Catch the QuBit

Capture the Flag Contest

The Solution

Document classification: Public

Prepared by

cesnet

Prizes by

QuBit Conference

CESNET, a. l. e.
Forensic Laboratory
Zikova 4, 160 00
Prague 6, Czech Republic

Table of Content

CESNET, a. l. e.
Forensic Laboratory
Zikova 4, 160 00
Prague 6, Czech Republic

1 About the Solution

The described solution is only one of many possible ones. If you find some other interesting solution that you want to share, send us a short description to flab@cesnet.cz with the subject `Catch the Qubit - Interesting solution`.

2 Initialization

Files from the downloaded archive are extracted.

```
$ unzip catch-the-qubit_bundle.zip
Archive: catch-the-qubit_bundle.zip
  inflating: catch-the-qubit_instructions.pdf
  inflating: usb_flash_image.dd.gz
 extracting: usb_flash_image.dd.gz.md5checksum
```

The correct checksum is checked.

```
$ md5sum -c usb_flash_image.dd.gz.md5checksum
usb_flash_image.dd.gz: OK
```

The USB flash drive image is extracted.

```
$ gzip -d usb_flash_image.dd.gz
```

Used tools

- `unzip`, `gzip`, `md5sum`

3 The Analysis

The goal of the analysis is to find out whether there are some internal classified documents are present on the seized USB flash drive. The owner of the device was pretty sure that nobody can find such documents.

3.1 USB Drive Content Analysis

The USB flash drive image is mounted and examined.

```
$ mkdir mount
$ sudo mount -o loop,ro,noatime,noload usb_flash_image.dd mount
```

The browsing of mounted image shows that there are some papers related to steganography (a known hobby of the suspect), some guitar related videos, and lyrics for some songs.

The unallocated space of the drive is analysed, because some data may be present in unallocated blocks or some deleted files may be there.

```
$ blkls -A usb_flash_image.dd > unallocated
$ binwalk -Mre unallocated
Scan Time: 2018-03-15 18:12:53
Target File: unallocated
MD5 Checksum: d3cd90f03b95cb10082efbe2bba23f6e
Signatures: 344

DECIMAL HEXADECIMAL DESCRIPTION
--------------------------------------------------------------------------------
```

Data carving is performed on the extracted unallocated space, but nothing is found. In fact, the unallocated space is full of zero bytes (0x00) as can be viewed by any hexeditor.

```
hexdump unallocated
0000000 0000 0000 0000 0000 0000 0000 0000 0000
*
d7aa6000
```

The `hexdump` shows there are 0xd7aa6000 zero bytes (i.e. 3618267136B) – the exact size of `unallocated` file. That means the hidden contents must be within one (or more) of the visible files.

There are many files on the USB drive and nobody wants to look at all of them. Thus, the file timeline is created in order to try to identify the important files.

```
$ fls -r -m / usb_flash_image.dd > filetimes.fls
$ mactime -d -y -b filetimes.fls > filetimes.csv
$ libreoffice filetimes.csv
```

Nearly all files were copied to the flash drive in the evening before exfiltration attempt (exactly between the timestamps `2018-03-13T22:57:05Z` and `2018-03-13T22:57:18Z`). The only exception are the files

- `/lyrics/lyrics_dear-john-letter.txt`
- `/video/Dear John Letter (original).mp4`
- `/video/How to play Dear John Letter.mp4`

that were copied to the drive at `2018-03-14T10:31:27Z` and `2018-03-14T10:31:28Z`.

The file types can be identified by an utility `file` on mounted image.

```
$ file lyrics_dear-john-letter.txt
lyrics_dear-john-letter.txt: ASCII text, with CRLF line terminators

$ file Dear\ John\ Letter\ \(original\).mp4
Dear John Letter (original).mp4: ISO Media, MP4 Base Media v1 [IS0 14496-12:2003]

$ file How\ to\ play\ Dear\ John\ Letter.mp4
How to play Dear John Letter.mp4: data
```

The first file is a plain text file, the second one is video file, and the third one contains some generic data (that is very suspicious for an `mp4` file).

