S*(un)*SZ

Samsung (un) Secret Zone

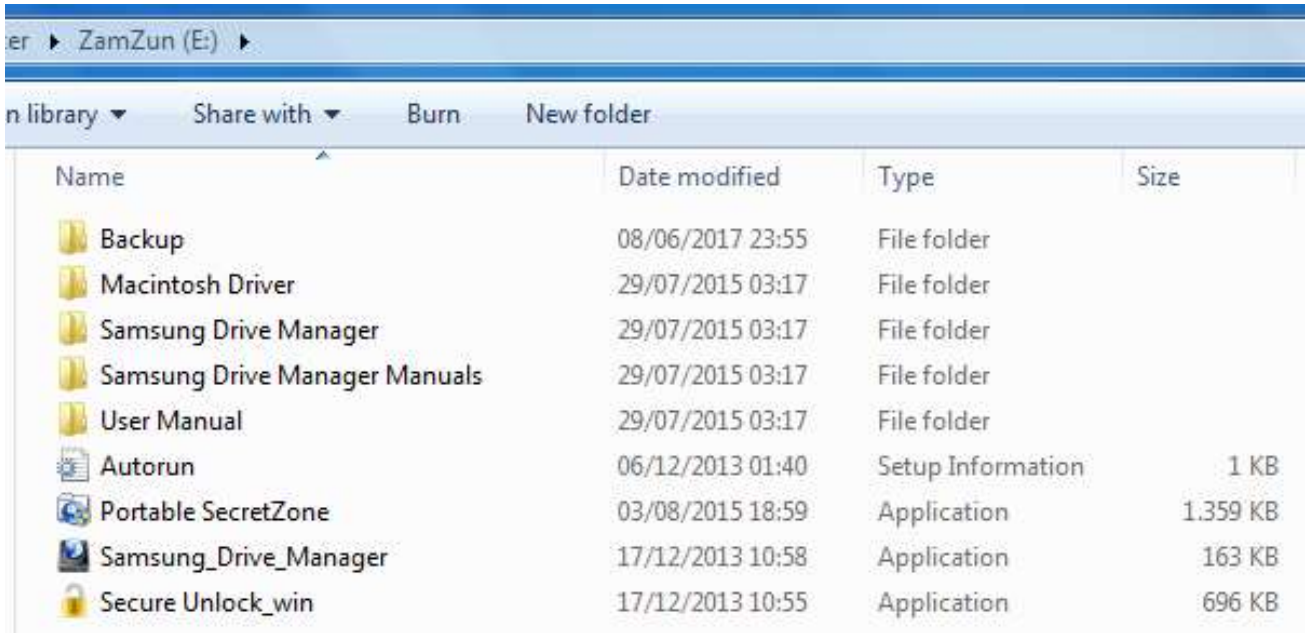
what is it?

SSZ (Samsung, now Seagate, Secret Zone is *"a software program that protects personal information by creating a secure, password-protected folder on the Samsung external drive."* [http://www.seagate.com/gb/en/support/downloads/item/samsung-secretzone-master-dl/]

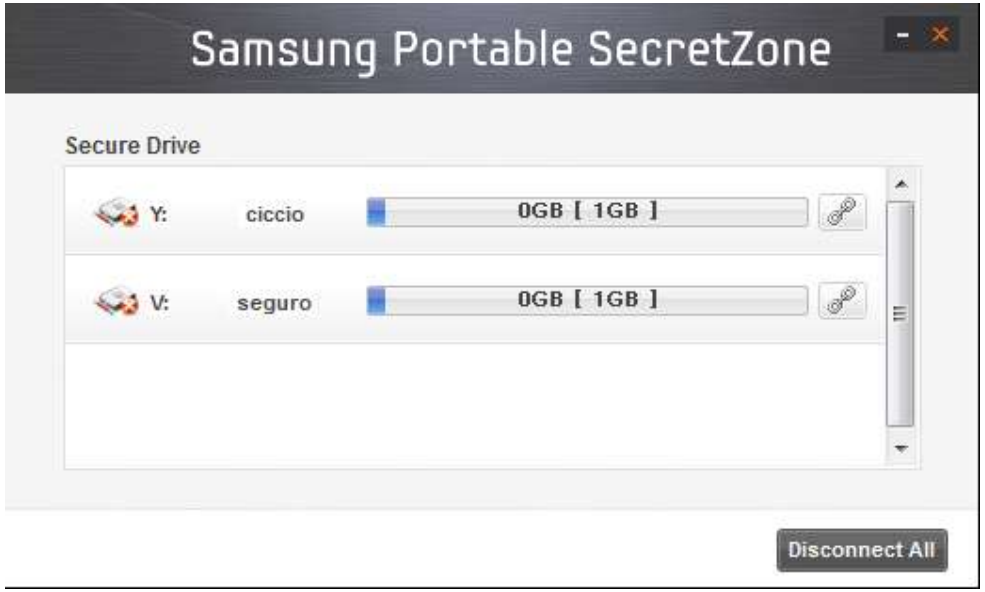
- A **software**...
- **Secure**... (*well, you can't simply say that*)
- Folder? *Ancient* versions, maybe...

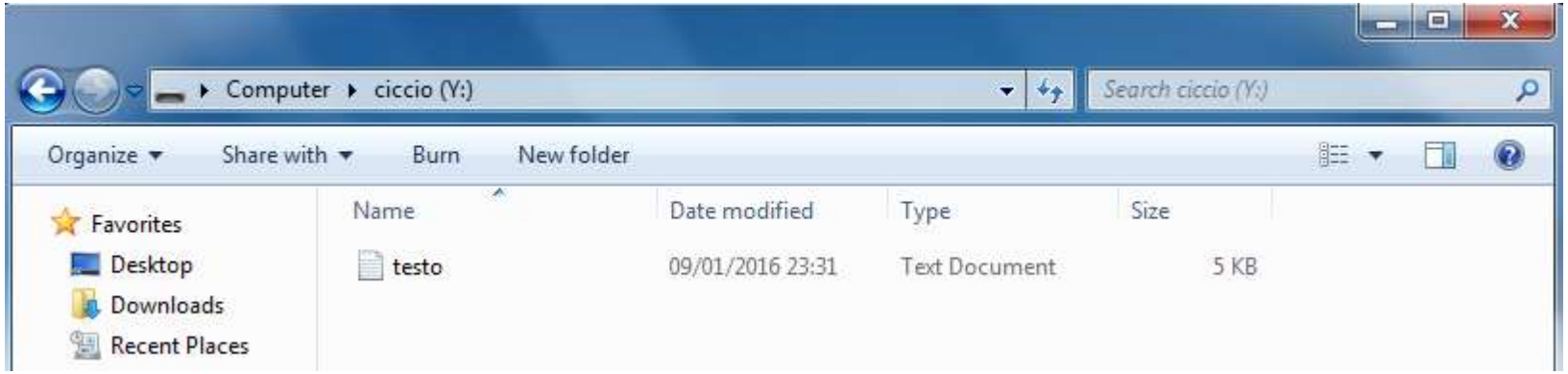
the software program

- The software if provided as *free* tool with Samsung/Seagate/Maxtor portable devices (e.g. the M3 USB disk)



