Cicada 3301 - Ransomware-as-a-Service - Technical Analysis

By Simon Hertzberg

Published: 2024-08-30 · Archived: 2026-04-05 13:58:29 UTC

Threat Insights

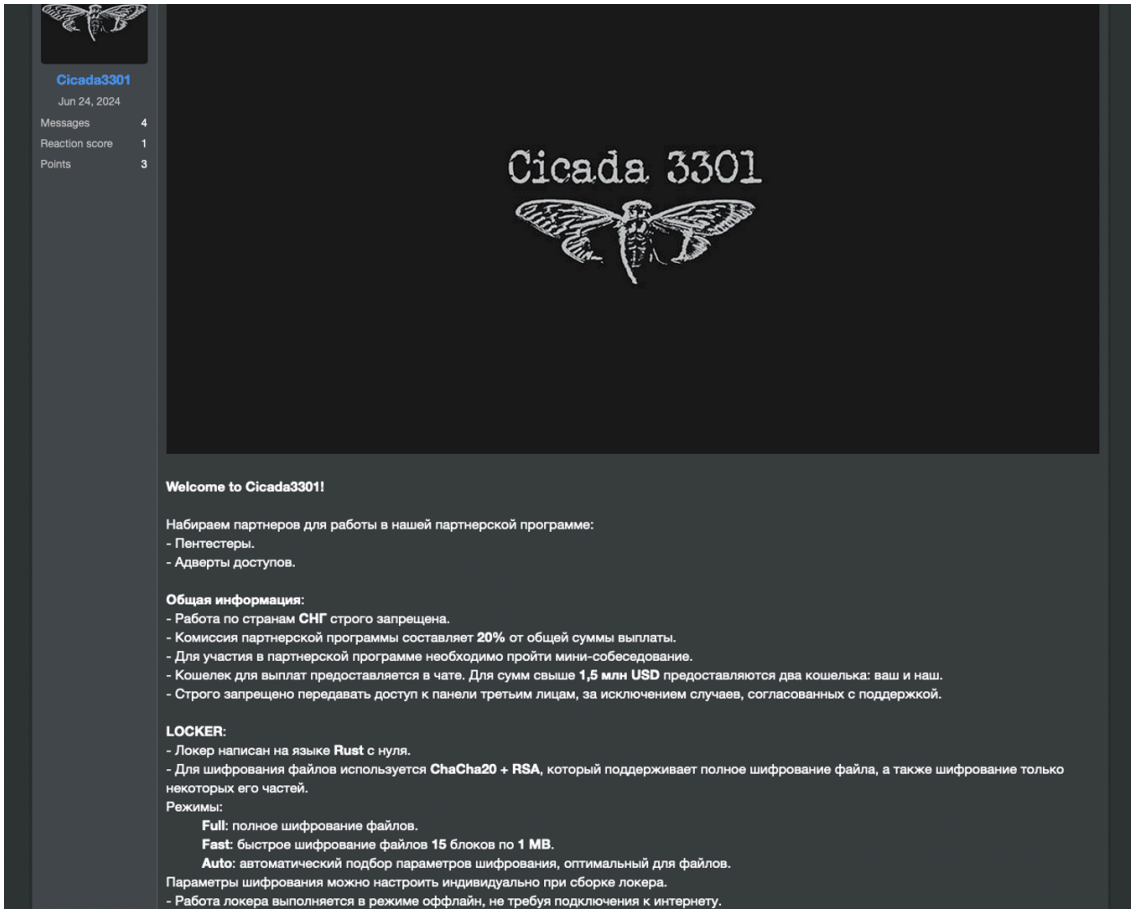
Dissecting the Cicada

A new ransomware group calling themselves Cicada3301 was first observed in June 2024, when they posted four victims on a victim blog. Since then, Cicada3301 has added more victims to this list. The name Cicada3301 appears to be taken from an Internet cryptography game a few years ago, but there is nothing to tie this ransomware group to that phenomenon.

Truesec has now investigated an incident with this new ransomware group and can publish what we have found.

Cicada3301 – A New Ransomware-as-a-Service

The Cicada3301 appears to be a traditional ransomware-as-a-service group that offers a platform for double extortion, with both a ransomware and a data leak site, to its affiliates. The first published leak on the group's data leak site is dated June 25, 2024. Four days later, on June 29, the group published an invitation to potential affiliates to join their ransomware-as-a-service platform on the cybercrime forum Ramp.



Cicada3301 announces its affiliate program on Ramp.

As advertised above, The Cicada3301 group uses a ransomware written in Rust for both Windows and Linux/ESXi hosts. This report will focus on the ESXi ransomware, but there are artifacts in the code that suggest that the Windows ransomware is the same ransomware, just with a different compilation.

While more and more ransomware groups are adding ESXi ransomware to their arsenal, only a few groups are known to have used ESXi ransomware written in Rust. One of them is the now-defunct Black Cat/ALPHV ransomware-as-a-service group. Analysis of the code has also shown several similarities in the code with the ALPHV ransomware.

The Cicada3301 ransomware has several interesting similarities to the ALPHV ransomware.

- Both are written in Rust
- Both use ChaCha20 for encryption
- Both use almost identical commands to shutdown VM and remove snapshots[1]
- Both use –ui command parameters to provide a graphic output on encryption
- Both use the same convention for naming files, but changing “RECOVER-“ransomware extension”-FILES.txt” to “RECOVER-“ransomware extension”-DATA.txt”[2]
- How the key parameter is used to decrypt the ransomware note

Below is an example of code from Cicada3301 that is almost identical to ALPHV.

Example of code shared between ALPHV and Cicada3301.

Analysis of the Threat Actor

The initial attack vector was the threat actor using valid credentials, either stolen or brute-forced, to log in using ScreenConnect. The IP address 91.92.249.203, used by the threat actor, has been tied to a botnet known as “Brutus” that, in turn, has been linked to a broad campaign of password guessing various VPN solutions, including ScreenConnect. This botnet has been active since at least March 2024, when the first article about it was published, but possibly longer.[3]

The IP address used in this initial login was used a few hours before the threat actor started to conduct actions on the systems, so it is highly unlikely that an access broker could compromise the system and pass on the access to a buyer in the span of a few hours unless there was an established connection between them.

This could mean that either (A) the threat actor behind the Brutus botnet is directly connected to the Cicida3301 ransomware group or (B) the use of the IP address by two separate threat actors, both using them to compromise victims using ScreenConnect, is purely coincidental. As far as we could observe, this IP address was still part of the “Brutus” botnet at the time of the ransomware attack.

The timeline is also interesting as the Brutus botnet activity began on March 18, two weeks after it was reported that the BlackCat/ALPHV ransomware group conducted an apparent exit scam and ceased their operations.[4]

It is possible that all these events are related and that part of the BlackCat group has now rebranded themselves as Cicada3301 and teamed up with the Brutus botnet, or even started it themselves, as a means to gain access to potential victims, while they modified their ransomware into the new Cicada3301. Having easy access to a reliable initial access broker can be a way to offer a more “complete” service for the group’s affiliates.

