

# AvosLocker Ransomware Behavior Examined on Windows & Linux

By Ghanshyam More

Published: 2022-03-07 · Archived: 2026-04-06 00:47:19 UTC

*AvosLocker is a ransomware group that was identified in 2021, specifically targeting Windows machines. Now a new variant of AvosLocker malware is also targeting Linux environments. In this blog, we examine the behavior of these two AvosLocker Ransomware in detail.*

AvosLocker is a relatively new ransomware-as-a-service that was first spotted in late June 2021. The attackers use spam email campaigns as initial infection vectors for the delivery of the ransomware payload. During the encryption, process files are appended with the “.avos” extension. An updated variant appends with the extension “.avos2”. Similarly, the Linux version appends with the extension “.avoslinux”.

After every successful attack, the AvosLocker gang releases the names of their victims on the Dark Leak website hosted on the TOR network and provides exfiltrated data for sale. URL structure: `hxxp://avosxxx...xxx[.]onion`

The AvosLocker gang also advertises their latest ransomware variants on the Dark Leak website. URL structure: `hxxp://avosjonxxx...xxx[.]onion`

The gang has claimed, “The AvosLocker’s latest Windows variant is one of the fastest in the market with highly scalable threading and selective ciphers.” They offer an affiliate program that provides ransomware-as-a-service (RaaS) for potential partners in crime.

Recently they have added support for encrypting Linux systems, specifically targeting VMware ESXi virtual machines. This allows the gang to target a wider range of organizations. It also possesses the ability to kill ESXi VMs, making it particularly nasty.

According to [deepweb research](#) by Cyble Research Labs, the Threats Actors of AvosLocker ransomware groups are exploiting Microsoft Exchange Server vulnerabilities using Proxyshell, compromising the victim’s network.

CVEs involved in these exploits are CVE-2021-34473, CVE-2021-31206, CVE-2021-34523, and CVE-2021-31207.

## Technical Analysis of AvosLocker Windows Variant

### Command-Line Options

The following figure shows a sample of Command-Line Options.

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. All rights reserved.

C:\samples>sample.exe -h
Build: Sonic
SonicMango
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
-s, --unsafe        Enable system&hidden attributed files encryption
                   (BSOD?)
--hide             Hide console window
-t, --threads arg  Max threads for encryption (default: 200)
-h, --help         Print usage
    
```

Fig. 1: Command Line Option

The available options allow for control over items like enabling/disabling SMB brute force, mutex creation, or control over the concurrent number of threads.

If no options are given, the malware runs with default options as shown in figure 2, where it ignores encryption of network drives and SMB share. It runs 200 threads concurrently of its file encryption routine.

```

Build: Sonic
b_bruteforce_smb_enable: 0
b_logical_disable: 0
b_network_disable: 1
b_system_disable: 0
b_mutex_disable: 0
concurrent_threads_num_max: 200
    
```

Fig. 2: Execution with Default Parameter

While execution, the malware console displays detailed information about its progress on the screen (fig. 3).

```

Skipped SYSTEM & HIDDEN file C:\DumpStack.log.tmp
Skipped SYSTEM & HIDDEN file C:\Users\Windows10\ntuser.dat.LOG1
Skipped SYSTEM & HIDDEN file C:\Users\Windows10\ntuser.dat.LOG2
Skipped SYSTEM & HIDDEN file C:\Users\Windows10\NTUSER.DAT{7684062e-ffff-11eb-ae7-005056b0f1b2}.TM.b1f
Skipped SYSTEM & HIDDEN file C:\Users\Windows10\NTUSER.DAT{7684062e-ffff-11eb-ae7-005056b0f1b2}.TMContainer000000000000000001.regtrans-ms
Skipped SYSTEM & HIDDEN file C:\Users\Windows10\NTUSER.DAT{7684062e-ffff-11eb-ae7-005056b0f1b2}.TMContainer000000000000000002.regtrans-ms
drive D: took 0.000000 seconds
    
```

Fig. 3: Progress Details

Most of the strings in the malware are kept in the XOR encrypted format. The decryption routines are similar, only registers and keys are different (fig. 4). Strings are decrypted just before their use.

```

loc_107D130:                                     ; CODE XREF
83 F8 06      cmp     eax, 6
73 0D         jnb    short loc_107D142
30 4C 05 11   xor    [ebp+eax+11h], cl
40           inc    eax
89 45 98      mov    [ebp-68h], eax
8A 4D 10      mov    cl, [ebp+10h]
EB EE       jmp    short loc_107D130
    
```

Fig. 4: Commonly Used Decryption Routine

Initially, the malware collects the command line options provided while launching the application (fig. 5).

```

.text:00DC3F54
.text:00DC3F54      sub_DC3F54 proc near                                ; DATA XREF: .rdata:00E094B0↓o
.text:00DC3F54 FF 15 50 21 E0 00  call    ds:GetCommandLineA
.text:00DC3F5A A3 20 FC E2 00  mov     dword_E2FC20, eax
.text:00DC3F5F FF 15 54 21 E0 00  call    ds:GetCommandLineW
.text:00DC3F65 A3 24 FC E2 00  mov     dword_E2FC24, eax
.text:00DC3F6A B8 01  mov     al, 1
.text:00DC3F6C C3  retn
.text:00DC3F6C      sub_DC3F54 endp

```

Fig. 5: Get command-line Options

Then it decrypts the mutex name “Cheic0WaZie6zey” and checks whether it is already running or not to avoid multiple instances (fig. 6).