executing *Portable SecretZone*

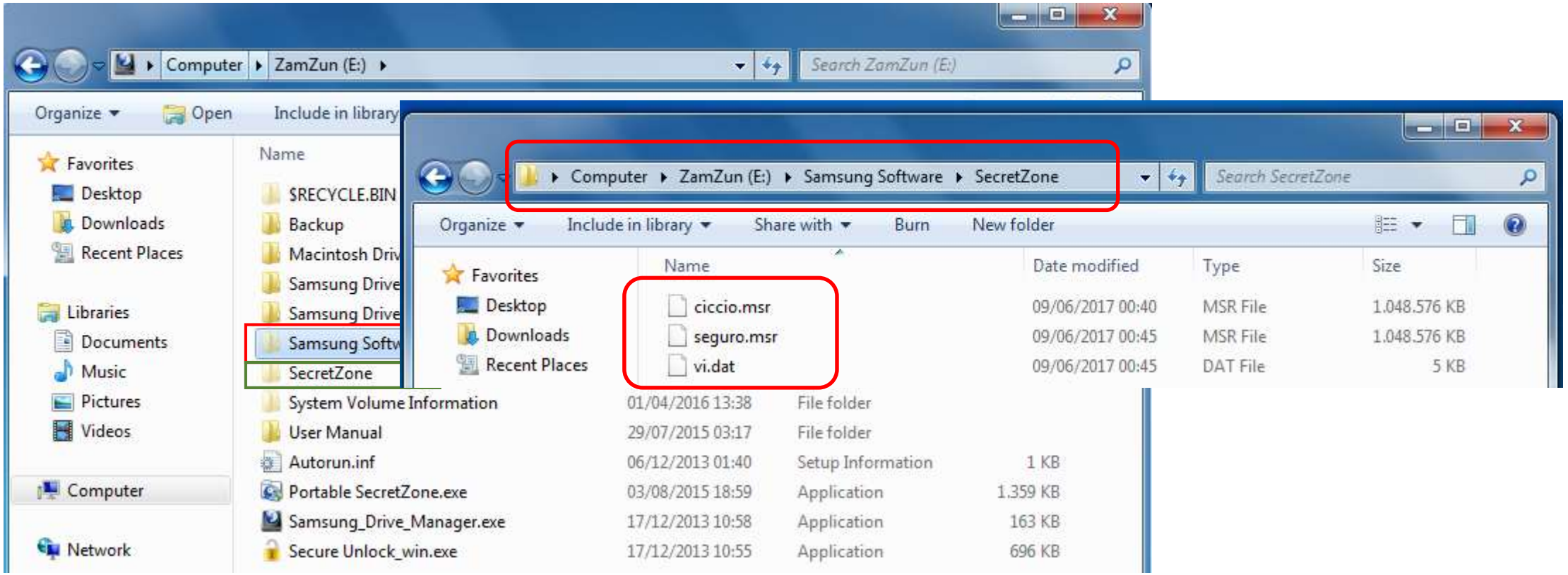




popcorns incoming

where the *secured* data is?

Please show me the *hidden* and the *system* files!



msr files and **vi.dat**

vi.dat

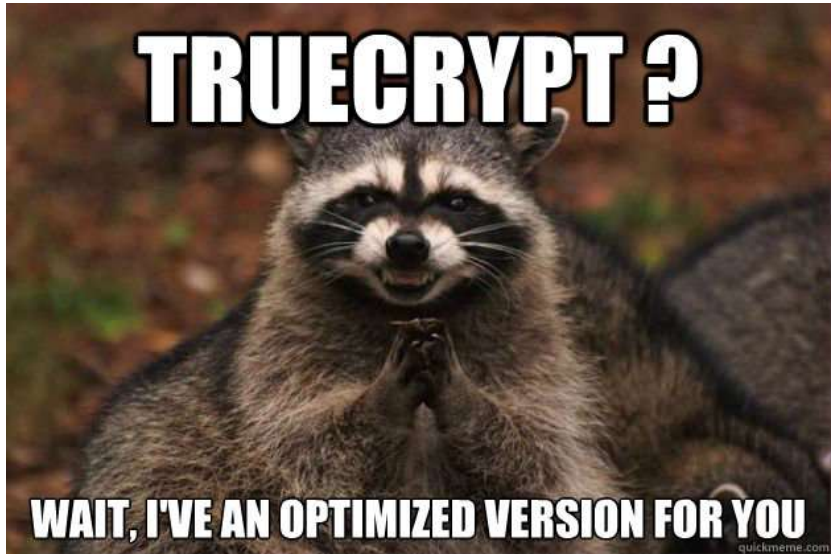
```
*
00000cd0: 00 00 00 00 00 00 00 00 - 00 56 73 65 67 75 72 6F |        Vseguro|
00000ce0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 |               |
*
00000d10: 00 00 00 00 00 00 00 00 - 00 00 5C 44 65 76 69 63 |          \Devic|
00000d20: 65 5C 53 5A 2D 4F 4E 32 - 56 38 53 42 00 00 00 00 |e\SZ-ON2V8SB    |
00000d30: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 |                |
*
00000d50: 00 00 00 00 00 00 00 00 - 00 00 45 00 3A 00 5C 00 |          E : \ |
00000d60: 53 00 61 00 6D 00 73 00 - 75 00 6E 00 67 00 20 00 |S a m s u n g   |
00000d70: 53 00 6F 00 66 00 74 00 - 77 00 61 00 72 00 65 00 |S o f t w a r e |
00000d80: 5C 00 53 00 65 00 63 00 - 72 00 65 00 74 00 5A 00 |\ S e c r e t Z |
00000d90: 6F 00 6E 00 65 00 5C 00 - 73 00 65 00 67 00 75 00 |o n e \ s e g u |
00000da0: 72 00 6F 00 2E 00 6D 00 - 73 00 72 00 00 00 00 00 |r o . m s r     |
00000db0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 |                |
*
00001150: 00 00 00 00 00 00 00 00 - 00 00 01 00 00 00 00 00 |                |
00001160: 00 40 00 00 00 00 00 F0 - E5 3D 00 00 00 00 00 00 | @       =      |
00001170: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 |                |
*
00001190: 00 00                   -                        |  |
00001192;
```

