

# Revix Linux Ransomware

By Vishal Thakur

Published: 2023-04-06 · Archived: 2026-04-05 18:52:56 UTC

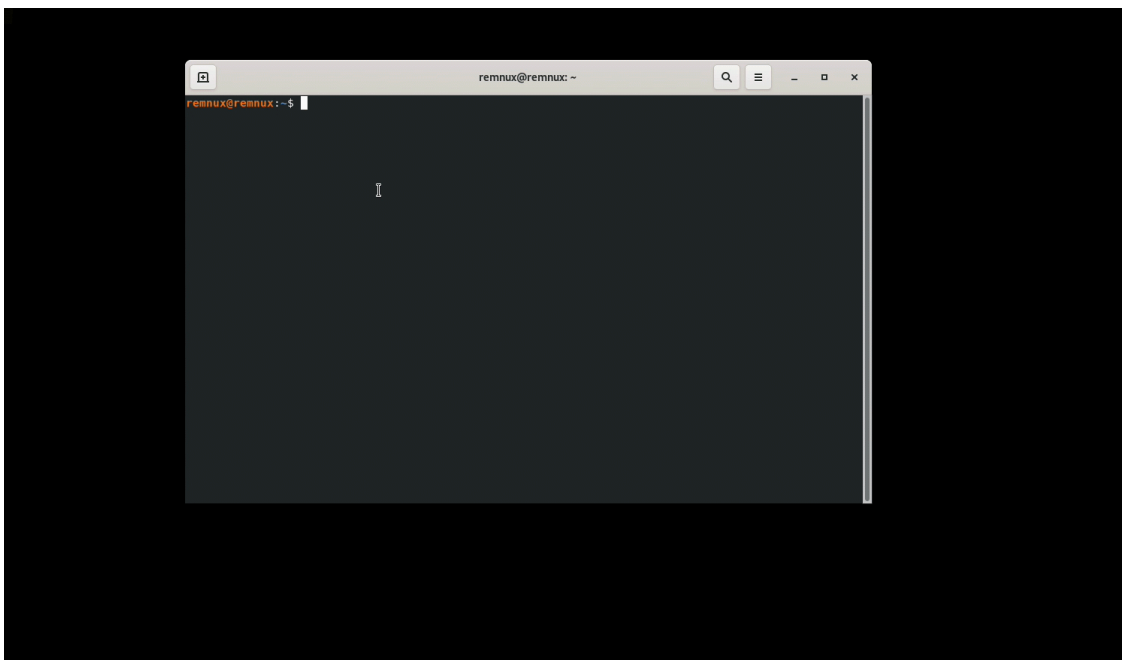


A duplicate copy of this article by Vishal can also be found here:  
<https://angle.ankura.com/post/102hcnv/revix-linux-ransomware>

In the first half of 2021, we started to see the REvil ransomware operators targeting Linux-based systems with a new Linux version of the more commonly seen Windows version of the same ransomware. There have been a few versions of this Linux-based malware since then.

In this publication, we take a look at the latest version, 1.2a.

Press enter or click to view image in full size



[Malpedia link](#)

[YouTube link](#)

## YARA Rulesets

[https://github.com/YaraExchange/yarasigs/blob/master/ransomware/crime\\_lin\\_revil.yar](https://github.com/YaraExchange/yarasigs/blob/master/ransomware/crime_lin_revil.yar)

[https://github.com/YaraExchange/yarasigs/blob/master/ransomware/crime\\_lin\\_revix.yar](https://github.com/YaraExchange/yarasigs/blob/master/ransomware/crime_lin_revix.yar)

## Quick Snapshot

```
Class: ELF64
Type: Dynamically Linked
Machine: X86-64
Number of section headers: 28
Entry Point: 0x401650
callq: __libc_start_main@plt
MD5: c83df66c46bcbc05cd987661882ff061
```

## Introduction

The execution of this malware is straight forward. It traverses through the directories targeted by it and encrypts the files present in those directories. Once encryption is complete, it drops a ransom note in the directory with the usual ransom message and instructions on how to pay the bad actors to get the decryption key.

This malware requires a couple of parameters to be passed to it in order for it to successfully execute. It also requires to be run with escalated privileges in order to be able to successfully encrypt files on the disk.

One of the main targets for this malware is VMware's ESX platform, which we've seen before in a different Linux ransomware, [Darkside](#).