```

50      push    eax                                ; lpName
6A 01   push    1                                  ; bInitialOwner
6A 00   push    0                                  ; lpMutexAttributes
FF 15 68 20 E0 00  call    ds:CreateMutexA

```

Fig. 6: Mutex Creation

As shown in figure 7, AvosLocker uses multi-threaded tactics. It calls the below APIs to create multiple instances of worker threads into memory and share file paths among multiple threads. Smartly utilizing the computing power of multi-core CPUs.

APIs called:

- CreateIoCompletionPort()
- PostQueuedCompletionStatus()
- GetQueuedCompletionPort()

```

6A 00   push    0                                ; NumberOfConcurrentThreads
6A 00   push    0                                ; CompletionKey
6A 00   push    0                                ; ExistingCompletionPort
6A FF   push    0FFFFFFFh                       ; FileHandle
FF 15 34 20 85 00  call    ds:CreateIoCompletionPort

```

Fig. 7: Use of CreateIoCompletionPort

The code creates multiple threads in a loop (fig. 8). The threads are set to the highest priority for encrypting data quickly.

```

.text:00D66540
.text:00D66540      Loop:                                          ; CODE XREF: sub_D64775+1E31↓j
.text:00D66540 8D 85 88 FD FF FF  lea     eax, [ebp-278h]
.text:00D66546 50      push   eax
.text:00D66547 68 10 CF D5 00  push   offset loc_D5CF10
.text:00D6654C 8D 8D 68 F7 FF FF  lea     ecx, [ebp-898h]
.text:00D66552 E8 49 A9 00 00  call    CreateThreadCodeInside
.text:00D66557 C6 45 FC AA  mov     byte ptr [ebp-4], 0AAh
.text:00D6655B 50      push   eax
.text:00D6655C 8D 8D D8 FE FF FF  lea     ecx, [ebp-128h]
.text:00D66562 E8 69 30 00 00  call    sub_D695D0
.text:00D66567 C6 45 FC A9  mov     byte ptr [ebp-4], 0A9h
.text:00D6656B 8D 8D 68 F7 FF FF  lea     ecx, [ebp-898h]
.text:00D66571 E8 5A E8 FE FF  call    sub_D54D00
.text:00D66576 6A 02  push   2                                ; nPriority
.text:00D66578 FF B5 88 FD FF FF  push   dword ptr [ebp-278h]
.text:00D6657E 8D 8D D8 FE FF FF  lea     ecx, [ebp-128h]
.text:00D66584 E8 77 2F 00 00  call    sub_D69500
.text:00D66589 8B C8  mov     ecx, eax
.text:00D6658B E8 A0 E8 FE FF  call    sub_D54E30
.text:00D66590 50      push   eax                                ; hThread
.text:00D66591 FF 15 70 20 E0 00  call    ds:SetThreadPriority
.text:00D66597 8B 85 88 FD FF FF  mov     eax, [ebp-278h]
.text:00D6659D 40      inc    eax
.text:00D6659E 89 85 88 FD FF FF  mov     [ebp-278h], eax
.text:00D665A4 3B C7  cmp     eax, edi
.text:00D665A6 7C 98  jl     short Loop

```

Fig. 8: Create Thread In-Loop and Set Priority

AvosLocker ransomware performs a recursive sweep through the file system (fig. 9), searches for attached drives, and enumerates network resources using API WNetOpenEnum() and WnetEnumResource().

```
.text:013625CD 50          push     eax
.text:013625CE 51          push     ecx
.text:013625CF 6A 00       push     0
.text:013625D1 6A 00       push     0
.text:013625D3 6A 02       push     2
.text:013625D5 C7 85 B4 FE FF FF 30 75+mov     [ebp+dwBytes], 7530h
.text:013625D5 00 00
.text:013625DF C7 85 E8 FE FF FF FF FF+mov     [ebp+var_118], 0FFFFFFFh
.text:013625DF FF FF
.text:013625E9 FF 15 58 07 43 01      call    mpr_WNetOpenEnumA
```

Fig. 9: Search Network Share

Before selecting the file for encryption, it checks for file attributes and skips it if “FILE\_ATTRIBUTE\_HIDDEN” or “FILE\_ATTRIBUTE\_SYSTEM” as shown in figure 10.

```
.text:0107CFE8 50          push     eax
.text:0107CFE9 FF 15 64 20 12 01      call    ds:GetFileAttributesW
.text:0107CFEF A8 04       test     al, 4
.text:0107CFF1 0F 84 A2 00 00 00     jz      loc_107D099
.text:0107CFF7 A8 02       test     al, 2
.text:0107CFF9 0F 84 9A 00 00 00     jz      loc_107D099
```

Fig. 10: Check File Attribute

Once the file attribute check is passed, it performs the file extension check. It skips files from encryption if its extension gets matched with one of the extensions shown in figure 11.

```
.....avos·avoslinux
avos2·avos2j·themepack·nls·diagpkg·msi·lnk·exe·cab·scr·bat·drv·r
tp·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·cc
m·deskthemepack·shs·ldf·theme·mpa·nomedia·spl·cpl·adv·icl·msu...
```

Fig. 11: Skip Extension List

It also contains the list of files and folders that need to be skipped from the encryption (fig. 12).