seguro.msr

```
D:\VMAC\vmacshare>hexdump -i ssz\seguro.msr | more
00000000: 6B AC FB F8 96 7B 7A FD - 78 E3 B9 7E 4E AD A3 E0 |k    {z x  ~N   |
00000010: C0 81 00 A8 05 48 11 9F - CD 5D B8 77 A3 AD EC 50 |     H   ] w   P|
00000020: 06 F4 8B 26 60 72 D5 5D - E7 C6 82 5C 54 34 9C DE |   &`r ]   \T4  |
00000030: 4B 6B B4 F4 3B B9 FD 5C - E5 13 F5 D7 16 DC 04 95 |Kk  ;  \        |
00000040: 1A 9D 48 08 AF BC 2D 66 - 48 63 AA FA 87 48 1A A9 |  H   -fHc   H  |
00000050: D3 2F F8 8A 81 EF 32 8A - 48 2B 0A 04 0F 2F 34 4F | /    2 H+   /4O|
00000060: 9F 6A 20 D9 16 1B D2 A8 - 01 40 1D AA 6C 27 AF 91 | j       @  l'  |
00000070: 26 76 32 3A CF 2F 2D 0E - 57 44 1B FB 5A D9 94 33 |&v2: /- WD  Z  3|
00000080: 68 11 F3 37 3C 88 CA 02 - 01 B5 F0 4B B6 BD AE BC |h  7<       K   |
00000090: 50 44 5B 5E 42 B4 19 B4 - 95 BA E3 C9 DF FB 7B 18 |PD[^B         {  |
000000a0: 91 30 DF D8 F2 D0 01 27 - C3 D4 35 5C 7C 13 28 66 | 0      '  5\| (f|
000000b0: 0D EE CD F7 6A D4 01 F8 - 2A 2E FD 51 20 85 CA 16 |    j   *. Q    |
000000c0: AA AE B8 1D 0E 5C 97 2C - CE 6B B7 80 A5 9A FD A1 |     \ , k      |
000000d0: 29 25 2D FF EC 0D A8 79 - BF 8B 19 5E 57 5C F4 10 |)%-    y    ^W\  |
000000e0: 76 8A C7 E4 5F 28 5B 0F - BD F6 4A 07 AE C6 0E A9 |v   _([   J     |
000000f0: 67 CA 40 69 2A DF 00 C6 - 2D A4 CE 19 1A A6 D3 CD |g @i*   -       |
00000100: 48 56 ED 24 D9 08 7A F7 - 01 52 64 31 E9 E2 9F 50 |HV $  z  Rd1   P|
00000110: 8E 98 81 F0 A4 30 D7 9C - 70 B7 87 43 58 0D D4 02 |     0  p  CX   |
00000120: 92 80 5B 22 8C 2F 8E 74 - 15 F4 51 3E 28 65 B2 C5 |  [" / t  Q>(e  |
00000130: 5A D3 31 4F DD EB 2D 55 - 8E 15 30 27 E6 3B F6 FD |Z 1O  -U  0' ;  |
00000140: 77 EF 19 19 A9 67 E0 11 - 59 C4 A6 1C 3F EB 33 D8 |w    g  Y   ? 3 |
00000150: CB 6B A5 48 98 36 7A 18 - C1 56 0D EF 4B 0E 12 78 | k H 6z  V  K  x|
00000160: C1 5D F5 03 05 DE 4A B5 - 31 00 4A C9 83 04 2E 8E | ]    J 1 J    . |
00000170: 7E F2 61 20 53 44 69 FA - 08 00 59 5A 6E B2 A8 73 |~ a SDi   YZn  s|
00000180: 5B 36 17 FE 13 EF 2C 7D - 9D CD 7F D4 A0 6C C1 16 |[6    ,}     l  |
00000190: 01 DB 37 15 02 CE 0B 5C - EC 56 C4 1B D5 C1 2D B2 |  7    \ V    -  |
000001a0: 8F F8 E9 B4 04 19 8E CE - 30 60 EF C9 50 25 CC 57 |        0`  P% W|
000001b0: 50 58 3E D2 39 A7 CD 36 - 08 6C 8F AC B1 50 E2 05 |PX> 9  6 l   P  |
000001c0: 3E BA 40 BA 91 22 81 18 - 13 22 29 D3 F9 63 84 1E |> @  "   ")  c  |
000001d0: BB CA 09 7E 4F 80 37 EE - 76 D7 E2 F3 43 AA 29 8B |   ~O 7 v   C ) |
000001e0: 66 4F 60 00 AC 31 B2 09 - E8 27 6C 70 2E 06 8B DC |fO`  1   'lp.   |
000001f0: 2F A2 79 8D 55 25 99 68 - 0A C6 86 4C 0E 32 D9 CC |/ y U% h   L 2  |
```

demo results

- Indeed, when unlocked, it does create a ***virtual disk*** (usually NTFS) which uses an encrypted file to read/write data.
- The encrypted file (***container***) is stored on the Samsung (Seagate/Maxtor) device.
- Who said ***Truecrypt?***



demo results

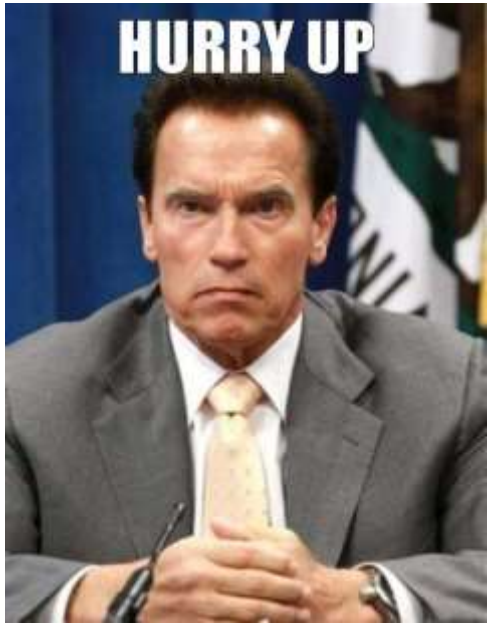
- Anyway the data looks like it's **encrypted**…
- … as specified in the user manual
  - http://www.seagate.com/files/www-content/support-content/documentation/samsung/downloads/ENG_Samsung%20SecretZone%20User%20Manual%20Ver%202.0.pdf

- " […] *Have the confidence that your data is secured using software-based 128-bit **AES**, 256-bit **AES**, or **Blowfish** 448 encryption. You can think of Samsung SecretZone as a safe within your computer.* […]"

cool but... **why bother?**

- I have no statistics...
- ... but SSZ does not seem widely used
- So, *why bother?*

- Because I had to
- During two criminal cases
- So, how could a **DF**ir guy handle it?



hurry up!

the 1st SSZ case

the *hurry up* case

- A domestic violence case
- I was asked from the Prosecutor to analyze the suspect's seized devices
- Four pc / laptops and many USB devices
  - *guess one of them was a Samsung pendrive?*
- I focused on the pc / laptops for chat / emails / web artifacts
  - Searching for evidences of *personal threats*
- No evidences
- Did not spot anything useful on USB devices too. ***BUT…***

check the ***checklist***!

- I have a checklist of steps to do for such cases
- One of them **is to check for encrypted files (known formats)**
- One **is the** entropy check
  - to spot encryption or *weird-ness*

- Cool, but I *did not* update the checklist for every devices
- I did not perform entropy test on a Samsung USB pendrive
- **When I realized that, one day before the milestone, I did it…**

ODI!

- *"OMG, what is that (high) entropy on that 4GB msr file?"*
  - This is how I discovered the existence of SSZ…
- How to decrypt an unknown *uber-secure safe*?
  - Googling? Just the manual and complains…
- ODI (*Offensive Digital Investigation*)!
  - To access protected data/system
  - To "exploit" passwords **re-use** or their **schema**
  - See "*ReVaulting! Decryption and opportunities*"

- *LSA secrets*
- *hiberfil*
- *System's secrets (dpapi)*
- *(maybe) rainbow table attack on NTLM*
- *Apps' credentials (LaZagne style!)*
- *User's credentials / vaults*

- *… whatever can be achieved in predetermined time…*

minidumps

- I got a bunch of *good* credentials from the suspect's system
- And two *juicy* minidumps of the (installed) SecretZone!
  - **%LOCALAPPDATA%\CrashDumps**
  - **%SYSTEMROOT%\Minidump**
- ***Strings*** on the minidumps
- ***grep*** to see if the password candidates were there
  - *to order the candidates*

the need of a Samsung device

- Is it possible then a dictionary attack? … Yes.. By *hand*!
- Internals unknown… no automation
- Need to use the SSZ software… **with a** Samsung **device**

- Could we use the original devide? Guess *not*…
  - *Tic tac tic tac…*

- I had an *expendable* pendrive and an *expendable* PC
  - The former for changing its Pid & Vid
  - Both annihilated after

- *using a write blocker… effectively blocked… not working*
- *fast patching SSZ crashed it… more time needed*

finally the data!

