 : [esp+C] 007389B0</td><td></td><td></td></kernel32.0<></td></mail<>	lware.&GetF exe:\$135C5	ileSizeEx>] #129C5	= <kernel32.0< td=""><td>GetFileSizeE</td><td>x></td><td></td><td></td><td>1 2 3 4</td><td>: [esp] 00000260 : [esp+4] 045F688 : [esp+8] 13BA7A55 : [esp+C] 007389B0</td><td></td><td></td></kernel32.0<>	GetFileSizeE	x>			1 2 3 4	: [esp] 00000260 : [esp+4] 045F688 : [esp+8] 13BA7A55 : [esp+C] 007389B0		
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	🛞 Watch 1	[x=] Locals	2 Struct	045FF4CC 00 045FF4D0 04	0002 5FF6	60 88		

Figure 62

AvosLocker reads 1MB at a time:

00CA39C9 6A 00 00CA39C5 8D 85 94 FE FF FF 00CA3902 50 A0 0F 00 00CA3902 68 00 A0 0F 00 00CA3905 57 00CA3905 5F 15 84 64 05 00	Dush 0 lea eax, dword ptr ss:[ebp-16C] Dush eax Dush FA000 Dush edi Dush depend ptr ds:[cs0ead51]eb]	x87Statusword 0000 x87Sw_B 0 x87Sw_C3 0 x87Sw_C2 0 x87Sw_LC 0 x87Sw_E5 0 x87Sw_LC 0 x87Sw_E5 0 x87Sw_S5 0 x87Sw_P 0 x87Sw_U 0
	>	Default (stdcall) 🔻 5 🗘 🗌 Unlock
<pre>dword ptr [00056484 <malware.&readfile>]=<kernel32.readf .text:00CA3909 malware.exe:\$13909 #12DD9</kernel32.readf </malware.&readfile></pre>	ile>	1: [esp] 0000260 2: [esp+4] 04D0E020 3: [esp+8] 000FA000 4: [esp+C] 045FF564
Ump 1 Ump 2 Ump 3 Ump 4 Ump 5	Watch 1 Ix=l Locals Struct 045FF4C0 0000 045FF4C1 045F 045F 045F 045F 045F	0260
Address Hex 04D0E020 00 00 00 00 00 00 00 00 00 00 00 00	ASCII 045FF4C8 000F 045FF4CC 045F	A000 F564

Figure 63

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

	00CA39D 00CA39E 00CA39E	F 6A 0 1 6A 0 3 6A 0 5 56 6 FF 1	5 <u>C4 64 D5 (</u>	10	oush 0 oush 0 oush 0 oush esi call dword pr	tr ds:[<&se	tFilePointer>]]		*	x875W_B 0 x87 x875W_C1 0 x87 x875W_SF 0 x87 Default (stdcall) 1: [esp] 000002	SW_C3 'SW_C0 'SW_P	0 x875w_C2 0 x875w_E5 0 x875w_U	5 🗘 🗌 Unlock
text:00CA3	ODSO4C4 Cild	eve: \$13956	#12056	-skernersz	. setriferoni	(er)				٦	2: [esp+4] 0000 3: [esp+8] 0000 4: [esp+C] 0000	0000 0000 0000		
. centrouchs	seo marmare.	CAC. 919900	FILDED							_				
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	💮 Watch 1	[x=] Locals	2 Struct		045FF4C4 0 045FF4C8 0	0000	0260			
Address He	ex	41 41 41	41 41 41 41	41 41 41 41	ASCII			^	045FF4CC 0 045FF4D0 0	0000	0000			

Figure 64

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

EIP	00CDE1A 00CDE1A 00CDE1A 00CDE1A 00CDE1A 00CDE1A 00CDE1A 00CDE1A	8 57 9 53 8 56 8 51 0 88 C/	0 1C		push edi push ebx push esi push ecx mov ecx,edx call dword pi	tr ds:[eax+	10]		× [x8/Statusword 0000 x875W_B 0 x875W_C3 x875W_C1 0 x875W_C0 x875W_F 0 x875W_C x855W_F 0 x875W_P Default (stdcall)	0 x875W_C2 0 0 x875W_E5 0 0 x875W_U 0
dword ptr [[eax+1C]=[ma]	lware.00D3D	#4D5AE	.00CDF150						1: [esp] 00000003 2: [esp+4] 04D0E020 3: [esp+8] 04D0E020 4: [esp+C] 00000040	
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	🛞 Watch 1	[x=] Locals	2 Struct	045FF498 0 045FF49C 0	0000 4D0E	003	
Address He 04D0E020 43	ex 1 41 41 41 43	1 41 41 41	41 41 41 41	41 41 41 4	ASCII 1 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA	AAAAA		▲ 045FF4A0 0 045FF4A4 0	4D0E	020 040	