```
...../Program·Fil
es·(x86).....GET_YOUR_FILES_BACK.txt...%System·Volume·Informati
on..;\GET_YOUR_FILES_BACK.txt...2·d.áö..fkrits...sTlitscanar)dsst
fautorun.inf...oProgram·Files..wbootsect.bak...!Windows.old....
.ProgramData...!desktop.ini...eMicrosoft..-config.msi..Microso
ft..3All·Users...Windows....4AppData...Lbootmgr....SSophos..Pub
lic.;Intel...ntldr..fGames..1WinNT...+è...=boot.....
.....+thumbs.db..iliteration·error·over·at·%s.nntu
ser.dat.log.&iconcache.db...bootfont.bin...Vntuser.dat.qntuser.
ini.vThumbs.db..vboot.ini...j\.....b.....)-4.d.+.ò!0+
```

Fig. 12: Skip File Folder List

AvosLocker uses RSA encryption, and it comes with a fixed hardcoded ID and RSA Public Key of the attacker (fig. 13).



```

00001480 2D 0B 09 FA FA 6B 2D 52 E1 C6 3F E9 A8 5D 0A 89 -.úúk-RáE?é`]%.%
00001490 CA B1 A1 87 9D 57 63 EB 4F 13 CF E7 1A E1 46 64 Ê±;#.WcëO.İç.áFd
000014A0 FD 13 B4 14 6F E2 39 D3 D2 B1 DE 85 E8 30 65 52 ý.'.oâ9ÓÔ±E...è0eR
000014B0 5D E2 13 30 B6 97 34 B0 32 33 BE 4E B0 FE CA 35 ]â.0q-4°23*N°pÊ5
000014C0 A8 6F 32 B3 9D EC 8D E2 A6 7D A1 C6 73 C9 58 47 ``B2'.i.â;};EsÉXG
000014D0 6F 45 34 66 2B 52 4D 70 32 53 76 56 67 79 6D 4F oE4f+RMp2SvVgymO
000014E0 31 74 78 54 78 35 61 64 70 46 46 68 55 43 55 65 ltxTx5adpFFhUCUe
000014F0 4F 75 48 42 31 2F 6F 53 58 7A 35 35 70 52 5A 2B OuHB1/oSXz55pRZ+
00001500 6B 62 5A 33 78 6C 7A 4A 41 64 71 63 33 2F 4C 78 kbZ3xlzJAdqc3/Lx
00001510 5A 48 55 55 74 70 37 41 79 77 45 68 6E 6C 31 77 ZHUUtp7AywEhnllw
00001520 4A 65 4B 52 45 34 72 64 44 68 4F 78 6F 41 58 63 JeKRE4rdDhOxoAXc
00001530 68 45 36 4A 78 37 33 50 59 75 77 37 2B 4D 70 55 hE6Jx73PYuw7+MpU
00001540 79 54 68 67 78 61 52 38 68 56 48 2F 48 47 57 43 yThgxaR8hVH/HGWC
00001550 5A 42 6E 34 64 30 36 70 2F 58 4C 41 31 6A 57 4D ZBn4d06p/XLAljWM
00001560 33 74 73 53 49 46 78 64 38 4A 4B 74 33 63 7A 5A 3tsSIFxd8JKt3czZ
00001570 72 48 65 32 78 66 76 4E 78 34 39 64 6C 55 66 44 rHe2xfvNx49dlUfD
00001580 39 63 49 72 6C 2B 45 76 6F 56 74 79 6A 75 79 72 9cIrl+EvoVtyjuyr
00001590 65 49 34 66 4B 30 50 56 41 4D 6D 32 69 64 6C 74 eI4fK0PVAMm2idlt
000015A0 45 62 5A 33 33 44 2F 65 71 2F 78 36 77 74 2F 41 EbZ33D/eq/x6wt/A
000015B0 58 57 48 6F 6B 62 57 65 31 78 73 58 72 75 4C 54 XWHokbWelxsXruLT
000015C0 6E 68 62 62 74 2B 44 5A 73 6B 70 62 58 37 36 58 nhbbt+DZskpbX76X
000015D0 71 53 66 55 59 42 71 52 45 65 6A 76 38 31 7A 31 qSfUYBqReEjv81zl
000015E0 6F 33 75 33 62 61 6F 5A 7A 7A 33 66 72 77 52 50 o3u3baoZzz3frwRP
000015F0 75 5A 30 36 54 55 42 52 31 6D 43 77 67 77 2B 59 uZ06TUBRlmCwgw+Y
00001600 56 71 61 38 79 4A 5A 6B 71 4C 56 6D 69 4F 6A 5A Vqa8yJZkqLVmiOjZ
00001610 58 62 4C 37 53 34 49 42 58 70 4C 50 6D 6F 72 71 XbL7S4IBXpLPmorq
00001620 57 70 36 41 3D 3D Wp6A==
    
```

Fig.15: Encryption Related Information

Then it appends the “avo2” extension to the file using MoveFileWithprogressW (fig. 16).