The group could also have teamed up with the malware developer behind ALPHV. This individual appears to have worked for several different ransomware groups in the past.[5]

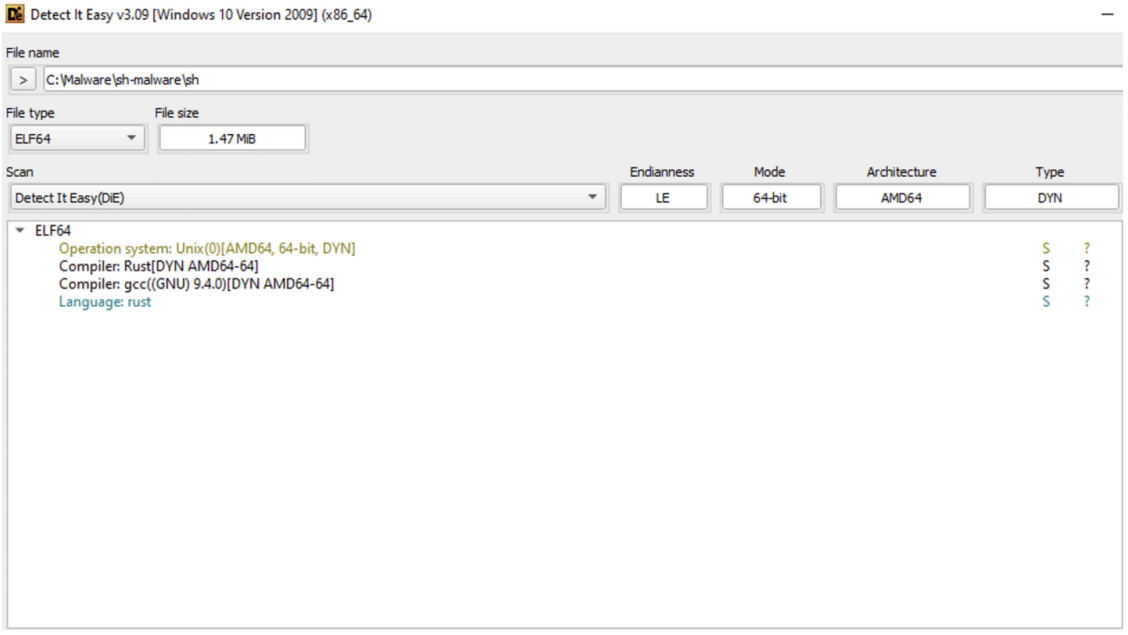
It is also possible that another group of cybercriminals obtained the code to ALPHV and modified it to suit their needs. When BlackCat shut down their operations, they stated that the source code to their ransomware was for sale for $5 million. It is also important to note that, as far as we can tell, the Cicada3301 is not quite as sophisticated as the ALPHV ransomware. The creators may decide to add additional features, such as better obfuscation, later.

Regardless of whether Cicada3301 is a rebrand of ALPHV, they have a ransomware written by the same developer as ALPHV, or they have just copied parts of ALPHV to make their own ransomware, the timeline suggests the demise of BlackCat and the emergence of first the Brutus botnet and then the Cicada3301 ransomware operation may possibly be all connected. More investigation is needed before we can say anything for certain, however.

Technical Details

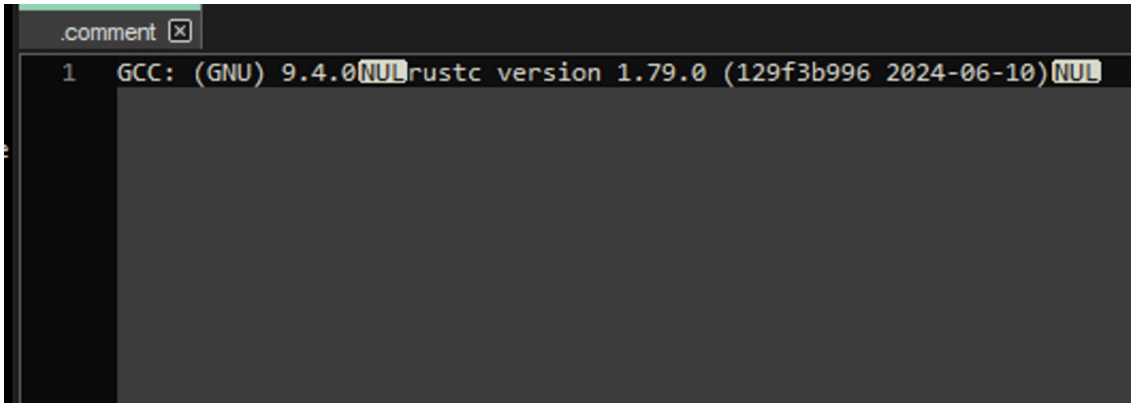
Initial Observations

The ransomware is an ELF binary, and as shown by Detect It Easy, it is compiled and written in Rust.



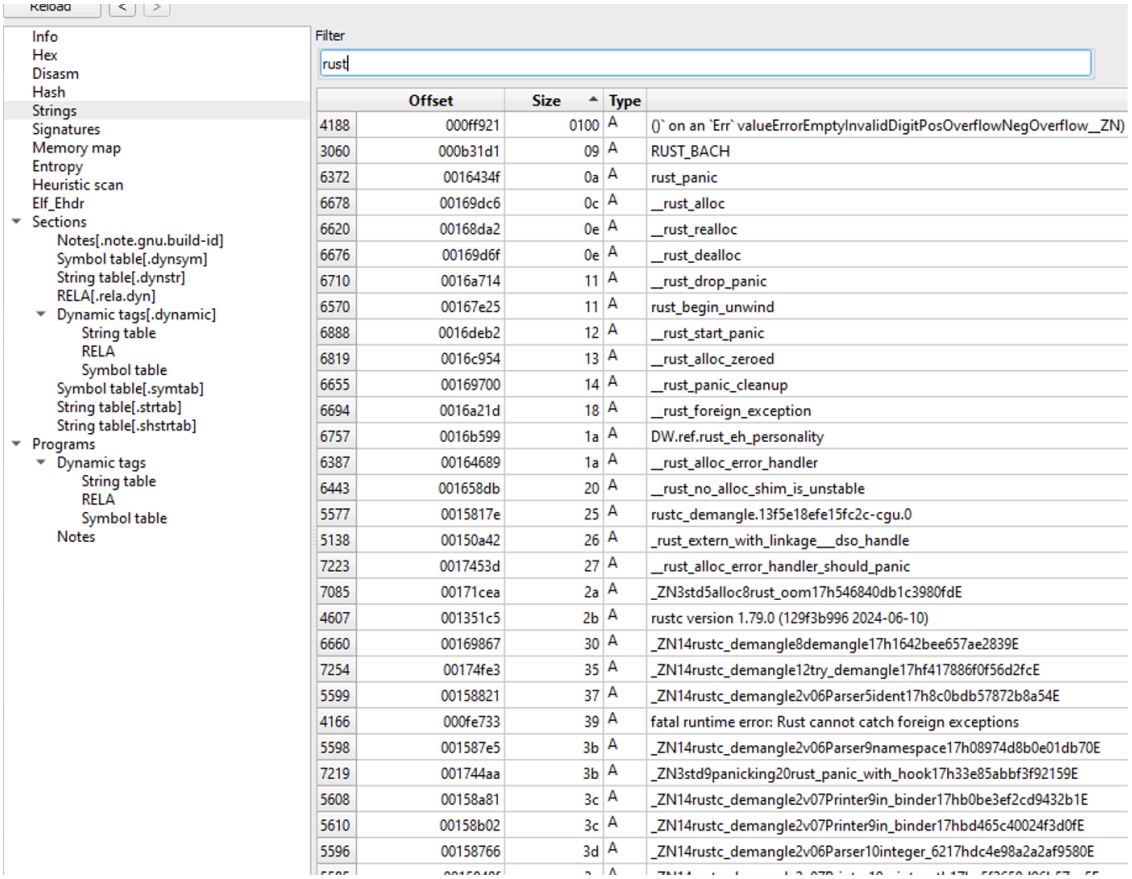
Initial triage of the ransomware

That the ransomware is written in Rust was further strengthened by investigating the .comment section of the binary. There, it was revealed that version 1.79.0 of Rust has been used.