This malware is not able to encrypt data if being executed by a non-privileged user. It also checks the files in the target directories to see if they are already encrypted.

## Analysis

For the purpose of this publication, we analyse this malware both statically and dynamically. We switch between the two methodologies as we go through the analysis process.

A quick look at section .init:

```
000000000401268 <.init>:
401268: 48 83 ec 08      sub  rsp,0x8
40126c: 48 8b 05 85 4d 21 00 mov  rax,QWORD PTR [rip+0x214d85] # 615ff8
<usleep@plt+0x2149b8>
401273: 48 85 c0         test rax,rax
401276: 74 05           je   40127d <free@plt-0x23>
401278: e8 03 02 00 00 call 401480 <__gmon_start__@plt>
40127d: 48 83 c4 08     add  rsp,0x8
401281: c3             ret
```

Section .text:

```

000000000401650 <.text>:
401650: 31 ed          xor  ebp,ebp
401652: 49 89 d1       mov  r9,rdx
401655: 5e            pop  rsi
401656: 48 89 e2       mov  rdx,rsq
401659: 48 83 e4 f0    and  rsp,0xfffffffffff0
40165d: 50            push rax
40165e: 54            push rsp
40165f: 49 c7 c0 90 02 41 00 mov  r8,0x410290
401666: 48 c7 c1 20 02 41 00 mov  rcx,0x410220
40166d: 48 c7 c7 7f 68 40 00 mov  rdi,0x40687f
401674: e8 b7 fd ff    call 401430 <__libc_start_main@plt>
000000000401430 <__libc_start_main@plt>:
401430: ff 25 aa 4c 21 00 jmp  QWORD PTR [rip+0x214caa] # 6160e0
<usleep@plt+0x214aa0>
401436: 68 19 00 00 00 push 0x19
40143b: e9 50 fe ff ff jmp  401290 <free@plt-0x10>

```

## Functions

The malware loads a number of functions upon initialisation. Following some of the interesting ones we are able to extract useful information that can be used to understand the flow of execution and write some detections as we'll see later in this article.

Press enter or click to view image in full size

```

(gdb) c
Continuing.

Breakpoint 3, 0x000000000401320 in puts@plt ()
(gdb) x/20l $pc
=> 0x401320 <puts@plt>: jmpq  *0x214d32(%rip) # 0x616058 <puts@got.plt>
0x401326 <puts@plt+6>: pushq $0x8
0x40132b <puts@plt+11>: jmpq  0x401290
0x401330 <fread@plt>: jmpq  *0x214d2a(%rip) # 0x616060 <fread@got.plt>
0x401336 <fread@plt+6>: pushq $0x9
0x40133b <fread@plt+11>: jmpq  0x401290
0x401340 <pthread_cond_wait@plt>: jmpq  *0x214d22(%rip) # 0x616068 <pthread_cond_wait@got.plt>
0x401346 <pthread_cond_wait@plt+6>: pushq $0xa
0x40134b <pthread_cond_wait@plt+11>: jmpq  0x401290
0x401350 <fclose@plt>: jmpq  *0x214d1a(%rip) # 0x616070 <fclose@got.plt>
0x401356 <fclose@plt+6>: pushq $0xb
0x40135b <fclose@plt+11>: jmpq  0x401290
0x401360 <opendir@plt>: jmpq  *0x214d12(%rip) # 0x616078 <opendir@got.plt>
0x401366 <opendir@plt+6>: pushq $0xc
0x40136b <opendir@plt+11>: jmpq  0x401290
0x401370 <strlen@plt>: jmpq  *0x214d0a(%rip) # 0x616080 <strlen@got.plt>
0x401376 <strlen@plt+6>: pushq $0xd
0x40137b <strlen@plt+11>: jmpq  0x401290
0x401380 <getopt_long@plt>: jmpq  *0x214d02(%rip) # 0x616088 <getopt_long@got.plt>
0x401386 <getopt_long@plt+6>: pushq $0xe
(gdb) █