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

.text:00CDF672	loc_CDF6	572:	
.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	paddd	xmm0,	xmm2
.text:00CDF694	paddd	xmm4,	xmm6
.text:00CDF698	movdqa	xmm1,	Ommx
.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+3	<pre>SF0h+var_38C], xmm0</pre>
.text:00CDF6CE	movdqa	[esp+3	<pre>3F0h+var_37C], xmm4</pre>
.text:00CDF6D4	movdqa	xmm1,	xmmØ
.text:00CDF6D8	movdqa	xmm5,	xmm4
.text:00CDF6DC	paddd	xmm0,	xmm2
.text:00CDF6E0	paddd	xmm4,	xmm6
.text:00CDF6E4	movdqa	xmm3,	×mmØ
.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+3	<pre>SF0h+var_34C], xmm0</pre>
.text:00CDF723	movdqa	[esp+3	<pre>SF0h+var_33C], xmm4</pre>
.text:00CDF72C	movdqa	xmm3,	xmmØ
.text:00CDF730	movdqa	xmm7,	xmm4
.text:00CDF734	paddd	xmm0,	xmm1
.text:00CDF738	paddd	xmm4,	xmm5
.text:00CDF73C	movdqa	xmm1,	xmmØ
.text:00CDF740	movdqa	xmm5,	xmm4
.text:00CDF744	pslld	xmm0,	ØDh
.text:00CDF749	pslld	xmm4,	ØDh
.text:00CDF74E	psrld	xmm1.	13h

Figure 66





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:

EIP	● 00C9D04 ● 00C9D04 ● 00C9D04 ● 00C9D04	6 51 7 50 8 FF 15	<u>C0 64 D5 0</u>		ush ecx ush eax all dword pt	tr ds:[<mark><&Mo</mark>	veFilew>]	ecx	~ D	x87SW_SF 0 x87SW_P	0 x875W_U	0 Unlock
dword ptr	000564C0 <ma< td=""><td>lware.&Move</td><td>eFileW>]=<ke #C448</ke </td><td>rnel32.MoveF</td><td>ilew></td><td></td><td></td><td></td><td>234</td><td><pre>c: [esp+4] 0074CBS8 L c: [esp+4] 0075D010 c: [esp+8] 13BA7B25 c: [esp+C] 007389B0</pre></td><td>L"\\\\?\\Z:\\te</td><td>st.txt.avos</td></ma<>	lware.&Move	eFileW>]= <ke #C448</ke 	rnel32.MoveF	ilew>				234	<pre>c: [esp+4] 0074CBS8 L c: [esp+4] 0075D010 c: [esp+8] 13BA7B25 c: [esp+C] 007389B0</pre>	L"\\\\?\\Z:\\te	st.txt.avos
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	🛞 Watch 1	[x=] Locals	2 Struct	045FF6D8 00 045FF6DC 00	074CE 075D0	B88 L"\\\\?\\Z:\\te D10 L"\\\\?\\Z:\\te	st.txt" st.txt.avos2"	