.comment section of the ransomware

Finally, it was further validated that the binary was written in Rust by just looking for strings in the ransomware. With string references to “Rust”, and strings referencing to “Cargo” that is Rust’s build system and package manager, it is concluded that the ransomware is written in Rust.



	Offset	Size	Type	
4188	000ff921	0100	A	()` on an `Err` valueErrorEmptyInvalidDigitPosOverflowNegOverflow__ZN)
3060	000b31d1	09	A	RUST_BACH
6372	0016434f	0a	A	rust_panic
6678	00169dc6	0c	A	__rust_alloc
6620	00168da2	0e	A	__rust_realloc
6676	00169d6f	0e	A	__rust_dealloc
6710	0016a714	11	A	__rust_drop_panic
6570	00167e25	11	A	rust_begin_unwind
6888	0016deb2	12	A	__rust_start_panic
6819	0016c954	13	A	__rust_alloc_zeroed
6655	00169700	14	A	__rust_panic_cleanup
6694	0016a21d	18	A	__rust_foreign_exception
6757	0016b599	1a	A	DW.ref.rust_eh_personality
6387	00164689	1a	A	__rust_alloc_error_handler
6443	001658db	20	A	__rust_no_alloc_shim_is_unstable
5577	0015817e	25	A	rustc_demangle.13f5e18efe15fc2c-cgu.0
5138	00150a42	26	A	_rust_extern_with_linkage___dso_handle
7223	0017453d	27	A	__rust_alloc_error_handler_should_panic
7085	00171cea	2a	A	_ZN3std5alloc8rust_oom17h546840db1c3980fdE
4607	001351c5	2b	A	rustc version 1.79.0 (129f3b996 2024-06-10)
6660	00169867	30	A	_ZN14rustc_demangle8demangle17h1642bee657ae2839E
7254	00174fe3	35	A	_ZN14rustc_demangle12try_demangle17hf417886f0f56d2fcE
5599	00158821	37	A	_ZN14rustc_demangle2v06Parser5ident17h8c0bdb57872b8a54E
4166	000fe733	39	A	fatal runtime error: Rust cannot catch foreign exceptions
5598	001587e5	3b	A	_ZN14rustc_demangle2v06Parser9namespace17h08974d8b0e01db70E
7219	001744aa	3b	A	_ZN3std9panicking20rust_panic_with_hook17h33e85abbf3f92159E
5608	00158a81	3c	A	_ZN14rustc_demangle2v07Printer9in_binder17hb0be3ef2cd9432b1E
5610	00158b02	3c	A	_ZN14rustc_demangle2v07Printer9in_binder17hbd465c40024f3d0fE
5596	00158766	3d	A	_ZN14rustc_demangle2v06Parser10integer_6217hdc4e98a2a2af9580E

Strings related to Rust in the ransomware

Ransomware Functionality

At the start of the ransomware main function, there are several references to parameters that should be passed as an argument to binary, using clap::args, that hold different functionalities that can be used in combination as well.

```
                top the vm and delete snapshots (ESXI)hPrints help informationsleepsINTSleep is ind
                icated in seconds***is_ok***Key is invalid\n(  --bg) &> /dev/null &/bin/nohup/bin/s
                etsidsetsid  --bg &nohup  --bg &> /dev/null &/bin/shCString::new failedsrc/main.rs-
                cStopping vm and deleting snapshots..\n/vmfs/volumes [  ]\n\nAdditional Information
                :\n    --key          Sets the keys for activation (Required parameter)\n-p, --path
                         Sets the path to the file or directory to be encrypted\n    --ui
                 Set graphical user interface\n    --no_vm_ss     Do not stop the vm and delete sna
                pshots (ESXI)\n-s, --sleep        Sleep is indicated in seconds\n\n\n\n TB GB MB KB
                 B -> Encrypted data size: -> Encrypted files:Error sleep: Specify the correct slee
                p parameter\n"
245             ,4);
246     clap::args::arg::Arg::short
247             (&local_558,local_800,
248             "pSets the path to the file or directory to be encrypteduiUISet graphical user inte
                rfacebgBGSystem argsDo not stop the vm and delete snapshots (ESXI)hPrints help info
                rmationsleepsINTSleep is indicated in seconds***is_ok***Key is invalid\n(  --bg) &>
                 /dev/null &/bin/nohup/bin/setsidsetsid  --bg &nohup  --bg &> /dev/null &/bin/shCSt
                ring::new failedsrc/main.rs-cStopping vm and deleting snapshots..\n/vmfs/volumes [
                 ]\n\nAdditional Information:\n    --key          Sets the keys for activation (Req
                uired parameter)\n-p, --path         Sets the path to the file or directory to be e
                ncrypted\n    --ui           Set graphical user interface\n    --no_vm_ss     Do no
                t stop the vm and delete snapshots (ESXI)\n-s, --sleep        Sleep is indicated in
                 seconds\n\n\n\n TB GB MB KB B -> Encrypted data size: -> Encrypted files:Error sle
                ep: Specify the correct sleep parameter\n"
249             ,1);
250     clap::args::arg::Arg::long
251             (local_800,&local_558,
252             "pathFILEhelp\n\n\n\n/keyKEYSets the keys for activationpSets the path to the file
                or directory to be encrypteduiUISet graphical user interfacebgBGSystem argsDo not s
                top the vm and delete snapshots (ESXI)hPrints help informationsleepsINTSleep is ind
                icated in seconds***is_ok***Key is invalid\n(  --bg) &> /dev/null &/bin/nohup/bin/s
                etsidsetsid  --bg &nohup  --bg &> /dev/null &/bin/shCString::new failedsrc/main.rs-
                cStopping vm and deleting snapshots..\n/vmfs/volumes [  ]\n\nAdditional Information
                :\n    --key          Sets the keys for activation (Required parameter)\n-p, --path
                         Sets the path to the file or directory to be encrypted\n    --ui
                 Set graphical user interface\n    --no_vm_ss     Do not stop the vm and delete sna
                pshots (ESXI)\n-s, --sleep        Sleep is indicated in seconds\n\n\n\n TB GB MB KB
```

Arguments passed to the ransomware

The binary has a built-in help function, giving an explanation of the different parameters and how they should be used.

```
remnux@remnux: ~/malware
remnux@remnux:~/malware$ ./sh --help

USAGE:
    sh [FLAGS] [OPTIONS]

Additional Information:
    --key          Sets the keys for activation (Required parameter)
-p, --path         Sets the path to the file or directory to be encrypted
    --ui           Set graphical user interface
    --no_vm_ss     Do not stop the vm and delete snapshots (ESXI)
-s, --sleep        Sleep is indicated in seconds
```

Help function of the ransomware

The main function of the binary, which is done by the malware developer, is called linux_enc. By searching for linux_enc function a general program flow of the binary could be found.