```

.text:0107D18D 51          push    ecx
.text:0107D18E 50          push    eax
.text:0107D18F FF 15 68 06 15 01  call   MoveFileWithProgressW
.text:0107D195 C6 45 FC 04  mov     byte ptr [ebp-4], 4
    
```

Fig. 16: Add Extension Using Move File

As seen in figure 17, it has appended “avos2” extensions.

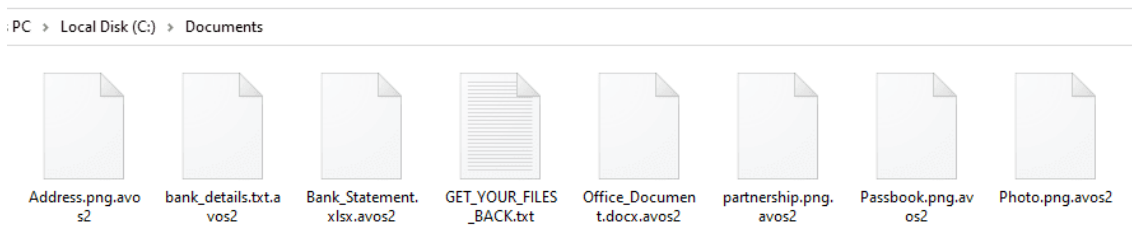


Fig. 17: File with Updated Extension

It writes a ransom note (fig. 18) named “GET\_YOUR\_FILES\_BACK.txt” to each encrypted directory before encryption of the file.



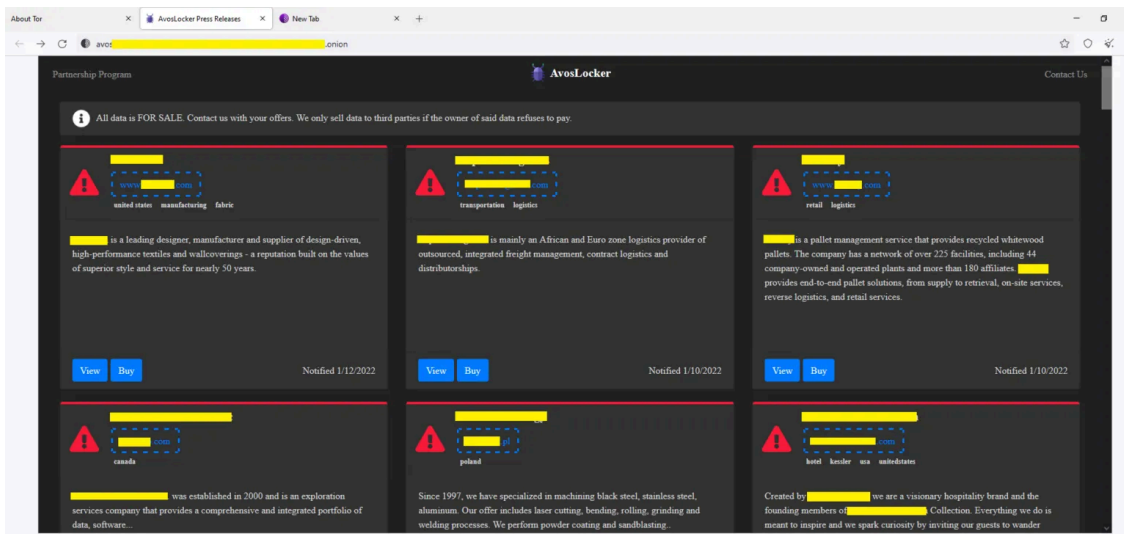


Fig. 20: List of Victims

AvosLocker also offers an affiliate program that provides ransomware-as-a-service (RaaS). They provide “helpful” services to clients such as:

- Supports Windows, Linux & ESXi.
- Affiliate panel
- Negotiation panel with push & sound notifications
- Assistance in negotiations
- Consultations on operations
- Automatic builds
- Automatic decryption tests
- Encryption of network resources
- Killing of processes and services with open handles to files
- Highly configurable builds
- Removal of shadow copies
- Data storage
- DDoS attacks
- Calling services
- Diverse network of penetration testers, access brokers and other contacts

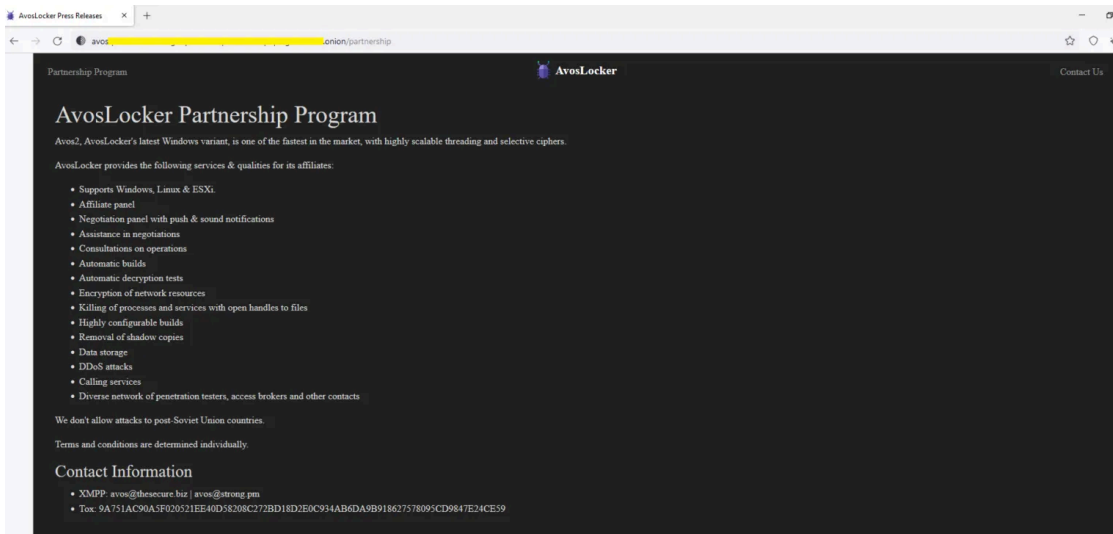


Fig. 21: Partnership Program

## Technical Analysis of AvosLocker Linux Variant

In this case, the AvosLocker malware arrives as an elf file. As shown in figure 22, the analyzed file is x64 based Linux executable file.