Figure 69

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

🔝 test.txt.avos	2																
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	OD	0E	OF	
00000F70	EF	71	48	31	9F	43	3A	1A	B7	37	49	DC	55	BB	53	35	igHlŸC:. 7IÜU»S5
00000F80	Bl	B9	3D	35	B8	73	46	Al	96	F9	0B	Fl	B9	E6	20	2A	±1=5,sF;-ù.ñ1æ *
00000F90	91	C2	93	94	Al	E4	4F	0C	53	FC	1E	2B	12	2F	EF	2C	'Â"";äO.Sü.+./ï,
00000FA0	7A	F4	FB	8D	B7	AO	2A	BA	69	EO	A2	3F	EB	95	15	28	zôû. · *°ià¢?ë•.(
00000FB0	4C	36	4D	EB	D2	64	38	E9	12	ED	FF	FO	A6	31	20	05	L6MëÒd8é.íÿð¦l .
00000FC0	99	Fl	2C	5B	1C	4E	29	99	82	15	42	B3	E6	F8	11	65	™ñ,[.N)™,.B'æø.e
00000FD0	35	27	64	C2	2F	CC	OF	27	36	19	35	C4	DC	28	9A	2F	5'dÂ/Ì.'6.5ÄÜ(š/
00000FE0	FF	ЗD	ЗD	62	E0	D5	E2	38	E8	36	24	26	A5	85	FD	B0	ÿ==bàÕâ8è6\$&¥…ý°
00000FF0	0B	15	24	BD	C8	DF	DE	A0	8E	AD	7C	D4	E9	1E	B5	18	\$¾ÈβϷ Ž. Ôé.μ.
00001000	55	43	36	47	41	65	73	2F	6F	6C	73	71	4C	54	51	41	UC6GAes/olsqLTQA
00001010	4F	37	4C	65	4B	61	4F	74	4F	71	4C	5A	47	75	69	76	07LeKaOtOqLZGuiv
00001020	59	2B	52	4B	6B	7A	54	2F	7A	7A	4B	50	65	70	7A	71	Y+RKkzT/zzKPepzq
00001030	62	73	35	70	37	4E	56	77	2B	41	46	2F	4D	4C	77	42	bs5p7NVw+AF/MLwB
00001040	37	6C	66	5A	2F	61	65	37	6D	4B	56	50	63	4A	65	72	71fZ/ae7mKVPcJer
00001050	2B	46	30	33	56	34	46	64	43	4F	4F	63	4B	52	45	73	+F03V4FdCOOcKREs
00001060	56	76	6F	31	76	50	33	39	42	31	36	6C	68	32	5A	4C	VvolvP39B161h2ZL
00001070	54	42	77	6A	55	62	4D	79	57	6B	68	30	54	36	39	79	TBwjUbMyWkh0T69y
00001080	69	7A	50	65	48	68	4E	67	41	79	30	6C	6B	69	6C	33	izPeHhNgAy01ki13
00001090	77	2B	7A	37	31	63	75	64	66	77	71	39	53	41	51	68	w+z7lcudfwq9SAQh
000010A0	79	52	36	4B	5A	6A	6C	51	30	44	31	57	49	4C	5A	69	yR6KZjlQ0D1WILZi
000010B0	4E	4E	63	74	46	56	54	59	4B	73	55	31	32	79	4D	58	NNctFVTYKsU12yMX
000010C0	4D	6B	64	38	68	32	30	75	45	4B	6E	79	77	5A	45	42	Mkd8h20uEKnywZEB
000010D0	77	61	4B	4D	30	32	2F	73	38	74	35	55	46	43	4B	43	waKM02/s8t5UFCKC
000010E0	2F	68	47	51	45	69	65	59	7A	51	67	58	76	73	51	52	/hGQEieYzQgXvsQR
000010F0	54	4C	74	49	41	74	35	79	6A	75	79	2F	43	46	45	44	TLtIAt5yjuy/CFED
00001100	44	38	5A	44	53	44	68	37	32	44	37	4F	69	38	4E	4D	D8ZDSDh72D7018NM
00001110	37	6C	2B	68	6E	4C	56	70	74	38	30	37	39	57	71	7A	71+hnLVpt8079Wqz
00001120	77	37	47	76	76	68	58	7A	57	64	65	72	7A	53	74	64	w7GvvhXzWderzStd
00001130	6F	73	6C	35	6E	6E	6C	6B	45	2F	69	2F	38	74	43	64	osl5nnlkE/i/8tCd
00001140	4E	73	5A	30	4C	49	4E	4E	64	67	4C	61	55	54	64	4B	NsZOLINNdgLaUTdK
00001150	4B	36	30	78	43	41	3D	3D									K60xCA==

Figure 70

We continue with the analysis of the main thread.