- The suspect re-used one of his Windows login passwords
  - By the way one of his email accounts passwords (source: his Thunderbird)
- What did I find? … suspence …

- Just **music**!
- It looks like he just did a test
  - Files timestamps
  - Files looked legit

- Not *proud* but finally I did it.

(*a word about ODI*)

- ODI *could work*
  - Just remember we are working **post-mortem**
- OS must expose something
- Still it needs some “wrong” user behavior
  - Password re-use, schemas, etc.

- The 500 Encrypted Archives case
- The FDE (Luks) case
- *Never give up!*

the *spy* case

the 2st SSZ case

the *spy* case

- Well, not real a 007 case… The suspect was charged with taking photo and video of his neighbors… without them knowing.
- Basically he was *spying* on the female neighbors.
- When he was discovered LE was involved.

- One of the seized device was a **M3 Samsung USB disk**
- This time I immediately spot a **200GB MSR** file…
- … with enough time to work on it.

automatic cracking

- The goal was to understand how encryption was used
- To make possible automatic cracking
  - *with our ripper friend john, hashcat, whatever*

- Virtual disk… a driver should be in place
- Expect a user-land application with one driver at least
- My first hypothesis was to work on the userland, but first…

who did it?

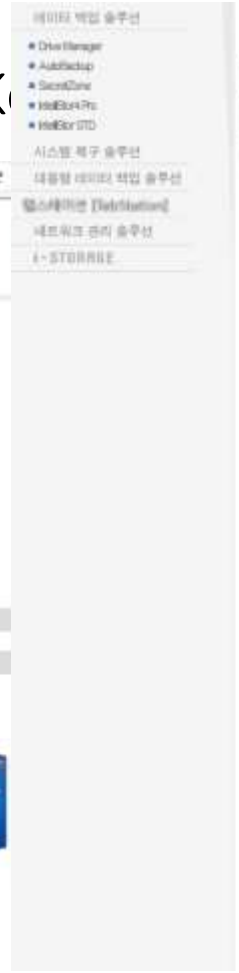
Portable SecretZone version

property	value
file-type	n/a
date	n/a
language	Korean
code-page	Unicode UTF-16, little endian
Comments	n/a
CompanyName	Clarus, Inc.
FileDescription	Samsung Portable SecretZone
FileVersion	1.0.81.0
InternalName	Portable SecretZone.exe
LegalCopyright	Copyright 2011 Clarus,Inc.
LegalTrademarks	n/a
OriginalFilename	Portable SecretZone.exe
PrivateBuild	n/a
ProductName	Samsung Portable SecretZone
ProductVersion	1.0.0.1
SpecialBuild	n/a

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            7c:12:97:b5:46:91:be:5c:26:6e:74:3a:7c:a2:18:bf
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, O=Thawte, Inc., CN=Thawte Code Signing CA - G2
        Validity
            Not Before: May  5 00:00:00 2012 GMT
            Not After : May  5 23:59:59 2013 GMT
        Subject: C=KR, ST=SEOUL, L=Seocho-gu, O=Clarus, Inc., CN=Clarus, Inc.
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                    [...]
X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 CRL Distribution Points:
                Full Name:URI:http://cs-g2-crl.thawte.com/ThawteCSG2.crl
            X509v3 Extended Key Usage:
                Code Signing, Microsoft Commercial Code Signing
            2.5.29.4: [...]
            Authority Information Access: OCSP - URI:http://ocsp.thawte.com
            Netscape Cert Type:
                Object Signing
    Signature Algorithm: sha1WithRSAEncyption [...]
```

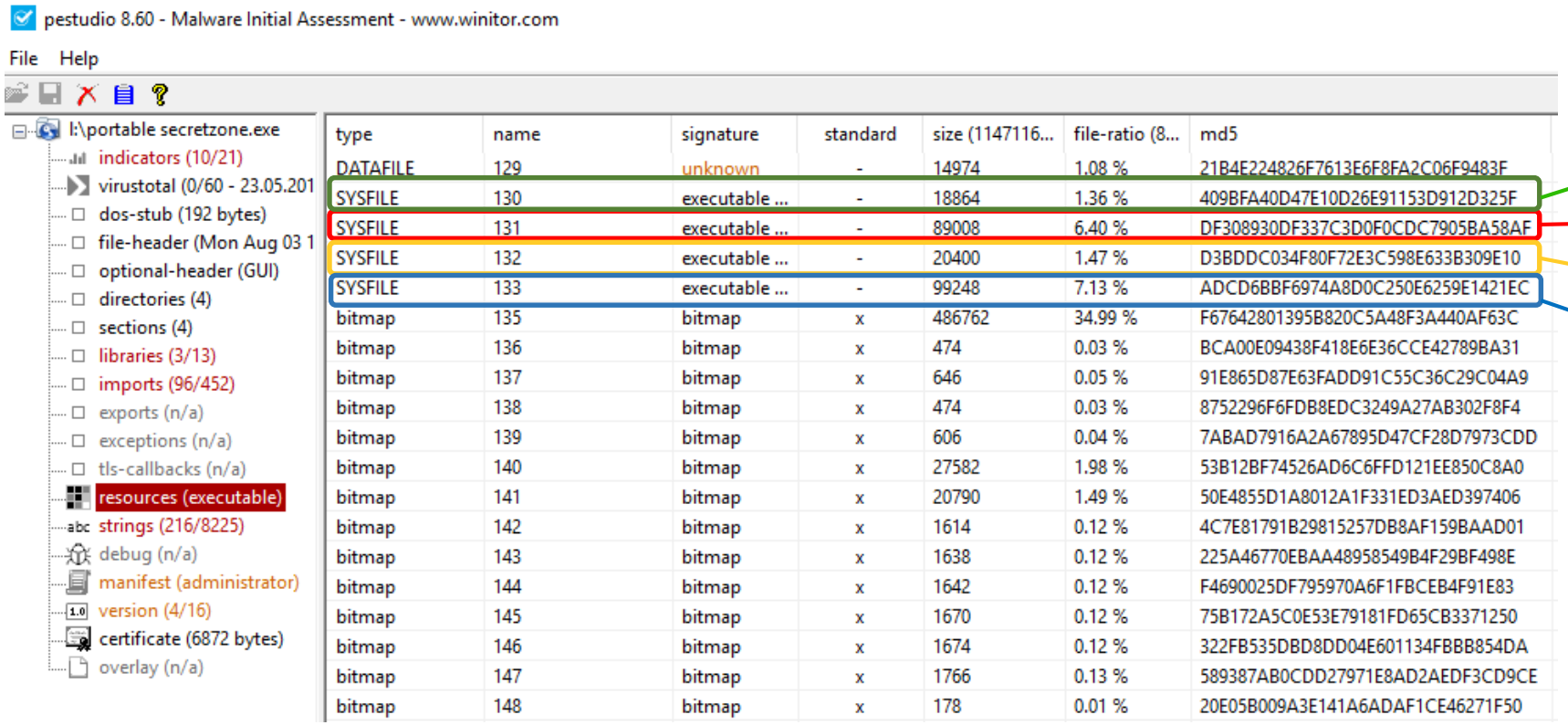
clarus

www.clarussoft.com (K



om (English language)

the drivers



type	name	signature	standard	size (1147116...	file-ratio (8...	md5
DATAFILE	129	unknown	-	14974	1.08 %	21B4E224826F7613E6F8FA2C06F9483F
SYSFILE	130	executable ...	-	18864	1.36 %	409BFA40D47E10D26E91153D912D325F
SYSFILE	131	executable ...	-	89008	6.40 %	DF308930DF337C3D0F0CDC7905BA58AF
SYSFILE	132	executable ...	-	20400	1.47 %	D3BDDC034F80F72E3C598E633B309E10
SYSFILE	133	executable ...	-	99248	7.13 %	ADCD6BBF6974A8D0C250E6259E1421EC
bitmap	135	bitmap	x	486762	34.99 %	F67642801395B820C5A48F3A440AF63C
bitmap	136	bitmap	x	474	0.03 %	BCA00E09438F418E6E36CCE42789BA31
bitmap	137	bitmap	x	646	0.05 %	91E865D87E63FADD91C55C36C29C04A9
bitmap	138	bitmap	x	474	0.03 %	8752296F6FDB8EDC3249A27AB302F8F4
bitmap	139	bitmap	x	606	0.04 %	7ABAD7916A2A67895D47CF28D7973CDD
bitmap	140	bitmap	x	27582	1.98 %	53B12BF74526AD6C6FFD121EE850C8A0
bitmap	141	bitmap	x	20790	1.49 %	50E4855D1A8012A1F331ED3AED397406
bitmap	142	bitmap	x	1614	0.12 %	4C7E81791B29815257DB8AF159BAAD01
bitmap	143	bitmap	x	1638	0.12 %	225A46770EBAA48958549B4F29BF498E
bitmap	144	bitmap	x	1642	0.12 %	F4690025DF795970A6F1FBCEB4F91E83
bitmap	145	bitmap	x	1670	0.12 %	75B172A5C0E53E79181FD65CB3371250
bitmap	146	bitmap	x	1674	0.12 %	322FB535DBD8DD04E601134FBBB854DA
bitmap	147	bitmap	x	1766	0.13 %	589387AB0CDD27971E8AD2AEDF3CD9CE
bitmap	148	bitmap	x	178	0.01 %	20E05B009A3E141A6ADAF1CE46271F50

mdf16.sys filter driver x86

mvd23.sys virtual disk driver x86

mdf16.sys filter driver x64

mvd23.sys virtual disk driver x64

Note: different drivers versions depending on the main exe version (es: mvd22.sys)

the kernel path

- These drivers are full of **useful** error messages
- Many clues of what routines will do
- So I started looking at drivers, **IOCTL** first

Error message

Error reporting