```
~$ readelf -h /home/.../samples/7c935.elf
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                   ELF64
  Data:                     2's complement, little endian
  Version:                  1 (current)
  OS/ABI:                   UNIX - System V
  ABI Version:              0
  Type:                     EXEC (Executable file)
  Machine:                  Advanced Micro Devices X86-64
  Version:                  0x1
  Entry point address:      0x4149d0
  Start of program headers: 64 (bytes into file)
  Start of section headers: 1617408 (bytes into file)
  Flags:                    0x0
  Size of this header:      64 (bytes)
  Size of program headers:  56 (bytes)
  Number of program headers: 8
  Size of section headers:  64 (bytes)
  Number of section headers: 31
  Section header string table index: 30
```

Fig. 22: File Details

It's a command-line application having some command-line options (fig. 23).

```
~/samples$ ./10ab.elf -h
AvosLinux | Branch SnowELF
Usage: ./elf <thread count> <path> [path] [path] ...
Example: ./elf 50 /vmfs/volumes/ /home/ /tmp/
Notes:
[path] can be set to 'esxi' as an alias to /vmfs/volumes/
ESXi VMs will be forced to shutdown when ran against ESXi paths.
Run in background: nohup ./elf 50 esxi &
```

Fig. 23: Command-Line Options

The `<Thread count>` parameter as shown above represents the number of threads that can be created to encrypt files simultaneously. It possesses the capability to kill ESXi VMs based on the parameter provided while executing.

Upon execution, the malware first collects information about the number of threads that need to be created. Then it checks for string “vmfs” in the file path provided as a command-line argument (fig. 24).

```
.text:000000000415C01 BE 00 10 00 00      mov     esi, 1000h          ; size
.text:000000000415C06 C7 45 C0 65 73 78 69  mov     dword ptr [rbp+s2], 'ixse'
.text:000000000415C0D C6 45 C4 00          mov     [rbp+var_3C], 0
.text:000000000415C11 C7 45 B0 76 6D 66 73  mov     dword ptr [rbp+needle], 'sfmv'
.text:000000000415C18 4C 89 F7           mov     rdi, r14          ; buf
.text:000000000415C1B C6 45 B4 00          mov     [rbp+var_4C], 0
.text:000000000415C1F E8 1C E6 FF FF      call   _getcwd
.text:000000000415C24 48 8D 55 B0          lea    rdx, [rbp+needle]
.text:000000000415C28 48 85 C0            test   rax, rax
.text:000000000415C2B C6 85 98 EF FF 00   mov     byte ptr [rbp+var_1068], 0
.text:000000000415C32 48 89 95 90 EF FF FF  mov     [rbp+var_1070], rdx
.text:000000000415C39 74 15              jz     short loc_415C50
.text:000000000415C3B 48 89 D6            mov     rsi, rdx          ; needle
.text:000000000415C3E 4C 89 F7           mov     rdi, r14          ; haystack
.text:000000000415C41 E8 9A E4 FF FF      call   _strstr            rdx=[stack]:aVmfs
.text:000000000415C46 48 85 C0            test   rax, rax          aVmfs db 'vmfs',0
.text:000000000415C49 AF 95 98 FF FF FF   setnz byte ptr [rbp+var_1068]
```

Fig. 24: Checks for “vmfs”

After that, it also checks for string “ESXi” in the file path provided as a command-line argument (fig. 25).

```
.text:000000000415C84 4F 8B 7C 35 10     mov     r15, [r13+r14+10h]
.text:000000000415C89 48 8B B5 90 EF FF FF  mov     rsi, [rbp+var_1070] ; needle
.text:000000000415C90 4C 89 FF           mov     rdi, r15          ; haystack
.text:000000000415C93 E8 48 E4 FF FF      call   _strstr
.text:000000000415C98 48 85 C0            test   rax, rax
.text:000000000415C9B 75 D3              jnz    short loc_415C70
.text:000000000415C9D 48 8D 75 C0          lea    rsi, [rbp+s2]      ; needle
.text:000000000415CA1 4C 89 FF           mov     rdi, r15          ; haystack
.text:000000000415CA4 E8 37 E4 FF FF      call   _strstr            [rbp+s2]=[[stack]:aEsxi]
.text:000000000415CA9 48 85 C0            test   rax, rax          aEsxi db 'esxi',0
.text:000000000415CAC 75 C2              jnz    short loc_415C70
.text:000000000415CAE E8 C7               jmp     short loc_415C77
```

Fig. 25: Checks for “ESXi”

If this parameter is found, then it calls a routine to kill the running ESXi virtual machine (fig. 26).