Search Text - "ddw linux_enc" [Listing Display Match] - (sh) (13 entries)

Location	Label	Name...	Preview
00217b6c			ddw linux_enc::format_bytes
00217fa4			ddw linux_enc::main
00218444			ddw linux_enc::collect_files_except
00218564			ddw linux_enc::check_file
002185a0			ddw linux_enc::get_args
0021895c			ddw linux_enc::encryption::is_extension_full
002189ac			ddw linux_enc::encryption::create_file_recovery
00218a18			ddw linux_enc::encryption::encrypt_file
00218cd0			ddw linux_enc::encryption::ecnrypted_files_full
00218d24			ddw linux_enc::encryption::ecnrypted_files_block
00218d78			ddw linux_enc::encryption::check_key_and_get_rec_text
00218f50			ddw linux_enc::esxi::get_path_exe
00218fa4			ddw linux_enc::esxi::stop_vm_removeall_snapshot

The function calls of main

The Ransomware Parameters

It is possible to add a sleep parameter of the binary, adding a delay in seconds when the ransomware should be executed. For the sleep function, the ransomware uses the built-in sleep function std::thread::sleep

```
                    LAB_001170d4
001170d4    LEA                         RDI=>local_b80, [RSP + 0x100]
001170dc    MOV                         RSI, R14
001170df    MOV                         RDX, RBX
001170e2    CALL                        qword ptr [->u64>::from_str]
001170e8    CMP                         byte ptr [RSP + local_b80[0]], 0x0
001170f0    JNZ                         LAB_00118b6e
001170f6    MOV                         RDI, qword ptr [RSP + local_b80[8]]
001170fe    XOR                         ESI, ESI
00117100    CALL                        qword ptr [->std::thread::sleep]
```

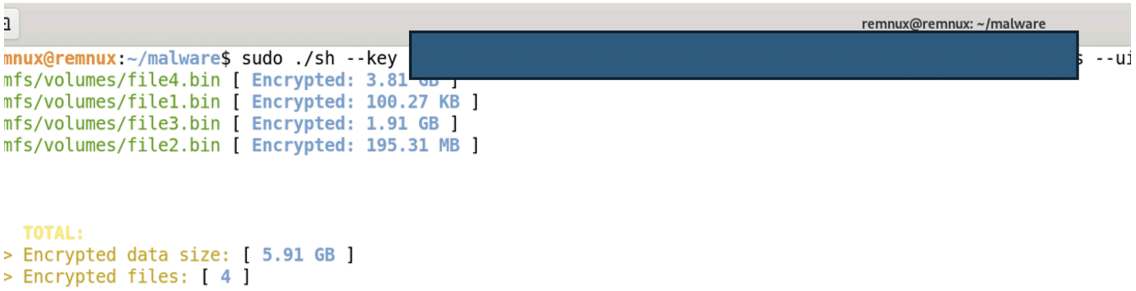
The sleep parameter of the ransomware

The ui parameter prints the result of the encryption to the screen, showing what files have been encrypted and a statistic of the total amount of files and data that has been successfully encrypted.

```
0117c1f     JZ                          LAB_001189bf
0117c25     MOV                         RCX, "etpyrcnE"
0117c2f     MOV                         qword ptr [RAX], RCX
0117c32     MOV                         word ptr [RAX + 0x8], ":d"
0117c38     MOV                         qword ptr [RSP + local_b80[0]], 0xa
0117c44     MOV                         qword ptr [RSP + local_b80[8]], RAX
0117c4c     MOV                         qword ptr [RSP + local_b70], 0xa
0117c58     MOV                         RAX, 0x1100000004
0117c62     MOV                         qword ptr [RSP + local_b68[0]], RAX
0117c6a     MOV                         byte ptr [RSP + local_b68[8]], 0x0

                    try { // try from 00117c72 to 00117c87 has its CatchHandler @ 001191fe
                    LAB_00117c72
0117c72     LEA                         RDI=>local_2f8, [RSP + 0x988]
0117c7a     LEA                         RSI=>local_b80, [RSP + 0x100]
0117c82     CALL                        qword ptr [-><colored::ColoredString_as_colored::Colorize>::bo
```

The ui parameter of the ransomware

The ui parameter was confirmed by running the ransomware and using the ui flag, showing the progress and statistics on the command prompt.



The ui parameter output

If the parameter no_vm_ss is chosen, the ransomware will encrypt files without shutting down the virtual machines that are running on ESXi. This is done by using the built-in esxicli terminal that will also delete snapshots.

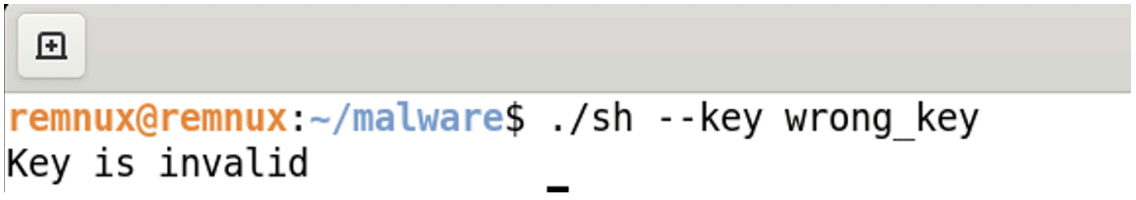
Built-in esxicli commands of the ransomware

The full commands that the ransomware is utilizing are the following.

esxcli –formatter=csv –format-param=fields==\"WorldID,DisplayName\" vm process list | grep -viE \",(),\" | awk -F \"\\\",\\\"*\" \'{system(\"esxcli vm process kill –type=force –world-id=\"$1)}\' > /dev/null 2>&1;*

for i in `vim-cmd vmsvc/getallvms| awk \'{print$1}\'`;do vim-cmd vmsvc/snapshot.removeall $i & done > /dev/null 2>&1

The most important parameter is the one named key. This needs to be provided, otherwise the binary will fail and show on the screen “Key is invalid”.



Output if wrong key is passed to the ransomware

The binary has a function called check_key_and_get_rec_text. It will make a check to see if the provided key is of length 0x2C to enter the function, but the size is also provided as an argument to the function. If the length is less than 0x2C the binary will terminate directly.

```
)V                              qword ptr [RSP + local_c58], RBX
:ST                             RAX, RAX
:                               LAB_00115d80
fP                              RDX, 0x2c                                   ; Check lenght of key, needs to be 45 cha
IZ                              LAB_00115d80

    try { // try from 00115cdd to 00115cf1 has its CatchHandler @ 00118fc3
    LAB_00115cdd                                                                XREF[1]:            0021
:A                              RDI=>local_558, [RSP + 0x728]
)V                              EDX, 0x2c                                   ; Size of key (0x2c)
)V                              RSI, RAX
ALL                             linux_enc::encryption::check_key_and_get_rec_text   ; undefined check_key_and_get_rec_text(un
                                                                            ; undefined8 param_2, undefined8 param_3)
```

Checking correct key length

If the size of the key is correct, the ransomware will enter the function check_key_and_get_rec_text. One of the first things that happen in the function is to load an encrypted base64 encoded data blob that is stored in the data section. The decoded data is then stored and will be used later in the function.