```

Malware functions loaded upon initialisation

↑	00000000:004014d0	ff 25 5a 4c 21 00	jmp qword [rel 0x616130]
↑	00000000:004014d6	68 23 00 00 00	push 0x23
↑	00000000:004014db	e9 b0 fd ff ff	jmp 0x401290
↑	00000000:004014e0	ff 25 52 4c 21 00	jmp qword [rel 0x616138]
↑	00000000:004014e6	68 24 00 00 00	push 0x24
↑	00000000:004014eb	e9 a0 fd ff ff	jmp 0x401290
↑	00000000:004014f0	ff 25 4a 4c 21 00	jmp qword [rel 0x616140]
↑	00000000:004014f6	68 25 00 00 00	push 0x25
↑	00000000:004014fb	e9 90 fd ff ff	jmp 0x401290
↑	00000000:00401500	ff 25 42 4c 21 00	jmp qword [rel 0x616148]
↑	00000000:00401506	68 26 00 00 00	push 0x26
↑	00000000:0040150b	e9 80 fd ff ff	jmp 0x401290
↑	00000000:00401510	ff 25 3a 4c 21 00	jmp qword [rel 0x616150]
↑	00000000:00401516	68 27 00 00 00	push 0x27
↑	00000000:0040151b	e9 70 fd ff ff	jmp 0x401290
↑	00000000:00401520	ff 25 32 4c 21 00	jmp qword [rel 0x616158]
↑	00000000:00401526	68 28 00 00 00	push 0x28

Function sequence during execution

### Initialization

Let's take a quick look at the program initialization:

	00000000:0040687e	c3	ret
rax	00000000:0040687f	55	push rbp
	00000000:00406880	48 89 e5	mov rbp, rsp
	00000000:00406883	48 83 ec 30	sub rsp, 0x30
	00000000:00406887	89 7d dc	mov [rbp-0x24], edi
	00000000:0040688a	48 89 75 d0	mov [rbp-0x30], rsi
	00000000:0040688e	c7 45 e8 00 00 00 00	mov dword [rbp-0x18], 0
	00000000:00406895	83 7d dc 01	cmp dword [rbp-0x24], 1
	00000000:00406899	7f 14	jg 0x4068af
	00000000:0040689b	bf 58 07 41 00	mov edi, 0x410758
	00000000:004068a0	e8 7b aa ff ff	call revil.elf!puts@plt

Execution initialisation

Press enter or click to view image in full size

```
ASCII "Revix 1.2a \r\nUsage example: elf.exe --path /vmfs/ --threads 5\r\n--silent (-s) use for not stoping
```

Parameters for the command-line arguments

### Execution

When executed as a non-privileged user, the malware is not able to achieve full execution.

As we can see in the image below, the malware has been provided the directory 'here' for the purpose of this analysis:

```
write(1, 0xcc32a0, 13Path: here/
) = 13
```

The malware tries to access the data in this directory for read/write and is not successful as the image below shows:

```
openat(AT_FDCWD, 0xcc37d0, 0_RDONLY|0_NONBLOCK|0_CLOEXEC|0_DIRECTORY) = 3
fstat(3, 0x7fff407d6aa0) = 0
getdents64(3, 0xcc6640, 32768) = 320
```

The malware also tries to encrypt the test file that we have provided for the purpose of this analysis in the target directory but the encryption process fails as that action requires higher privileges:

Press enter or click to view image in full size

```
write(1, 0x88b2a0, 48Error create note in dir here//vemar-readme.txt
)
= 48
close(3[here//test.txt] semms to be protected by os but let's encrypt anyway...
)
= 0
```

As a result, the execution fails to achieve the desired outcome for the malware, as shown by the result in the image below:

Press enter or click to view image in full size

```
ij|-----|ji i
ij| ENCRYPTED |ji i
ij| - - - - -|ji i
ij| 00000000 |ji i
ij|  FILES   |ji i
ij|          |ji i
ij| 00000000 |ji i
ij|   MBs    |ji i
ij|_____ |ji i
iji iji tYiji iji i
iji iji tYiji iji i
```

Another point of interest from this failed execution is that the malware attempted to execute a `esxcli` command but was not able to do so:

```
sh: 1: esxcli: not found
```

All of this changes when we run the malware with escalated privileges.

Firstly, the malware is able to access the data in the target directory:

```
openat(AT_FDCWD, "here/", O_RDONLY|O_NONBLOCK|O_CLOEXEC|O_DIRECTORY) = 3
fstat(3, {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
getdents64(3, /* 3 entries */, 32768) = 80
```

Next, we can see that the malware is able to perform read/write functions on the data in the target directories, resulting in successful encryption of that data:

```
Encrypting [here//test.txt]
clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=0, tv_nsec=10000}, NULL) = 0
```

We can see from the image below that the malware is able to write the ransom-note text file to the disk:

```
fstat(5, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
write(5, "---=== Welcome. Again. ===---\n\n["..., 2311) = 2311
close(5) = 0
close(3) = 0
```

And finally, we can see that the execution is completed successfully, resulting in the data present in the target directory being encrypted:

```
ij| ENCRYPTED |ji
ij|- - - - -|ji
ij| 00000001 |ji
ij|  FILES  |ji
```

The file we provided in the target directory is now encrypted and a ransom-note is created in the same directory:

```
remnux@remnux:~/Documents$ ls here
test.txt.vemar  vemar-readme.txt
```

The malware also checks if the data in the target directory is already encrypted. To demonstrate this, we ran the malware against the same target directory one more time.