```
loc_1BCD0:
mov     eax, [rdi-8]
lea     rdx, aAddmsrinform_3 ; "AddMsrInformation [%s]: invalid Compone"...
mov     r9d, r12d
mov     r8, rbx
mov     ecx, 5
mov     [rsp+48h+var_28], eax
call    lock_report
mov     eax, 0C000000Dh
jmp     loc_1BC53
AddMsrInformation endp
```

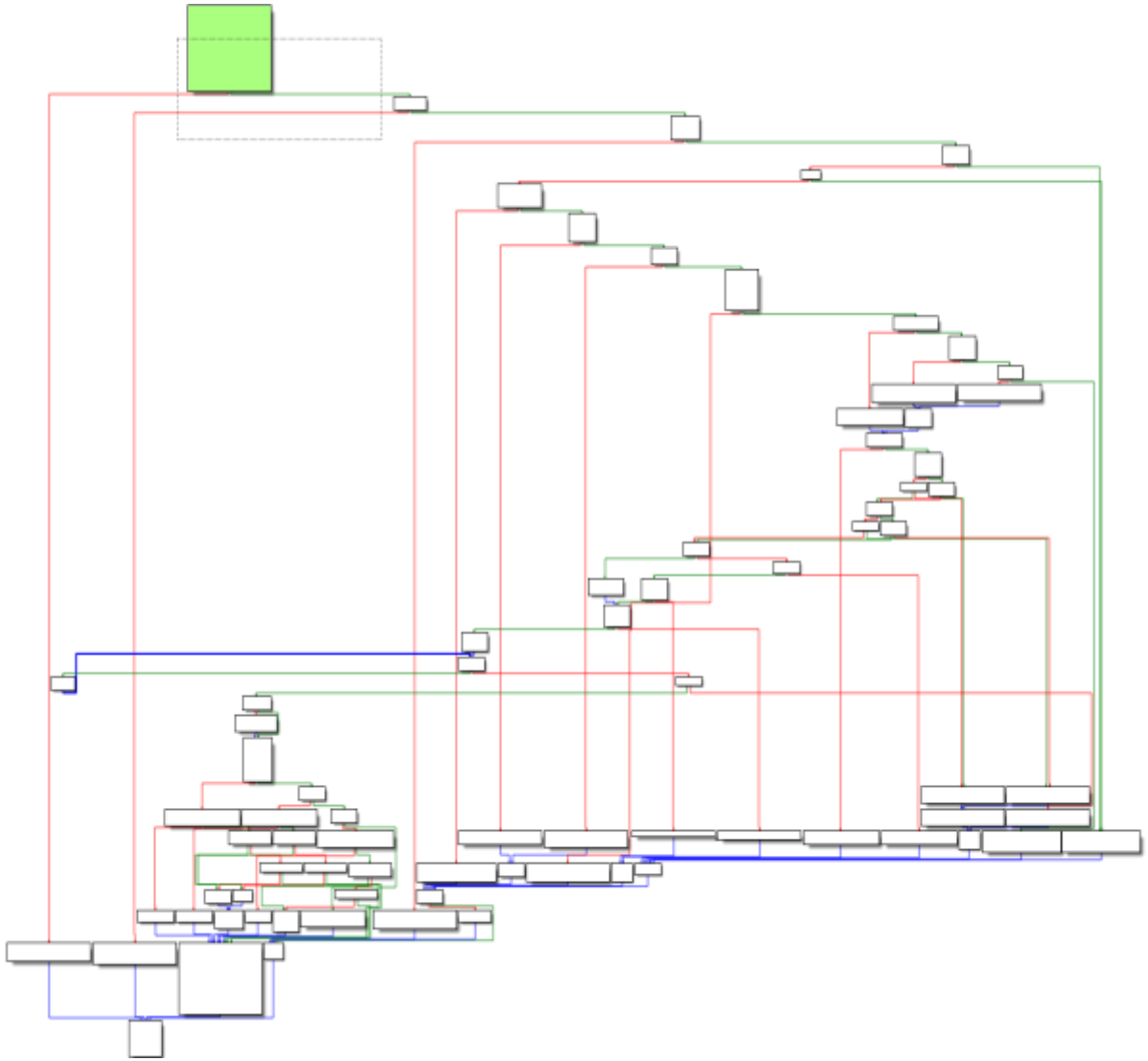
the mvd*nn* driver

- From the error messages, is clear that great part of the job is done by the **virtual disk driver**
  - Creation of the container, mounting, setting password *id* (?), AES and BlowFish algorithms etc.
- So I started looking at the driver, which did look promising
- Both static analysis and kernel debugging