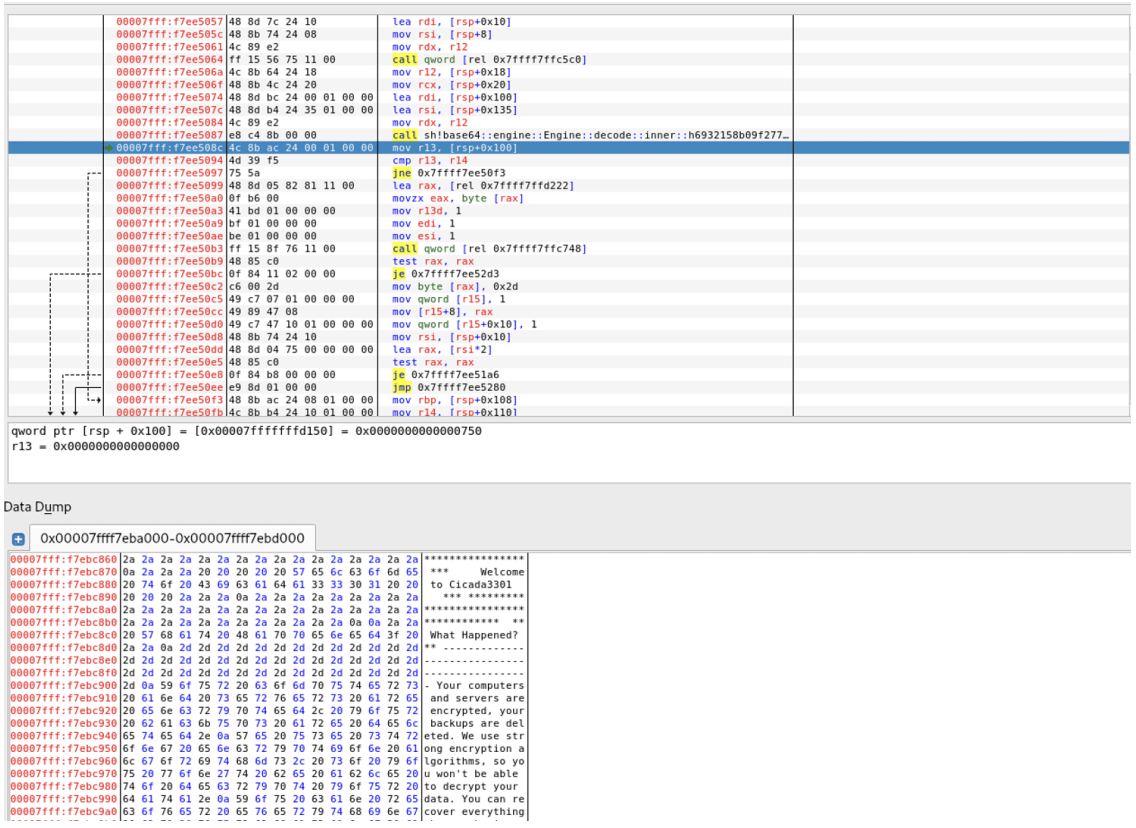
```
                R14, -0x8000000000000000
                RSI, [s__001f3528+1296]                         ; = 01h
                RBX=>local_173, [RSP + 0x135]
                EDX, 0x143
                RDI, RBX
                qword ptr [->memcpy]                            ; void * memcpy(void * __dest, void * __src, size_
                RDX, [s_5NCoPBAaqDlK6gt0DLDuJLoPNweGPSdr_001f3b7b]  ; =
                                                                ; "5NCoPBAaqDlK6gt0DLDuJLoPNweGPSdrpuOOIX1LoGaoxyJ
                                                                ; Eb4GIPXPpyquK35sC6fC18x0PWByhABiVTeo0e7CNOCX772z
                                                                ; ypjPjDHotIRLq8dH5kpz0WaihbNx+UmQ1LTy4h3l513ixseD
                                                                ; xnLJdo/TB3o3dz1rzgGiPsMfnxP1MbxaAtTFiK4okramHWOl
                                                                ; yXleT2007YOe82uAwNhP05OzkxHDAq+eD+JrHQEfhf+zK5ih
                RDI=>local_190, [RSP + 0x118]
                ECX, 0xd00
                RSI, RBX
                base64::engine::Engine::decode::inner           ; undefined inner()
                RBX, qword ptr [RSP + local_190]
```

Encoded and encrypted ransomware note inside the ransomware

The provided parameter key is then taken as a key to decrypt, using ChaCha20, the encoded data blob. If the provided key is correct the message that is shown in the ransomware note will be decrypted.

```
CALL                            base64::engine::Engine::decode::inner       ; undefined inner()
MOV                             RBX, qword ptr [RSP + local_190]
CMP                             RBX, R14
JNZ                             LAB_0011dbf1
LEA                             RAX, [__rust_no_alloc_shim_is_unstable]     ; = ??
MOVZX                           EAX=>__rust_no_alloc_shim_is_unstable, byte ptr [RAX]   ; = ??
MOV                             EDI, 0x1
MOV                             ESI, 0x1
CALL                            qword ptr [->__rust_alloc]                  ; undefined __rust_alloc()
TEST                            RAX, RAX
JZ                              LAB_0011e29c
MOV                             byte ptr [RAX], 0x2d
MOV                             qword ptr [R15], 0x1
MOV                             qword ptr [R15 + 0x8], RAX
MOV                             qword ptr [R15 + 0x10], 0x1
JMP                             LAB_0011e1be

    LAB_0011dbf1                                                            XREF[1]:            0011
MOV                             RAX, qword ptr [RSP + local_188]
MOV                             qword ptr [RSP + local_2a0], RAX
CMP                             R12, 0x2c
JNZ                             LAB_0011dc9a
MOV                             R12, qword ptr [RSP + local_180]
MOVUPS                          XMM0, xmmword ptr [CRYPT_ChaChaInitStates_expanded32k]  ; = "expand 32-byte k"
MOVAPS                          xmmword ptr [RSP + local_278[0]], XMM0
```

Decryption of the ransomware note



Decrypted ransomware note

To verify that the provided key was correct after exiting the check_key_and_get_rec_text function, there is a check that the ransomware note has been decrypted properly.

```
349           encryption::check_key_and_get_rec_text(&local_558,SUB168(auVar49,0),0x2c);
350           ppppuVar30 = local_548;
351           pppppuVar26 = pcStack1360;
352           if (local_558 != 0x8000000000000000) {
353             local_b90 = 0x8000000000000000;
354             ppppppuVar33 = local_558;
355             if (local_548 < &DAT_0000000c) {
356               if ((local_548 != &DAT_0000000b) ||
357                  ((*(pcStack1360 + 3) ^ L'\x6f5f7369' | *pcStack1360 ^ L'\x692a2a2a') != 0)) {
358 LAB_00115e9d:
359                 local_b80 = &PTR_s_Key_is_invalid_(_--bg)_&>_/dev/n_0022d520;
360                 pcStack2936 = 0x1;
361                 local_b70 = &DAT_00000008;
362                 _local_b68 = 0x0;
363                     /* try { // try from 00115ed1 to 00115ede has its CatchHandler @ 00118f39 */
364                 std::io::stdio::_print(0,&local_b80);
365                 if (pppppuVar33 != 0x0) {
366                   __rust_dealloc(pppppuVar26,ppppppuVar33,1);
367                 }
368                 goto LAB_00115dc2;
369               }
370             }
```

Validation that the ransomware note has been decrypted

File Encryption

The functions start by using OsRng to generate entropy for the symmetric key. OsRng is a random number generator that retrieves randomness from the operating system.

```
                try { // try from 0011af37 to 0011af65 has its CatchHandler @ 0011c9af
                LAB_0011af37                                                    XREF[1]:
)11af37     MOV                         R12, RAX
)11af3a     LEA                         RDI=>local_308, [RSP + 0x130]
)11af42     MOV                         EDX, 0x20
)11af47     MOV                         RSI, R15
)11af4a     CALL                        qword ptr [-><rand_core::os::OsRng_as_rand_core::RngCore>::fill_bytes]
)11af50     LEA                         RDI=>local_308, [RSP + 0x130]
)11af58     MOV                         EDX, 0xc
)11af5d     MOV                         RSI, R12
)11af60     CALL                        qword ptr [-><rand_core::os::OsRng_as_rand_core::RngCore>::fill_bytes]
            } // end try from 0011af37 to 0011af65
)11af66     MOV                         qword ptr [RSP + local_3c0], 0x20
)11af6f     MOV                         qword ptr [RSP + local_3b8], R15
)11af77     MOV                         qword ptr [RSP + local_3b0], 0x20
```