### Get Vishal Thakur’s stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

Upon execution, the malware runs a check on the data present in the target directory and identifies it to be already encrypted:

Press enter or click to view image in full size

```
futex(0xd2c5e8, FUTEX_WAIT_PRIVATE, 0, NULL[here//test.txt.vemar] already encrypted
```

As a result, the execution ends up with no data being encrypted:

```
j| ENCRYPTED |j
j|- - - - -|j
j| 00000000 |j
j|  FILES  |j
```

## VMware ESX targeting

This malware also tries to use the esxcli, the command line interface for VMware ESX platform. Let's take a quick look at the parameters passed to esx as command-line arguments.

```
esxcli --formatter=csv --format-param=fields=="WorldID,DisplayName" vm process list | awk -F "\"*,\""
```

### vm process list

List the virtual machines on this system. This command currently will only list running VMs on the system.

### vm process kill

Used to forcibly kill Virtual Machines that are stuck and not responding to normal stop operations.

#### — type

There are three types of VM kills that can be attempted: [soft, hard, force].

#### — world-id | -w

The World ID of the Virtual Machine to kill. This can be obtained from the 'vm process list' command (required)

So basically what these esx command-line arguments are doing is shutting down all VMs running on the ESX platform.

The idea is to run the malware targeting the '/vmfs' directory and encrypt all the data present in that directory so all the VMs are rendered inoperable until the data is decrypted.

This targeting is similar to that seen in DarkSide's Linux variant.

## Command-line Arguments

The malware requires the following parameters to be passed for its execution to begin:

**elf.exe — path /vmfs/ — threads 5**

It also allows the '— silent' option that executes the malware without stopping the VMs

**— silent (-s) use for not stopping VMs mode \***

Parameter	Purpose
--path	Specifies the path of the data that needs to be encrypted
--threads	Specifies the number of threads, by default the malware uses 50 threads
--silent	Executes the malware without stopping the VMs running on ESX



## uname -a && echo “ | “ && hostname

```
RAX 00000000004110a8 ASCII "uname -a && echo \" | \" && hostname"
RCX 0000000000000072
RDX 000000000000002c
RBX 0000000000410220
RSP 00007ffc29fe2068
RBP 00007ffc29fe20a0
RSI 00000000004110cb
RDI 00000000004110a8 ASCII "uname -a && echo \" | \" && hostname"
R8 000000000080ca10 ASCII "4n0Nu4GmaHf40RvBhHclpampcsKyZMxfSelgMmZE/nI="
R9 000000000080d3e0
```

And we can see the result in the stack:

Press enter or click to view image in full size

```
|0000000000405b61|a[@.....|return to 0x0000000000405b61
|000000000080ccb0|[].....|ASCII "Linux remnux 5.4.0-72-generic #80-Ubuntu SMP Mon Apr 12 17:35:00 UTC
|000000000080c8c0|[].....|
```

The info is then passed through the registers:

Press enter or click to view image in full size

```
RDX 000000000080cb00 ASCII "BRcu0S8WVoNH0t5LRPQzvUgP/6vWUnx2FYqbfTrVqvybgAndwMKrWwpwHJHUsPeSF0tFZ7Cx6ZYGFIorKnRf
RBX 00007fc439b4bc93
RSP 00007ffc29fe1878
RBP 00007ffc29fe1df0
RSI 00007ffc29fe1db8
RDI 000000000080cb00 ASCII "BRcu0S8WVoNH0t5LRPQzvUgP/6vWUnx2FYqbfTrVqvybgAndwMKrWwpwHJHUsPeSF0tFZ7Cx6ZYGFIorKnRf
R8 00000000ffffffff
R9 00000000000000b0
R10 0000000000410fbc ASCII "s\", \"os\": \"%s\", \"ext\": \"%s\"}"
R11 000000000080cb00 ASCII "BRcu0S8WVoNH0t5LRPQzvUgP/6vWUnx2FYqbfTrVqvybgAndwMKrWwpwHJHUsPeSF0tFZ7Cx6ZYGFIorKnRf
R12 00007ffc29fe1e00
R13 0000000000410f80 ASCII "{ \"ver\": %d, \"pid\": \"%s\", \"sub\": \"%s\", \"pk\": \"%s\", \"uid\": \"%s\", \"sk\": \"%s\"
R14 00007ffc29fe1f40
```