```
.text:000000000415CB7 74 2A              jz     short loc_415CE3
.text:000000000415CB9 BF B0 5A 4F 00     mov     edi, offset aKillingEsxiVms ; "[+] Killing ESXi VMs ..."
.text:000000000415CBE 31 C0              xor     eax, eax
.text:000000000415CC0 E8 DB E3 FF FF      call   _printf
.text:000000000415CC5 BF 30 62 4F 00     mov     edi, offset aEsxcliFormatte ; "esxcli --formatter=csv --format-param=f"
.text:000000000415CCA E8 01 E7 FF FF      call   _system
.text:000000000415CCF BF 05 00 00 00     mov     edi, 5            ; seconds
.text:000000000415CD4 E8 87 E6 FF FF      call   _sleep
.text:000000000415CD9 BF CA 5A 4F 00     mov     edi, offset aOk    ; "[OK]"
.text:000000000415CDE E8 5D EA FF FF      call   _puts
```

Fig. 26: Code to Kill ESXi Virtual Machine

The command used for killing the ESXi virtual machine is as shown in figure 27.

```
esxcli --formatter=csv --format-param=fields=="WorldID,DisplayName" vm process list | tail -n +2 | awk -F '$:' '{system("esxcli vm process kill --type=force --world-id=" $1)}'.
```

Fig. 27: Command to Kill Running ESXi Virtual Machine

Further, AvosLocker drops a ransom note file (fig. 28) at the targeted directory.

```
0000000004153AA ; // starts at 4153AA
0000000004153AF 48 8B BC 24 70 01 00 00 mov     rdi, [rsp+1C8h+filename] ; filename
0000000004153B7 BE 1A 59 4F 00          mov     esi, (offset modes+2) ; modes
0000000004153BC ; try {
0000000004153BC E8 BF F1 FF FF      call   _topen
0000000004153BC ; } // starts at 4153BC
0000000004153C1 4C 89 EF           mov     rdi, r13
0000000004153C4 48 89 C5           mov     rbp, rax          ; void *
0000000004153C7 ; try {
0000000004153C7 E8 A4 F3 FF FF      call   _ZNs01Ev          ; std::string
0000000004153C7 ; } // starts at 4153C7
0000000004153C7 4C 89 E7           mov     rdi, r12          ; void *
0000000004153CF ; try {
0000000004153CF E8 9C F3 FF FF      call   _ZNs01Ev          ; std::string
0000000004153D4 48 85 ED           test   rbp, rbp
0000000004153D7 74 21              jz     short loc_4153FA
0000000004153D9 48 8B BC 24 90 01 00 00  mov     rdi, [rsp+1C8h+ptr] ; ptr
0000000004153E1 48 89 E9           mov     rcx, rbp          ; s
0000000004153E4 BE 01 00 00 00     mov     esi, 1            ; size
0000000004153E9 48 8B 57 E8           mov     rdx, [rdi-10h]   ; n
0000000004153ED E8 6E F5 FF FF      call   _fwrite
0000000004153F2 48 89 EF           mov     rdi, rbp          ; stream
0000000004153F5 E8 96 F0 FF FF      call   _fclose
```

Fig. 28: Create ransom note

After that, it starts creating a list of files that must be encrypted. Before adding a file path to the list, it checks whether it is a regular file or not (fig. 29). Only regular files are added to the encryption list.

```
.text:0000000004154F0                                ; sub_415490+94;j ...
.text:0000000004154F0 48 89 DF      mov     rdi, rbx                                ; dirp
.text:0000000004154F3 E8 28 F2 FF  call   _readdir
.text:0000000004154F8 48 85 C0      test   rax, rax
.text:0000000004154FB 0F 84 BF 00 00 00  jz     loc_4155C0
.text:000000000415501
.text:000000000415501                                loc_415501:
.text:000000000415501 0F B6 50 12  movzx  edx, byte ptr [rax+12h]                ; CODE XREF: sub_415490+129;j
.text:000000000415505 80 FA 08      cmp    dl, 8                                  Check for Regular File
.text:000000000415508 0F 84 C2 00 00 00  jz     loc_4155D0
.text:00000000041550E 80 FA 04      cmp    dl, 4                                  Check for Directory
.text:000000000415511 75 DD      jnz   short loc_4154F0
.text:000000000415513 4C 8D 60 13  lea   r12, [rax+13h]
.text:000000000415517 4C 89 FF      mov   rdi, r15
.text:00000000041551A B9 03 00 00 00  mov   ecx, 3
.text:00000000041551F 4C 89 E6      mov   rsi, r12
```

Fig. 29: Checks File Info

AvosLocker skips the ransom note file and any files with the extension “avoslinux” from adding into the encryption list (fig. 30).