Function used to generate keys to ChaCha20

The binary contains a function called encrypt_file that handles the encryption of the files. The first function is to extract another public pgp key that is stored in the data section. This key is used for encryption to encrypt the symmetric key that is generated for file encryption.

```
            try { // try from 0011af83 to 0011af9b has its CatchHandler @ 0011c992
            LAB_0011af83                                                    XREF[1]:    002111b0(*)
f83     LEA                         RSI, [s_-----BEGIN_PUBLIC_KEY-----_MIIBI_001f379a]  ; = "-----BEGIN PUBLIC
                                                                            ; KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAzoCjgss/6hs
                                                                            ; GYO1d\nkz88a82BZwKB2GlahI+FzvXpjCXtM4ky/eMDfXeWJFXbo0pXykLeeJXfdX
                                                                            ; gm\n7qyMu1X3NgPKZ3CHZlGqZpN0WaTurgFbJ7152wKR491EY1WmyxcGw2b0RvsjN
                                                                            ; nCcSURC2iH/2pTzmUDgKCtkvy+IqAVJ9VaCWjI177iW9koNf0EYAbhfCMLX/Hbwxc
                                                                            ; Fznuk6oPoc6GsQ92ON05CSv+hk1ryt7/3+1sIZpKIDPDuWYV3E/+AyzHie+ovW\nq
f8a     LEA                         RDI=>local_308, [RSP + 0x130]
f92     MOV                         EDX, 0x1c3
f97     CALL                        spki::traits::DecodePublicKey::from_public_key_pem  ; undefined from_public_key_pem()
            } // end try from 0011af83 to 0011af9b
f9c     MOV                         RAX, qword ptr [RSP + local_308[0]]
fa4     MOVUPS                      XMM0, xmmword ptr [RSP + local_308[8]]
fac     MOVAPS                      xmmword ptr [RSP + local_c8[0]], XMM0
fb4     MOVUPS                      XMM0, xmmword ptr [RSP + local_2f0[0]]
fbc     MOVAPS                      xmmword ptr [RSP + local_b8[0]], XMM0
fc4     MOVUPS                      XMM0, xmmword ptr [RSP + local_2e0[0]]
fcc     MOVAPS                      xmmword ptr [RSP + local_a8[0]], XMM0
fd4     MOV                         RCX, qword ptr [RSP + local_2d0]
fdc     MOV                         qword ptr [RSP + local_98], RCX
```