```
IOCTL_MVD_CREATE_VOLUME
IOCTL_MVD_DELETE_VOLUME
IOCTL_MVD_CONNECT_VOLUME
IOCTL_MVD_DISCONNECT_VOLUME
IOCTL_MVD_GET_VOLUME_COUNT
IOCTL_MVD_GET_VOLUME_INFO
IOCTL_MVD_GET_ALL_VOLUME_INFO
IOCTL_MVD_GET_VOLUME_MSR_INFO
IOCTL_MVD_CREATE_SYMLINK
IOCTL_MVD_DELETE_SYMLINK
IOCTL_MVD_SET_TIMER
IOCTL_MVD_GET_IDPASS
IOCTL_MVD_SET_IDPASS
IOCTL_MVD_SET_VOLUME_REMOVED
IOCTL_MVD_WRITE_IMAGE_FILE
IOCTL_MVD_READ_IMAGE_FILE
IOCTL_MVD_RESET_VOLUME_SIZE
IOCTL_MVD_SET_FILE_INFO
IOCTL_MVD_SET_VALID_DATA
IOCTL_MVD_SET_VALID_DATA_DEV
IOCTL_MVD_SET_IMAGE_PWD
IOCTL_MVD_GET_IMAGE_VERSION
IOCTL_MVD_UPDATE_VERSION0
IOCTL_MVD_ADD_MSR_LIST
IOCTL_MVD_OPEN_EVENT
IOCTL_MVD_CLOSE_EVENT
IOCTL_MVD_CREATE_EVENT
IOCTL_MVD_MOUNTMGR_MOUNT
IOCTL_MVD_MOUNTMGR_DISMOUNT
….
```

pizza connection?
No, *volume connection*

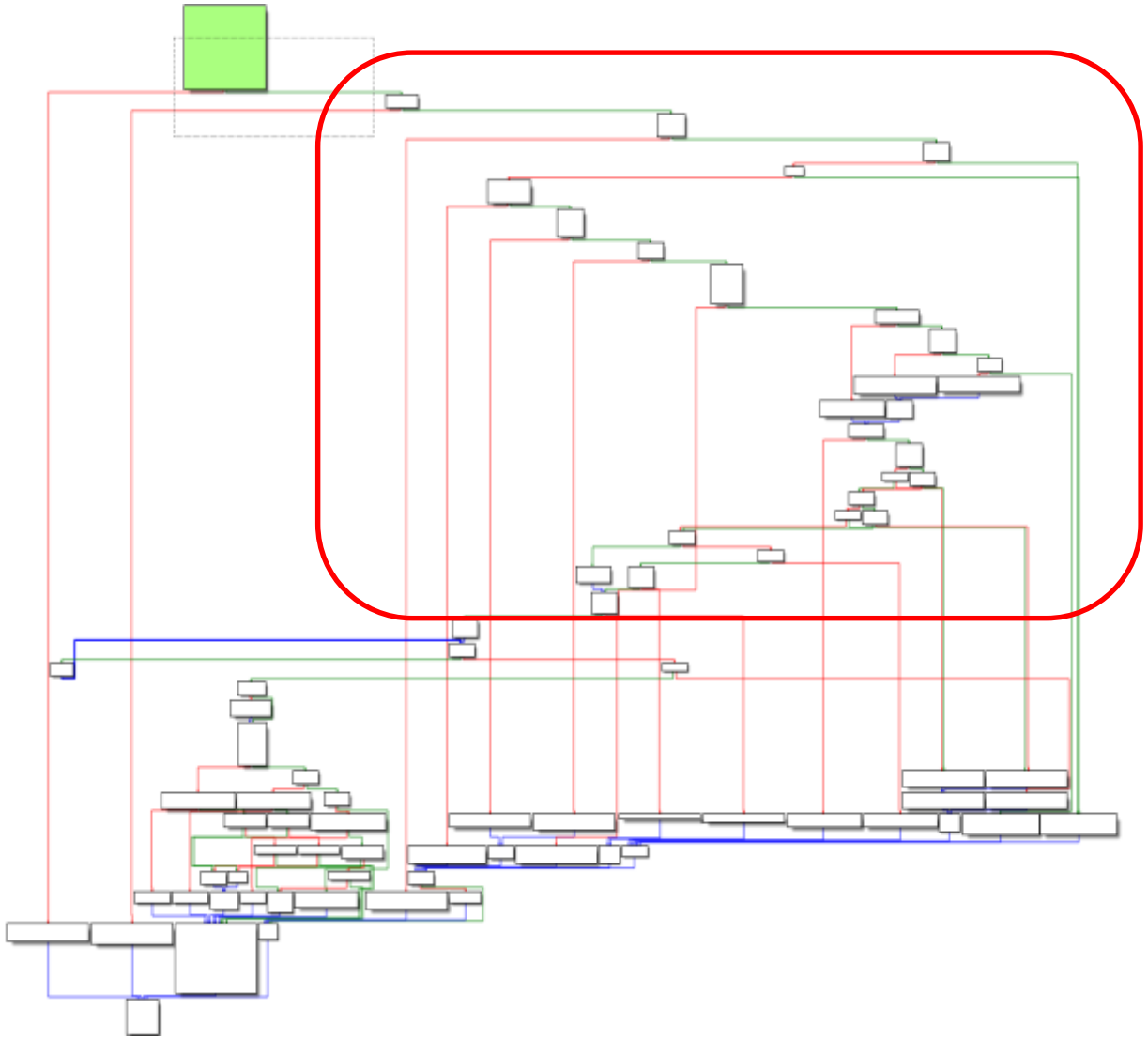
- I isolated an interesting routine I called **VolumeConnect**
- With **parameters,** but apparently **nothing related** to a password or to a password derivation
- Still, when finished, **the volume was accessible**!
- Weird



header decryption

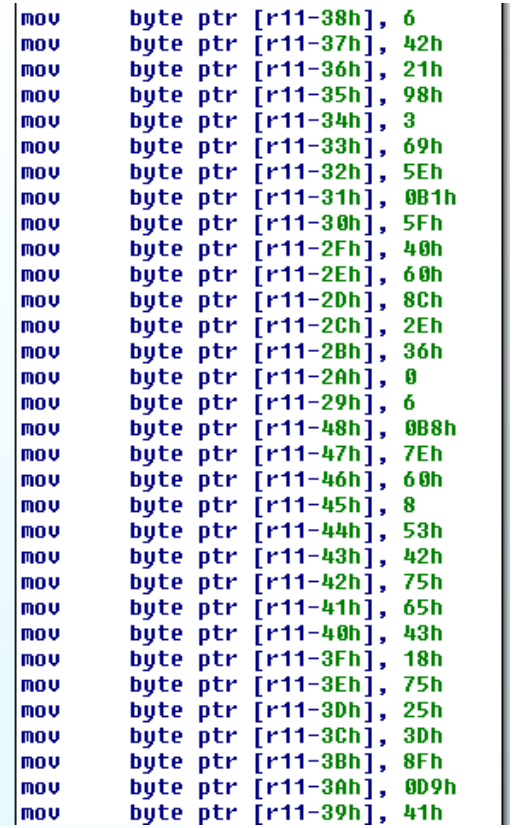
- The routine decrypts a sort of an header
- First **16KB** of the MSR file
- Using AES 128 in CBC mode
- Guess what? With a **fixed key**
- At least for version **1 & 2**
  - Version 0 should be cleartext

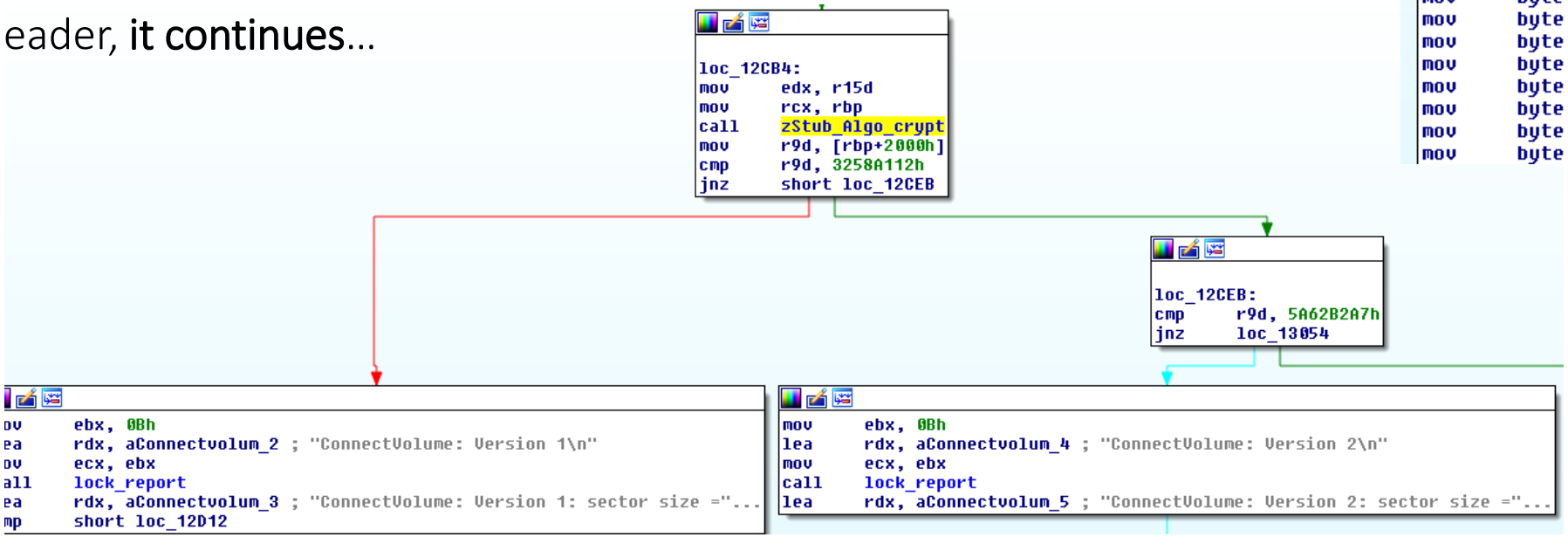
```
HEADER_KEY =
'\x06\x42\x21\x98\x03\x69\x5E\x
B1\x5F\x40\x60\x8C\x2E\x36\x00\
x06'
```



header decryption

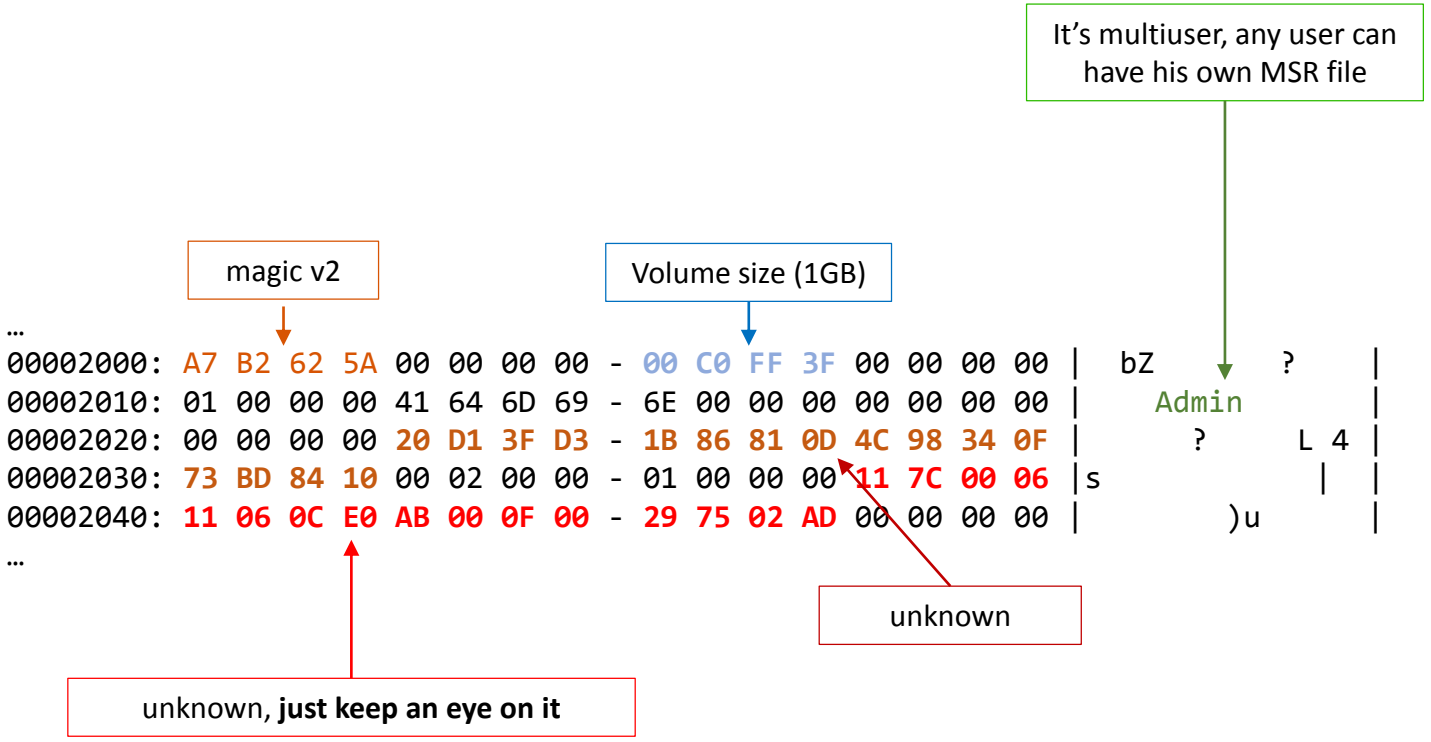
- But the routine does not stop after parsing the header
- The real header starts after **8KB**
- It does not return back some data from the header, **it continues**…





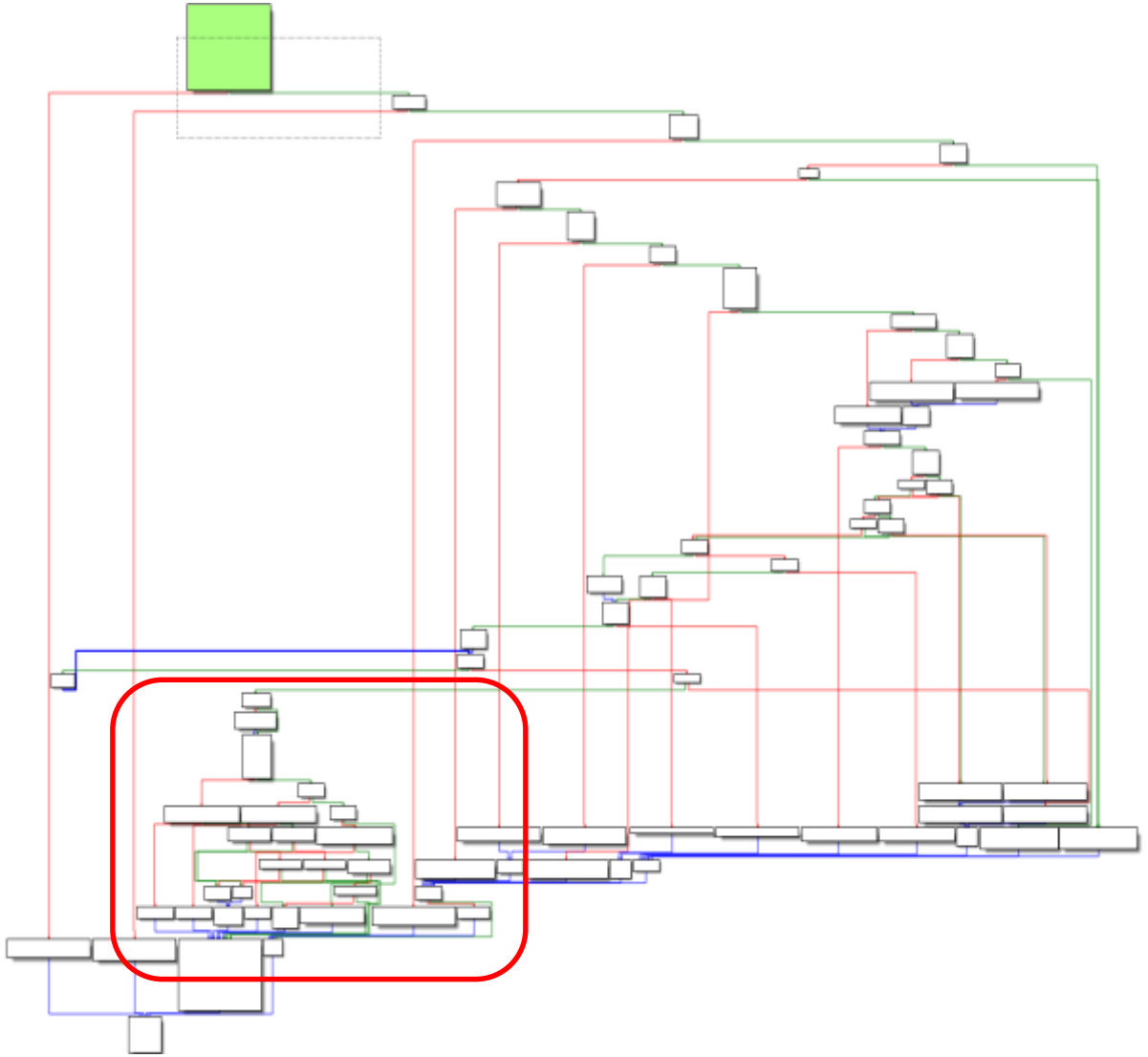
header decrypted