```
000000000414E60                                ; _unwind { // __gxx_personality_v0
000000000414E60 48 89 5C 24 F0  mov   [rsp+var_10], rbx
000000000414E65 48 89 6C 24 F8  mov   [rsp+var_8], rbp
000000000414E6A BA 05 00 00 00  mov   edx, 5                                ; cflags
000000000414E6F 48 81 EC 18 10 00 00  sub   rsp, 1018h
000000000414E76 BE 31 59 4F 00  mov   esi, offset pattern                    ; "\\.(key|avoslinux)$"
000000000414E7B 48 89 FB      mov   rbx, rdi
000000000414E7E E8 BD F6 FF FF  call  _regcomp
000000000414E83 89 C7      mov   edi, eax                                ; errcode
000000000414E85 31 C0      xor   eax, eax
000000000414E87 85 FF      test  edi, edi
```

Fig. 30: Skip “avoslinux” Extension File

Then it calls the mutex lock/unlock API for thread synchronization as shown in figure 31.

```
000000000415D80                                loc_415D80:
000000000415D80 BF C0 B8 78 00  mov   edi, offset stru_78B8C0                ; CODE XREF: main+213;j
000000000415D85 E8 56 E9 FF FF  call  _pthread_mutex_lock                    ; mutex
000000000415D8A BF C0 B8 78 00  mov   edi, offset stru_78B8C0                ; mutex
000000000415D8F C6 05 52 5B 37 00 01  mov   cs:byte_78B8E8, 1
000000000415D96 E8 D5 E5 FF FF  call  _pthread_mutex_unlock
000000000415D9B 48 83 BD 90 EF FF FF 01  cmp   [rbp+var_1222], 1
000000000415DA3 7E 63      jle   short loc_415E08
000000000415DA5 BB 02 00 00 00  mov   ebx, 2
000000000415DAA EB 11      jmp   short loc_415DBD
```

Fig. 31: Lock-Unlock Mutex for Thread Synchronization

Based on the number of threads specified, it creates concurrent CPU threads (fig. 32). This helps in encrypting different files simultaneously at a very fast speed.

```
000000000415DB0                                loc_415DB0:
000000000415DB0 48 83 C3 01  add   rbx, 1                                ; CODE XREF: main+299;j
000000000415DB4 48 39 9D 90 EF FF FF  cmp   [rbp+var_1222], rbx
000000000415DBB 7C 4B      jl    short loc_415E08
000000000415DBD                                loc_415DBD:
000000000415DBD 48 8B 85 98 EF FF FF  mov   rax, [rbp+var_1068]                    ; CODE XREF: main+26A;j
000000000415DC4 31 F6      xor   esi, esi                                ; attr
000000000415DC6 48 89 D9      mov   rcx, rbx                                ; arg
000000000415DC9 BA B0 8C 41 00  mov   edx, offset start_routine              ; start_routine
000000000415DCE 48 8D 3C D8  lea   rdi, [rax+rbx*8]                        ; newthread
000000000415DD2 E8 A9 E4 FF FF  call  _pthread_create
000000000415DD7 85 C0      test  eax, eax
000000000415DD9 74 D5      jz    short loc_415DB0
000000000415DDB BF E0 62 4F 00  mov   edi, offset aErrorPthreadCr          ; "Error: pthread_create() failed"
000000000415DE0 E8 5B E9 FF FF  call  _puts
```

Fig. 32: Create Threads in Loop

AvosLocker’s Linux variant makes use of Advanced Encryption Standard (AES) and elliptic-curve cryptography (ECC) algorithms for data encryption.

File-related information along with the encryption key used might be encrypted and then encoded with base 64 formats. This encoded information is added at the end of each encrypted file (fig. 33).