And the end-result is created in the form of this config with the victim information:

```
{"ver":512,
"pid":"$2a$12$D3Wk4d.cy0e0EiVqDPJe1.06OMR3duoMRIH78i7;
"Sub":"8639",
"pk":"4nONu4GmaHf40RvBhHclpampcsKyZMxfSelgMmZE/nI=",
"uid":"7E73E5407E73E540",
"sk":"BRcu0S8WVoNH0t5LRPQzvUgP/6vWUnx2FYqbfTrVqvybg
UuNGEKZv5FH7XwzXXu36tLCA==",
"Os":"linux",
"Ext":"vemar"}
```

## Encryption

The malware uses Salsa20 encryption algorithm (just like its Windows variant) to encrypt the data and here’s the pseudocode for the function that implements it:

Press enter or click to view image in full size

```

void FUN_00401ad3(uint *param_1,uint *param_2,int param_3)
{
    uint *local_18;

    param_1[1] = *param_2;
    param_1[2] = param_2[1];
    param_1[3] = param_2[2];
    param_1[4] = param_2[3];
    if (param_3 == 0x100) {
        local_18 = param_2 + 4;
        DAT_0061a318 = "expand 32-byte kexpand 16-byte kvmx-*";
    }
    else {
        DAT_0061a318 = "expand 16-byte kvmx-*";
        local_18 = param_2;
    }
    param_1[0xb] = *local_18;
    param_1[0xc] = local_18[1];
    param_1[0xd] = local_18[2];
    param_1[0xe] = local_18[3];
    *param_1 = (int)DAT_0061a318[1] << 8 | (int)*DAT_0061a318 | (int)DAT_0061a318[2] << 0x10 |
              (int)DAT_0061a318[3] << 0x18;
    param_1[5] = (int)DAT_0061a318[5] << 8 | (int)DAT_0061a318[4] | (int)DAT_0061a318[6] << 0x10 |
              (int)DAT_0061a318[7] << 0x18;
    param_1[10] = (int)DAT_0061a318[9] << 8 | (int)DAT_0061a318[8] | (int)DAT_0061a318[10] << 0x10 |
              (int)DAT_0061a318[0xb] << 0x18;
    param_1[0xf] = (int)DAT_0061a318[0xd] << 8 | (int)DAT_0061a318[0xc] |
              (int)DAT_0061a318[0xe] << 0x10 | (int)DAT_0061a318[0xf] << 0x18;
    return;
}

```

## Mitigation

### Detections

The malware runs this command to determine machine info:

```
uname -a && echo " | " && hostname
```

The malware tries to query this directory:

```
/dev/urandom
```

The malware runs this command to stop VMs running on the ESX platform in order to encrypt the data on those VMs:

```
esxcli --formatter=csv --format-param=fields=="WorldID,DisplayName" vm process list | awk -F "\"*,\""
```

### Typos:

In some instances, typos that malware authors commit to the code are useful in detection of the malware or similar code used in other malware. These are a couple of typos we found in this variant of Revix:

```
--silent (-s) use for not stoping VMs modesemms to be protected by os but let's encrypt anyway...
```

## Conclusion

As we can see in the analysis notes above, the execution is a bit clunky in this variant and requires multiple conditions to be met before the ransomware is successful in encrypting data. The malware needs to be executed as a command-line argument with elevated privileges and specified target directories and number of threads. If the malware is not run in silent mode, it will try to stop the VMs which could trigger off a detection and quite possibly fail to encrypt data on the VMs due to reduced/restricted access.

## References

[ESXi 7.0 U3 ESXCLI Command Reference](#)

[DarkSide on Linux: Virtual Machines Targeted](#) — Naiim, M.,2021

[getdents64\(2\) — Linux man page](#)

[Code Analysis details by Intezer Analyse](#)

---

Source: <https://malienist.medium.com/revix-linux-ransomware-d736956150d0>