```
mov     byte ptr [r11-38h], 6
mov     byte ptr [r11-37h], 42h
mov     byte ptr [r11-36h], 21h
mov     byte ptr [r11-35h], 98h
mov     byte ptr [r11-34h], 3
mov     byte ptr [r11-33h], 69h
mov     byte ptr [r11-32h], 5Eh
mov     byte ptr [r11-31h], 0B1h
mov     byte ptr [r11-30h], 5Fh
mov     byte ptr [r11-2Fh], 40h
mov     byte ptr [r11-2Eh], 60h
mov     byte ptr [r11-2Dh], 8Ch
mov     byte ptr [r11-2Ch], 2Eh
mov     byte ptr [r11-2Bh], 36h
mov     byte ptr [r11-2Ah], 0
mov     byte ptr [r11-29h], 6
mov     byte ptr [r11-48h], 0B8h
mov     byte ptr [r11-47h], 7Eh
mov     byte ptr [r11-46h], 60h
mov     byte ptr [r11-45h], 8
mov     byte ptr [r11-44h], 53h
mov     byte ptr [r11-43h], 42h
mov     byte ptr [r11-42h], 75h
mov     byte ptr [r11-41h], 65h
mov     byte ptr [r11-40h], 43h
mov     byte ptr [r11-3Fh], 18h
mov     byte ptr [r11-3Eh], 75h
mov     byte ptr [r11-3Dh], 25h
mov     byte ptr [r11-3Ch], 3Dh
mov     byte ptr [r11-3Bh], 8Fh
mov     byte ptr [r11-3Ah], 0D9h
mov     byte ptr [r11-39h], 41h
```



surprise!

- The routine continues and when it finishes, the volume is **unlocked**
- What happens is that the routine will use the previous 16 bytes key (unknown) to decrypt the rest of the volume!



- Using **AES 128** in ECB mode (default)
  - Depending on the settings, issue holds