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





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



OCCSDDC8 B8 OCCSDDC8 03 OCCSDDC4 03 OCCSDDC4 03 OCCSDDC5 05 OCCSDDD5 85 OCCSDD05 85 OCCSDD05 85 OCCSDD05 85 OCCSDD05 85	40 24 CA C2 B7 04 78 04 81 40 18 C2 D0	<pre>mov eax,dword ptr ds:[eax+24] add ecx,edx add ecx,edx movzx eax,word ptr ds:[eax+edi*2] mov eax,dword ptr ds:[ecx+eax+4] lea ecx,dword ptr ss:[ebp+18] add eax,edx push ecx call eax</pre>	eax eax eax eax eax	x87TW_6 3 (Empty) x87TW_7 3 (Empty) x875StatusWord 0000 x875W_8 0 x875W_C3 0 x875W_C2 0 x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0 Default (stdcall) ▼ 5 ↓ Unlock
eax= <kernel32.winexec> (76AB3640 .text:00C9DDDC malware.exe:\$DDDC</kernel32.winexec>)) : #D1DC			1: [esp] 004FE974 2: [esp+4] 00000000 3: [esp+6] 17AA65D9 4: [esp+C] 00F80439
Dump 1 Dump 2 Dump 3	3 💭 Dump 4 🖓 Dump 5	😨 Watch 1 🛛 🛛 🖉 Struct	004FE55C 0000	00000
Address Hex 004FE574 1/0 6F 77 65 72 73 68 60 004FE594 61 6E 64 20 22 46 12 24 61 20 04 12 24 61 20 04 12 24 61 20 04 12 24 61 20 04 12 24 61 20 04 12 24 61 20 04 12 14 62 62 62 62 62 62 62 62 62 62 62 62 62 62 63 63 74 21 04 14 62 62 63 64 74 74 22 93 14 04 63 74 74 24 04 14 15 62 74 74 14 14 15 14 15 14 14 14 15 <th>\$ 6C 6C 20 2D 43 6F 6D 6 3D 20 58 53 79 73 74 6 6 C 65 5D 3A 3A 52 65 61 6 8 5C 22 5A 3A 5C 47 45 5 9 4C 45 53 5F 42 41 43 48 41 64 64 2D 54 79 70 6</th> <th>ASCII ^ 05 @powershell -comm ^ 55 and "Sa = [Systed </th> <th>004FE560 17AA 004FE564 00F8 004FE568 0000 004FE56C 0026 004FE570 776F 004FE574 6873 004FE578 2060</th> <th>65D9 90439 90023 1000 704F 77265 6665</th>	\$ 6C 6C 20 2D 43 6F 6D 6 3D 20 58 53 79 73 74 6 6 C 65 5D 3A 3A 52 65 61 6 8 5C 22 5A 3A 5C 47 45 5 9 4C 45 53 5F 42 41 43 48 41 64 64 2D 54 79 70 6	ASCII ^ 05 @powershell -comm ^ 55 and "Sa = [Systed	004FE560 17AA 004FE564 00F8 004FE568 0000 004FE56C 0026 004FE570 776F 004FE574 6873 004FE578 2060	65D9 90439 90023 1000 704F 77265 6665

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:

Avoslocker
Attention! Your systems have been encrypted, and your confidential documents were downloaded.
In order to restore your data, you must pay for the decryption key & application. You may do so by visiting us at http://avosjon4pfh3y7ew3jdwz6ofw7lljcxlbk7hcxxmnxlh5kvf2akcqjad.onion.
This is an onion address that you may access using Tom Browser which you may download at https://www.torproject.org/download/ Details such as pricing, how long before the price increases and such will be available to you once you enter your ID presented to you below in this note in our website. Contact us soon, because those who don't have their data leaked in our press release blog and the price thev'll have to pay will go up significantly.
The corporations whom don't pay or fail to respond in a swift manner have their data leaked in our blog, accessible at http://avosqxh72b5ia23d15fgwcpndkctuzqvh2iefk5imp3pi5gfhe15klad.onion
Your ID: 6fe8a214a13dc49ce542b47e1d63135f48e40ccf6e0cd20fdbeafdc720a717cb





Running with the -h (--help) parameter

AvosLocker displays the help menu:





Running with the -p (--path) parameter

The ransomware only encrypts this specific directory:

Administrator: Administrator Command Prompt – C × 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_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:

COCA76EB FF 15 64 D0 D2 00	call dword ptr ds:[<&GetConsoleWindow>]	> D	efault (stdcall)	▼ 5 🗘 🗌 Unlocke
dword ptr [0002D064 <malware.&getconsolewindow>]=<ker .text:00CA76EB malware.exe:\$176EB #16AEB</ker </malware.&getconsolewindow>	nel32.GetConsolewindow>	2334	2: [esp+4] 656F49C2 2: [esp+8] 03336D88 &" 4: [esp+C] 03339610 &"	'ALLUSERSPROFILE=C:\\Progr 'C:\\Users\\\\Desktop\\