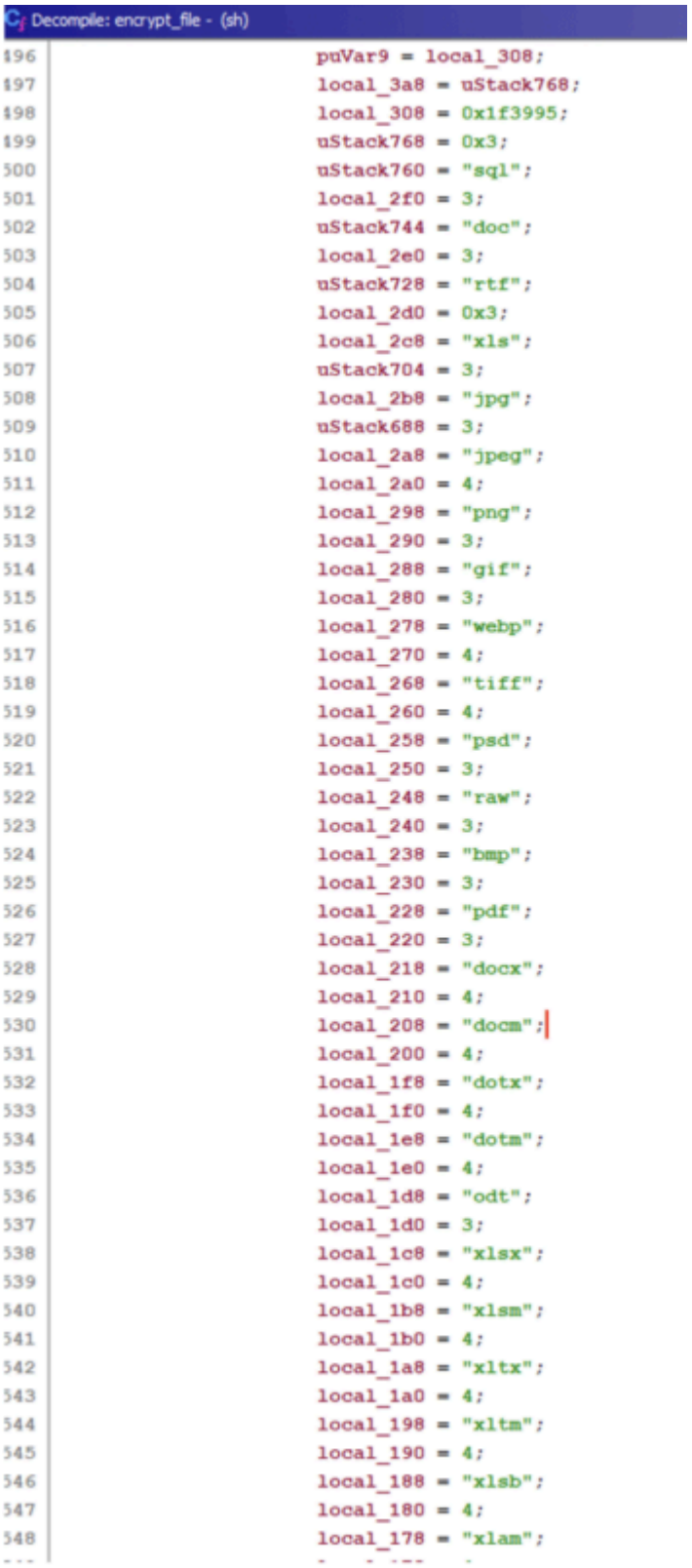
RSA key used for key encryption

It then creates the file that will store the ransomware message in the folder of the encrypted files. It will be named “RECOVER-’ending of encrypted file’-DATA.txt”

```
0011b13e    JZ                          LAB_0011c572
0011b144    MOV                         RCX, "-REVOCER"
0011b14e    MOV                         qword ptr [RAX], RCX
0011b151    MOV                         qword ptr [RSP + local_308[0]], 0x8
0011b15d    MOV                         qword ptr [RSP + local_308[8]], RAX
0011b165    MOV                         qword ptr [RSP + local_308[16]], 0x8

            try { // try from 0011b171 to 0011b187 has its CatchHandler @ 0011c968
            LAB_0011b171                                                    XREF[1]:    002111bc(*)
0011b171    LEA                         RDI=>local_308, [RSP + 0x130]
0011b179    MOV                         ESI, 0x8
0011b17e    MOV                         EDX, 0x7
0011b183    CALL                        alloc::raw_vec::RawVec<T,A>::reserve::do_reserve_and_handle  ; undefined do_reserve_and_handle()
            } // end try from 0011b171 to 0011b187
0011b188    MOV                         RAX, qword ptr [RSP + local_308[8]]
0011b190    MOV                         RSI, qword ptr [RSP + local_308[16]]
0011b198    MOV                         dword ptr [RAX + RSI*0x1 + 0x3], "m300"
0011b1a0    MOV                         dword ptr [RAX + RSI*0x1], "0khb"
0011b1a7    ADD                         RSI, 0x7
0011b1ab    MOV                         qword ptr [RSP + local_3f8[0]], RSI
0011b1b0    MOVUPS                      XMM0, xmmword ptr [RSP + local_308[0]]
0011b1b8    MOVAPS                      xmmword ptr [RSP + local_408[0]], XMM0
0011b1bd    MOV                         RBX, qword ptr [RSP + local_408[0]]
0011b1c2    MOV                         RAX, RBX
0011b1c5    SUB                         RAX, RSI
0011b1c8    CMP                         RAX, 0x8
0011b1cc    JA                          LAB_0011b1e7

            try { // try from 0011b1ce to 0011b1dc has its CatchHandler @ 0011c938
            LAB_0011b1ce                                                    XREF[1]:    002111c2(*)
0011b1ce    LEA                         RDI=>local_408, [RSP + 0x30]
0011b1d3    MOV                         EDX, 0x9
0011b1d8    CALL                        alloc::raw_vec::RawVec<T,A>::reserve::do_reserve_and_handle  ; undefined do_reserve_and_handle()
            } // end try from 0011b1ce to 0011b1dc
0011b1dd    MOV                         RBX, qword ptr [RSP + local_408[0]]
0011b1e2    MOV                         RSI, qword ptr [RSP + local_3f8[0]]

            LAB_0011b1e7                                                    XREF[1]:    0011b1cc(j)
0011b1e7    MOV                         R14, qword ptr [RSP + local_408[8]]
0011b1ec    MOV                         RAX, "xt.ATAD-"
0011b1f6    MOV                         qword ptr [R14 + RSI*0x1], RAX
0011b1fa    MOV                         byte ptr [R14 + RSI*0x1 + 0x8], 0x74
0011b200    ADD                         RSI, '\t'
```

Creating the ransomware note

Inside the encryption function there is a list of file extensions where most of them are related to either documents or pictures. This indicates that the ransomware has been used to encrypt Windows systems before being ported to ransomware ESXi hosts.



File extensions for encryption (not used)

The full list of extensions:

jpeg, webp, tiff, docx, docm, dotx, dotm, xlsx, xlsm, xltx, xltm, xlsb, xlam, pptx, pptm, ptox, potm, ppsx, ppsm, sql, doc, rtf, xls, jpg, png, gif, psd, raw, bmp, pdf, odt, ods, odp, mdf, txt

Then it checks the size of the file. If it is greater than 0x6400000, then it will encrypt the file in parts, and if it is smaller, the whole file will be encrypted.

```
575                     if (local_418 < 0x6400000) {
576                       puVar13 = puVar7;
577                     }
578                     if (cVar1 != '\0') {
579                       puVar6 = local_3a8;
580                       puVar13 = puVar10;
581                     }
582                     if (puVar13 == 0x1) {
583                       bVar2 = *puVar6;
584                       bVar11 = bVar2 + 0xbf;
585                       if (((bVar11 < 0x1a) << 5 | bVar2) == 0x30) {
586                         auVar15 = ecnrypted_files_full
587                                             (&local_380,local_328,local_3b8,local_3b0,lVar4,0xc);
588                         if (SUB168(auVar15,0) == 0) {
589                           bVar2 = *puVar6;
590                           bVar11 = bVar2 + 0xbf;
591                           goto LAB_0011bdaa;
592                         }
593                       }
594                       else {
595 LAB_0011bdaa:
596                         if (((bVar11 < 0x1a) << 5 | bVar2) == 0x31) {
597                           auVar15 = ecnrypted_files_block
598                                               (&local_380,local_328,local_3b8,local_3b0,lVar4,0xc,
599                                                local_418,local_418 / 0xf,0x100000,1);
600                           if (SUB168(auVar15,0) != 0) goto LAB_0011bf05;
601                           bVar2 = *puVar6;
602                           bVar11 = bVar2 + 0xbf;
603                         }
604                         if (((bVar11 < 0x1a) << 5 | bVar2) == 0x32) {
605                           auVar15 = ecnrypted_files_block
606                                               (&local_380,local_328,local_3b8,local_3b0,lVar4,0xc,
607                                                local_418,local_418 / 0x1e,0x100000,1);
608                           if (SUB168(auVar15,0) != 0) goto LAB_0011bf05;
609                           bVar2 = *puVar6;
610                           bVar11 = bVar2 + 0xbf;
611                         }
612                         if ((((bVar11 < 0x1a) << 5 | bVar2) != 0x33) ||
613                            (auVar15 = ecnrypted_files_block
```

Checking file size for encryption

The files will then be encrypted with a symmetric key generated by OsRng using ChaCha20.

```
                    LAB_0011cc0d
0011cc0d    CMP         byte ptr [RSP + local_161], 0x0
0011cc12    JNZ         LAB_0011d084
0011cc18    MOVUPS      XMM0, xmmword ptr [CRYPT_ChaChaInitStates_expanded32k]
0011cc1f    MOVAPS      xmmword ptr [RSP + local_128[0]], XMM0
0011cc24    MOV         RAX, qword ptr [RSP + local_e8]
0011cc2c    MOVDQU      XMM0, xmmword ptr [RAX]
0011cc30    MOVDQU      XMM1, xmmword ptr [RAX + 0x10]
0011cc35    LEA         RAX=>local_118, [RSP + 0x50]
0011cc3a    MOVDQU      xmmword ptr [RAX + local_108[0]], XMM1
0011cc3f    MOVDQU      xmmword ptr [RAX]=>local_118, XMM0
0011cc43    MOV         RCX, qword ptr [RSP + local_e0]
0011cc4b    MOV         RAX, qword ptr [RCX]
0011cc4e    MOV         qword ptr [RSP + local_138], RAX
0011cc53    MOV         EAX, dword ptr [RCX + 0x8]
0011cc56    MOV         dword ptr [RSP + local_130], EAX
0011cc5a    LEA         RAX, [chacha20::avx2_cpuid::STORAGE]
0011cc61    MOVZX       EAX=>chacha20::avx2_cpuid::STORAGE, byte ptr [RAX]
0011cc64    CMP         AL, 0xff
0011cc66    JNZ         LAB_0011ccd5
```