The Text File

Inside the *text file* is lyrics for a well known song *Dear John Letter*, nothing interesting is present... oh, wait, the last strophe contains words that are not related to the hell of love and war!

```
Will you please decode the data, your boss wants it now.
When I tell you where the password is, you won't care, dear, anyhow.
Now watch the video carefully and I'll wed your brother Don
Will you decode the veracrypt file forever, Dear John
```

It looks like that the *data file* may be a VeraCrypt container and the appropriate password for decryption could be hidden in the *video file*.

The Video File

The first attempt to find the password is to play the video file in the `vlc player`.

```
$ vlc Dear\ John\ Letter\ \(original\).mp4
```

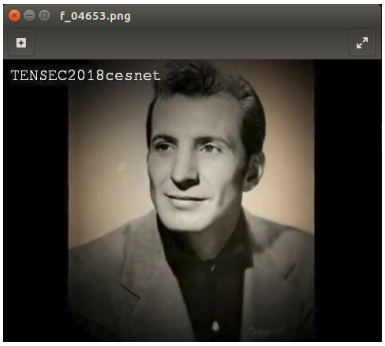
The song takes less than three minutes (exactly 2:35) and about halfway throught some white text shortly appears at upper left corner. It is a nearly impossible to stop the video at the right time and going throught the video frame by frame by pressing `E` key is exhausting. Fortunately, there is a more convenient way based on `ffmpeg`, which will extract each frame to a separate image.

```
$ mkdir frames
$ ffmpeg -i Dear\ John\ Letter\ \(original\).mp4 -qscale:v 2 "frames/f_%05d.png"
```

There are 9303 frames in total. The middle one (4652th frame, to be exact) is checked at first.

```
$eog frames/f_04652.png
```

No password-like string can be seen, thus the adjanced frames have to be checked as well. There is nothing interesting is be found in the preceding frames, but the frames 4653, 4654, and 4655 all contain a white text in the upper left corner.



Great! The recipient of the USB Flash Drive should use the password `TENSEC2018cesnet` to mount the VeraCrypt container.

The Data File

According to the *text file*, this *data file* should be a VeraCrypt container and the string from the *video file* should be used as the password for this container. This hypothesis is checked by an attempt to mount the VeraCrypt container in a forensic way (meaning in read-only mode).

```
# mkdir veramount
# veracrypt -m ro --password=TENSEC2018cesnet How\ to\ play\ Dear\ John\ Letter.mp4 veramount
# ls veramount/
form-air2018-027v3.pdf materials_cze materials_eng test.txt
```

Hypothesis confirmed. The next step is to find the classified internal documents in the mounted VeraCrypt container.

Used tools

- Sleuthkit (`blkls`, `fls`, `mactime`), `hexdump`, `binwalk`, `file`, `ffmpeg`, VLC, Eye of GNOME, VeraCrypt, LibreOffice

3.2 Veracrypt Container Analysis

First of all, the unused space should be scanned for deleted files (the appropriate device has to be identified).

```
# mount | grep veramount
/dev/mapper/veracrypt1 on /root/veramount type vfat ...
# blkls -A /dev/mapper/veracrypt1 > verafree
$ binwalk -Mre verafree

Scan Time: 2018-03-15 18:32:47
Target File: verafree
MD5 Checksum: c1901f047f23b998a98e43233bc73e2e
Signatures: 344

DECIMAL HEXADECIMAL DESCRIPTION
--------------------------------------------------------------------------------
```

No known file format seems to be hidden in the unallocated space. Contrary to the USB flash drive image, the unallocated space is full of random data as can be viewed by any hexeditor.

Closer examination of the mounted VeraCrypt containter shows, that there are present

- several PDF documents in two directories – some leaflets about CESNET services (obviously public documents, they are even not classified),
- a PDF form – a form for using personal aircraft for business trips (could be an internal document, but is also not classified),
- a text file `test.txt` with the text *Veracrypt Container* (possibly marking this directory as safe to use), and
- a (hidden) directory `.git`.

The most promissing place to look at is the `.git` directory, which is usually used by Git – a popular version control system for tracking changes in computer files and coordinating work on those files among multiple people. It may be worth it to examine this clue further and analyse how Mr. Ethan has used this tool.