```
133980 7A 43 4F 4A 25 B5 76 40 14 20 0C 0E 6C B8 AC 29 15 9F 3C 90 28 E7 36 D5 EB B9 9A 3C 72 36 8D 08 E9 18 5F zC0J%1v@1 9z1q1) $f<E(t6 0ll U<r6i00+
134015 EF 94 18 01 A0 ED CA 28 0B C3 D6 A8 36 70 71 32 CF 00 18 78 39 E8 CF E1 75 AD 9D C1 16 EB 7A 88 9E E3 65 n0:0ap^ (σ | r z 6 p q 2 - + x 9 0 + 6 u j + 6 z 6 t m e
134050 1B 2C 8F 6F CD FB 8C 88 00 4F BF D7 46 FA 45 9B DE 41 8C 26 66 0D EF 56 0D EA 1B 60 2B 49 74 67 44 84 9C -,Ao=viē 01 F:Ec Ai&f:nVr:0-+ItgD&f
134085 C6 8A 83 82 24 08 C7 8D 23 FF 81 21 D4 0D EB 2F 6A B8 F7 77 41 2A 49 2E FD CE BC 94 3A BA 0D F4 EC 26 5A heâes□|i# u! k r 6 / j = w A * I . ² ³ 0 : j | ∞ 6 Z
134120 EE 88 2A D1 BC DC 98 13 2B 1B 41 A2 07 5D F4 E8 2D F5 4F 3A 44 75 24 DB E1 96 33 C9 B7 53 6B 05 8D 1B 5B eē*# | V ! ! + A 0 * ) [ 0 - ] 0 : D u S | 8 0 3 p S k * i - [
134155 3F 44 DB A4 2D 73 F5 6B 40 65 5B 1C 72 69 53 C7 DD 78 A5 B4 45 79 2B 35 43 65 96 3A B8 65 DE F4 85 71 0C ?D | n - s | k @ e ( - r i s | | x N | E y + 5 C e u : 7 e | | a q 0
134190 71 61 A6 98 9E 01 EE A8 98 91 0C 2C E3 57 0A 0B 19 0B 0A 41 91 B4 C0 69 D3 73 ED 65 97 99 85 FF 3C 1E B3 qa^c#oεzjv&o, nV#o:0A&: | i | s p e u 0 a < A |
134225 81 81 42 AE 4A 02 B9 86 14 15 0C 36 FE 4F DF 5B 9A 45 77 EF F7 21 12 4F 71 B1 45 D4 5D E9 DA 4A 28 36 78 üüB<J&|â|š06m0 (UEnw=:0q|E|)0rJ (6x
134260 6C 92 BC B6 8B BE A7 57 EE 95 01 A0 8E 5D 25 AA 04 B3 56 4E C7 D1 6D 77 21 1E FB 11 3B B8 D5 93 BF 0D 58 l&|i|d|Weo0&A|&#>|VN|mmw|v-;7 r 0 z X
134295 24 D6 A6 23 3E 5D 38 1B B2 C4 B5 FC 5C 04 CF 3C FC 09 42 4A 65 4D 71 52 58 6B 4B 6C 75 57 62 75 56 34 72 $r#>|8-8-| "\<#<"oBjEgRXkklUwbuV4r
134330 57 4B 49 55 45 30 65 63 68 4A 32 71 32 6E 78 45 42 32 58 5A 73 6E 47 48 38 4B 71 51 39 38 37 50 41 6C 6F WKIE0echJ2q2nxEB2XZsnGH8kkQ987PALo
134365 71 46 7A 67 50 6C 68 33 66 37 31 47 79 75 70 33 68 61 2B 59 0A 61 7A 41 30 71 6A 31 56 58 77 43 35 64 79 qFzgLh3f71Gyup3ha+Y&azA0qj1VXwC5dy
134400 6B 2B 73 5A 4D 48 57 43 4A 4C 4D 45 38 61 74 71 44 47 42 58 32 64 6D 63 7A 61 49 72 65 38 4D 75 37 4A 39 k+sZMHWCJLME&atqDGBX2dmczaIre8Mu7J9
134435 4A 68 72 37 36 48 58 4B 64 51 6E 69 6B 62 6B 4A 6B 34 78 62 31 71 0A 67 50 32 55 34 4F 73 51 6D 43 49 Jhr76HKdQnIkiRk4xb1q0gP2U40s0mCI
134470 69 69 77 74 47 43 4C 30 50 52 44 6B 3D 0A iitwGL0PRDK=
```

Fig. 33: File-related Info added at the end

Figure 34 shows the malware appending the extension “.avoslinux” to the encrypted file names.

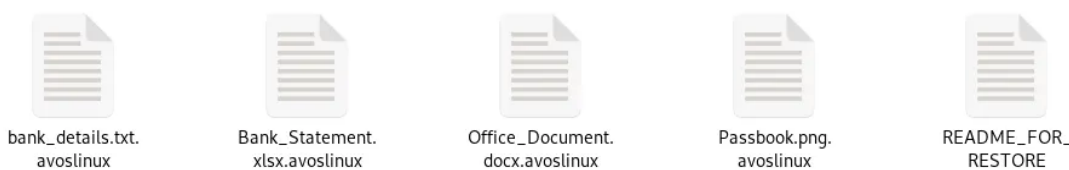


Fig. 34: Append file extension “.avoslinux” after encryption

Before starting file encryption, it creates a ransom note named “README\_FOR\_RESTORE “. The content of this ransom note is shown in figure 35.

```
1 README_FOR_RESTORE
Attention!
Your files have been encrypted.
We highly suggest not shutting down your computer in case encryption process is not finished, as your files may get corrupted.
In order to decrypt your files, you must pay for the decryption key & application.
You may do so by visiting us at http://avosjon[redacted].onion.
This is an onion address that you may access using Tor 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 they'll have to pay will go up significantly.
The corporations whom don't pay or fail to respond in a swift manner can be found in our blog, accessible at http://avos[redacted].onion

Your ID: 2b[redacted]b57
```

Fig. 35: Ransom Note

The ransom note instructs the victim not to shut down the system in case encryption is in progress to avoid file corruption. It asks the victim to visit the onion address with a TOR browser to pay the ransom and to obtain the decryption key and decryption application.

### Indicators of Compromise (IOCs):

Windows: C0A42741EEF72991D9D0EE8B6C0531FC19151457A8B59BDCF7B6373D1FE56E02

Linux: 7C935DCD672C4854495F41008120288E8E1C144089F1F06A23BD0A0F52A544B1

URL:  
 hxxp://avosjon4pfh3y7ew3jdwz6ofw7lljcxlbk7hcxmnlh5kvf2akcqjad[.]onion.

hxxp://avosqhx72b5ia23dl5fgwcpndkctuzqvh2iefk5imp3pi5gfhel5klad[.]onion

### TTP Map:

<b>Initial Access</b>	<b>Execution</b>	<b>Defense Evasion</b>	<b>Discovery</b>	<b>Impact</b>
Phishing (T1566)	User Execution (T1204)	Obfuscated Files or Information (T1027)	System Information Discovery (T1082)	Data Encrypted for Impact (T1486)
			File and Directory Discovery (T1083)	Inhibit System Recovery (T1490)

---

Source: <https://blog.qualys.com/vulnerabilities-threat-research/2022/03/06/avoslocker-ransomware-behavior-examined-on-windows-linux>