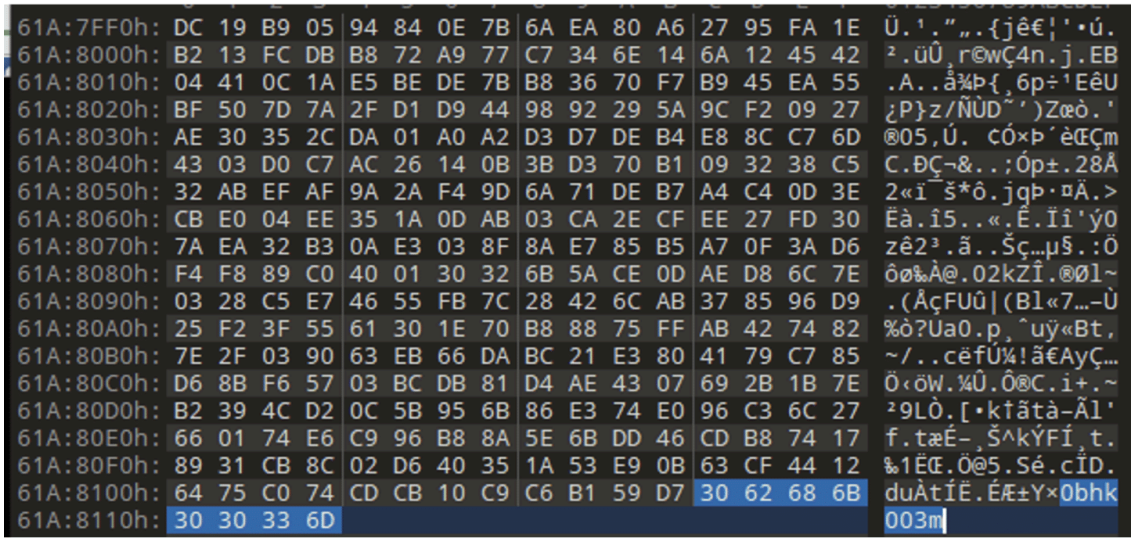
Use of ChaCha20 for file encryption

After the encryption is done, the ransomware encrypts the ChaCha20 key with the provided RSA key and finally writes the extension to the encrypted file.

```
/* try { // try from 0011c0a0 to 0011c0db has its CatchHandler @ 0011c757 */
        rsa::padding::PaddingScheme::new_oaep(&local_408);
        <rsa::key::RsaPublicKey_as_rsa::key::PublicKey>::encrypt
                  (&local_308,&local_90,&local_308,&local_408,local_3b8,
                   local_3b0);
        uVar12 = local_2f0;
        puVar10 = uStack768;
        puVar7 = local_308;
        file_size_chunking = uStack760;
        if (local_308 == &DAT_00000012) {
/* try { // try from 0011c10d to 0011c12b has its CatchHandler @ 0011c717 */
          <alloc::vec::Vec<Z>as_zeroize::Zeroize>::zeroize(&local_3c0);
          puVar7 = <std::io::buffered::bufwriter::BufWriter<W>as_std::io::Write>
                   ::write_all(&local_350,file_size_chunking,uVar12);
          if (puVar7 == 0x0) {
/* try { // try from 0011c1c7 to 0011c1db has its CatchHandler @ 0011c717 */
            puVar7 = <std::io::buffered::bufwriter::BufWriter<W>as_std::io::Writ
            e>
                     ::write_all(&local_350,lVar4,0xc);
            if (puVar7 == 0x0) {
/* try { // try from 0011c1fb to 0011c20d has its CatchHandler @ 0011c717 */
              puVar7 = <std::io::buffered::bufwriter::BufWriter<W>as_std::io::Wr
              ite>
                       ::write_all(&local_350,puVar6,puVar13);
              if (puVar7 == 0x0) {
/* try { // try from 0011c22a to 0011c242 has its CatchHandler @ 0011c717 */
                puVar7 = <std::io::buffered::bufwriter::BufWriter<W>as_std::io::
                Write>
                         ::write_all(&local_350,"bhk003m",7);
```

Adding the encryption file extension

The file extension is also added to the end of the encrypted file, together with the RSA-encrypted ChaCha20 key.



61A:7FF0h:	DC	19	B9	05	94	84	0E	7B	6A	EA	80	A6	27	95	FA	1E	Ü.¹.”„.{jê€¦'•ú.
61A:8000h:	B2	13	FC	DB	B8	72	A9	77	C7	34	6E	14	6A	12	45	42	².üÛ¸r©wÇ4n.j.EB
61A:8010h:	04	41	0C	1A	E5	BE	DE	7B	B8	36	70	F7	B9	45	EA	55	.A..å¾Þ{¸6p÷¹EêU
61A:8020h:	BF	50	7D	7A	2F	D1	D9	44	98	92	29	5A	9C	F2	09	27	¿P}z/ÑÙD˜')Zœò.'
61A:8030h:	AE	30	35	2C	DA	01	A0	A2	D3	D7	DE	B4	E8	8C	C7	6D	®05,Ú. ¢Ó×Þ´èŒÇm
61A:8040h:	43	03	D0	C7	AC	26	14	0B	3B	D3	70	B1	09	32	38	C5	C.ÐÇ¬&..;Óp±.28Å
61A:8050h:	32	AB	EF	AF	9A	2A	F4	9D	6A	71	DE	B7	A4	C4	0D	3E	2«ï¯š*ô.jqÞ·¤Ä.>
61A:8060h:	CB	E0	04	EE	35	1A	0D	AB	03	CA	2E	CF	EE	27	FD	30	Ëà.î5..«.Ê.Ïî'ý0
61A:8070h:	7A	EA	32	B3	0A	E3	03	8F	8A	E7	85	B5	A7	0F	3A	D6	zê2³.ã..Šç…µ§.:Ö
61A:8080h:	F4	F8	89	C0	40	01	30	32	6B	5A	CE	0D	AE	D8	6C	7E	ô ø‰À@.02kZÎ.®Øl~
61A:8090h:	03	28	C5	E7	46	55	FB	7C	28	42	6C	AB	37	85	96	D9	.(ÅçFUû|(Bl«7…–Ù
61A:80A0h:	25	F2	3F	55	61	30	1E	70	B8	88	75	FF	AB	42	74	82	%ò?Ua0.p¸ˆuÿ«Bt‚
61A:80B0h:	7E	2F	03	90	63	EB	66	DA	BC	21	E3	80	41	79	C7	85	~/..cëfÚ¼!ã€AyÇ…
61A:80C0h:	D6	8B	F6	57	03	BC	DB	81	D4	AE	43	07	69	2B	1B	7E	Ö‹öW.¼Û.Ô®C.i+.~
61A:80D0h:	B2	39	4C	D2	0C	5B	95	6B	86	E3	74	E0	96	C3	6C	27	²9LÒ.[•k†ãtà–Ãl'
61A:80E0h:	66	01	74	E6	C9	96	B8	8A	5E	6B	DB	46	CD	B8	74	17	f.tæÉ–¸Š^kÛFÍ¸t.
61A:80F0h:	89	31	CB	8C	02	D6	40	35	1A	53	E9	0B	63	CF	44	12	‰1ËŒ.Ö@5.Sé.cÏD.
61A:8100h:	64	75	C0	74	CD	CB	10	C9	C6	B1	59	D7	30	62	68	6B	duÀtÍË.ÉÆ±Y×0bhk
61A:8110h:	30	30	33	6D													003m

File extension at the end of the file

YARA Rule for Cicada3301 Threat Hunting

```
rule elf_cicada3301{

      meta:
             author = "Nicklas Keijser"
             description = "Detect ESXi ransomware by the group Cicada3301"
             date = "2024-08-31"

      strings:
             $x1 = "no_vm_ss" nocase wide ascii
             $x2 = "linux_enc" nocase wide ascii
             $x3 = "nohup" nocase wide ascii
             $x4 = "snapshot.removeall" nocase wide ascii
             $x5 = {65 78 70 61 6E 64 20 33 32 2D 62 79 74 65 20 6B} //Use of ChaCha20 constant expand 32-byte k

      condition:
             uint16(0) == 0x457F
             and filesize < 10000KB
             and (all of ($x*))
}
```

---

[1] https://www.forescout.com/resources/analysis-of-an-alphv-incident
[2] https://blogs.vmware.com/security/2022/09/esxi-targeting-ransomware-the-threats-that-are-after-your-virtual-machines-part-1.html
[3] https://annoyed.engineer/2024/03/23/the-brutus-botnet/
[4] https://www.bleepingcomputer.com/news/security/cisco-warns-of-large-scale-brute-force-attacks-against-vpn-services/
[5] https://krebsonsecurity.com/2022/01/who-wrote-the-alphv-blackcat-ransomware-strain/