Figure 77

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



EIP	00CA76E9 00CA76EB 00CA76F1 00CA76F2 00CA76F2	6A 00 FF 15 50 FF 15	64 D0 D2 00		all dword pt ush eax all dword pt	tr ds:[<mark><&Ge</mark> tr d <u>s</u> :[<&St	etConsoleWind	low>]	~	x87 x87 Defa	SW_C1 0 SW_SF 0	x875W_C x875W_P	0 0 0	x87SW_ES x87SW_U ▼	0 0 5 🗣 🗌 Unloc
dword ptr [00D2D254 <ma]ware.&showwindow>]=<user32.showwindow> .text:00CA76F2 ma]ware.exe:\$176F2 #16AF2</user32.showwindow></ma]ware.&showwindow>							1: 2: 3: 4:	[esp] 00 [esp+4] [esp+8] [esp+C]	00000000 656F49C2 03336D88	&"ALLU	ISERSPROF	ILE=C:\\Pro			
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	🛞 Watch 1	[x=] Locals	2 Struct	010FF08	001/	0326					

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**):



COCA43A5 OCCA43A7 OCCA43A7 OCCA43A4 OCCA43A4 OCCA43A4 OCCA43A4 OCCA43A4 OCCA43A4 OCCA43A4 OCCA43A5 OCCA45 OCCA45	Image: Constraint of the second sec						
Dump 1 Dump 2	Dumo 3 📖 Dumo 4 📖 Dumo	5 🛞 Watch 1 🛛 🖉 Str	010FE554 03	346A08			
Address Hex		ASCII	010FESS8 000 010FESSC 000 010FESSC 000 010FESSC 000	000000			

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**):

00CA4992 6A 04 00CA4994 6A 00 00CA4994 6A 00 00CA4995 50 00CA4995 50 00CA4995 FF 15 30 D2 D2 00 C	push 4 push 0 push eax call dword ptr ds:[<&wNetAddConnection2A>]	~	x875W_B 0 x875W_C3 x875W_C1 0 x875W_C0 x875W_SF 0 x875W_P Default (stdcall)	0 x875W_C2 0 0 x875W_E5 0 0 x875W_U 0 • 5 0 Unlock
dword ptr [00D2D230 <malware.&wnetaddconnection2a>]=<m .text:00CA4999 malware.exe:\$14999 #13D99</m </malware.&wnetaddconnection2a>	pr.WNetAddConnection2A>		2: [esp+4] 0000000 3: [esp+8] 0000000 4: [esp+C] 00000004	
Ump 1 Ump 2 Ump 3 Ump 4 Ump Dump 4	5 👹 Watch 1 🛛 🕸 Locals 🖉 Struct	00E5E720 0311 00E5E724 0000	54E8 0000	
Address Hex	ASCII	O0E5E728 0000 00E5E72C 0000	0000	
00E5E8E0 4F 5C 41 24 00 00 00 00 00 00 00 00 00 00 00 00	00 0\A\$	00E5E730 0B2E	02FE	



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:

E Administr	rator: C:\Users\	top\malware.exe								-	×
The oper	ation complet	ed success	fully.								^
REMOTEDI	SK! Localname	NULL Remo	teName	\\DESKTOP-	∖share P	rovider Mic	rosoft Winde	ows Networ	k Comment		
SCAN											
Bruting	\\DESKTOP-	\A\$. []								
Bruting	\\DESKTOP-	\B\$. []								
Bruting	\\DESKTOP-	\C\$. [X]								
Bruting	\\DESKTOP-	\D\$. []								
Bruting	\\DESKTOP-	\E\$. []								
Bruting	\\DESKTOP-	\F\$. []								
Bruting	\\DESKTOP-	\G\$. []								
Bruting	\\DESKTOP-	\H\$. []								
Bruting	\\DESKTOP-	\I\$. []								
Bruting	\\DESKTOP-	\J\$. []								
Bruting	\\DESKTOP-	\κ\$. []								
Bruting	\\DESKTOP-	\L\$. []								
Bruting	\\DESKTOP-	\M\$. []								
Bruting	\\DESKTOP-	\N\$. []								
Bruting	\\DESKTOP-	\0\$. []								
Bruting	\\DESKTOP-	\P\$. []								
Bruting	\\DESKTOP-	\Q\$. []								
Bruting	\\DESKTOP-	\R\$. []								
Bruting	\\DESKTOP-	\S\$. []								
Bruting	\\DESKTOP-	\T\$. []								
Bruting	\\DESKTOP-	\U\$. []								
Bruting	\\DESKTOP-	\v\$. []								
Bruting	\\DESKTOP-	\w\$.[]								
Bruting	\\DESKTOP-	\x\$.[]								
Bruting	\\DESKTOP-	\Y\$. []								
D mouth a more	\\DECKTOD	\ 76	EV1								

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 }\"