```
…
00002000: A7 B2 62 5A 00 00 00 00 - 00 C0 FF 3F 00 00 00 00
00002010: 01 00 00 00 41 64 6D 69 - 6E 00 00 00 00 00 00 00
00002020: 00 00 00 00 20 D1 3F D3 - 1B 86 81 0D 4C 98 34 0F
00002030: 73 BD 84 10 00 02 00 00 - 01 00 00 00 11 7C 00 06
00002040: 11 06 0C E0 AB 00 0F 00 - 29 75 02 AD 00 00 00 00
```

put it in action

- I did not complete 100% the RE process
- Still to look at special cases, different configurations, etc.
- E.g. in the Python script on the right, I **hardcoded** the decryption key position
- Didn't test if it could change due to username length

```
from __future__ import print_function
from Crypto.Cipher import AES
import sys

CHUNK_SIZE = 4096
HEADER_SIZE = 16384
# AES key, for different crypto algorithms, different keys.
HEADER_KEY = '\x06\x42\x21\x98\x03\x69\x5E\xB1\x5F\x40\x60\x8C\x2E\x36\x00\x06'

def main():
    with open(sys.argv[1], 'rb') as input_file:
        header_enc = input_file.read(HEADER_SIZE)
        decryptor = AES.new(HEADER_KEY, AES.MODE_CBC, 16 * '\x00')
        header_dec = decryptor.decrypt(header_enc)

        body_decryption_key = header_dec[0x203c:0x204C]
        print('Decoding key: {}'.format(body_decryption_key.encode('hex')))

        if len(sys.argv) != 3:
            print('No output file specified, giving up...')
            sys.exit(0)

        decryptor = AES.new(body_decryption_key, AES.MODE_ECB, 16 * '\00')

        with open(sys.argv[2], 'wb') as output_file:
            while True:
                chunk_enc = input_file.read(CHUNK_SIZE)
                if len(chunk_enc) == 0:
                    break
                chunk_dec = decryptor.decrypt(chunk_enc)
                output_file.write(chunk_dec)
                sys.stdout.write('.')
                sys.stdout.flush()
```

demo time

conclusions

- 2nd case:
  - once decrypted the SSZ container I found many images and video
  - Part of them already recovered from the seized devices
    - delete files and file carving
- The **SSZ** software made by Clarus “suffers” a major vulnerability
  - by design?
- The password (its derivation) is kept inside the container.
- Encrypted data (MSR containers)

can be decrypted without knowing the user password!

let’s keep in touch


Francesco Picasso


Reality Net System Solutions


@dfirfpi


https://github.com/dfirfpi


blog.digital-forensics.it

Thank you!