Used tools

- VeraCrypt, Sleuthkit (`blkls`), `binwalk`

3.3 Git Repository Analysis

The Git log contains important information.

```
# git reflog
3c957c1 HEAD@{0}: checkout: moving from transport to master
6138acd HEAD@{1}: commit: Transport ready.
3c957c1 HEAD@{2}: checkout: moving from master to transport
3c957c1 HEAD@{3}: commit (initial): Carrier ready to use.
```

Good boy, this Evilboy! Our villain has a tendency to document everything, good for us! The term *carrier* is related to steganography, it is usually some file capable of *carrying* a payload. The Git commit with the description *Transport ready* also indicates that the payload is already in place. Further details about this commit are also available.

```
# git log 6138acd -p -1
commit 6138acd45965feb2c2145029a7bedf8b9cce07f2
Author: Ethan <ethan@tensec.cz>
Date: Wed Mar 14 09:05:26 2018 +0100

   Transport ready.
```

```
diff --git a/form-air2018-027v3.pdf b/form-air2018-027v3.pdf
old mode 100755
new mode 100644
index 011f5cc..fd6bae0
Binary files a/form-air2018-027v3.pdf and b/form-air2018-027v3.pdf differ
```

The commit changed just one file – `form-air2018-027v3.pdf`. Both versions of the file (from *initial* commit and from *Transport ready* commit) are extracted, so they can be compared.

```
$ mkdir ../files
$ cp form-air2018-027v3.pdf ../files/file-3c957c1.pdf
$ git checkout 6138acd
$ cp form-air2018-027v3.pdf ../files/file-6138acd.pdf
```

The size of the files differs significantly.

```
$ ls -sh1
total 4,1M
 76K file-3c957c1.pdf
4,0M file-6138acd.pdf
```

Used tools

- Git

3.4 PDF File Analysis

The *pdf-parser* (free tool by Stevens Didier) is used to get the basic idea about the structure and the embedded objects of the PDF files.

```
$ pdf-parser.py -a file-3c957c1.pdf
Comment: 3
XREF: 1
Trailer: 1
StartXref: 1
Indirect object: 27
  16: 2, 3, 5, 7, 9, 10, 12, 14, 15, 17, 19, 20, 22, 24, 25, 27
 /Catalog 1: 26
 /Font 3: 13, 18, 23
 /FontDescriptor 3: 11, 16, 21
 /Page 1: 1
 /Pages 1: 8
 /XObject 2: 4, 6
```

```
$ pdf-parser.py -a file-6138acd.pdf
Comment: 3
XREF: 1
Trailer: 1
StartXref: 1
Indirect object: 28
  17: 2, 3, 5, 7, 9, 10, 12, 14, 15, 17, 19, 20, 22, 24, 25, 27, 666
 /Catalog 1: 26
 /Font 3: 13, 18, 23
 /FontDescriptor 3: 11, 16, 21
 /Page 1: 1
 /Pages 1: 8
 /XObject 2: 4, 6
```

Both files contain the same objects, except for one – an object number `666`, which is only present in the newer (transport ready) file. The same tool is used to get more information about this particular object.

```
$ pdf-parser.py -o 666 file-6138acd.pdf
obj 666 0
 Type:
 Referencing:
 Contains stream
  <<
    /Length 8775010
    /Filter /FlateDecode
  >>
```

Object consists of just one stream, which is extracted to a dedicated file.

```
$ pdf-parser.py -o 666 -d object666 -f file-6138acd.pdf
$ file object666
object666: ASCII text, with very long lines, with no line terminators
```

The *object666* file consists of small subset characters – digits `0-9` and letters `a-f`, typical for hex encoding. The conversion to characters represented by the hex values is simple.

```
$ xxd -r -p -c 0 object666 > object666.dehex
$ file object666.dehex
object666.dehex: ASCII text
```

The file *object666.dehex* contains character set typical for base64 encoding. Decoding is piece of cake.

```
$ base64 -d object666.dehex > object666.debase
$ file object666.debase
object666.debase: Zip archive data, at least v2.0 to extract
```

Used tools

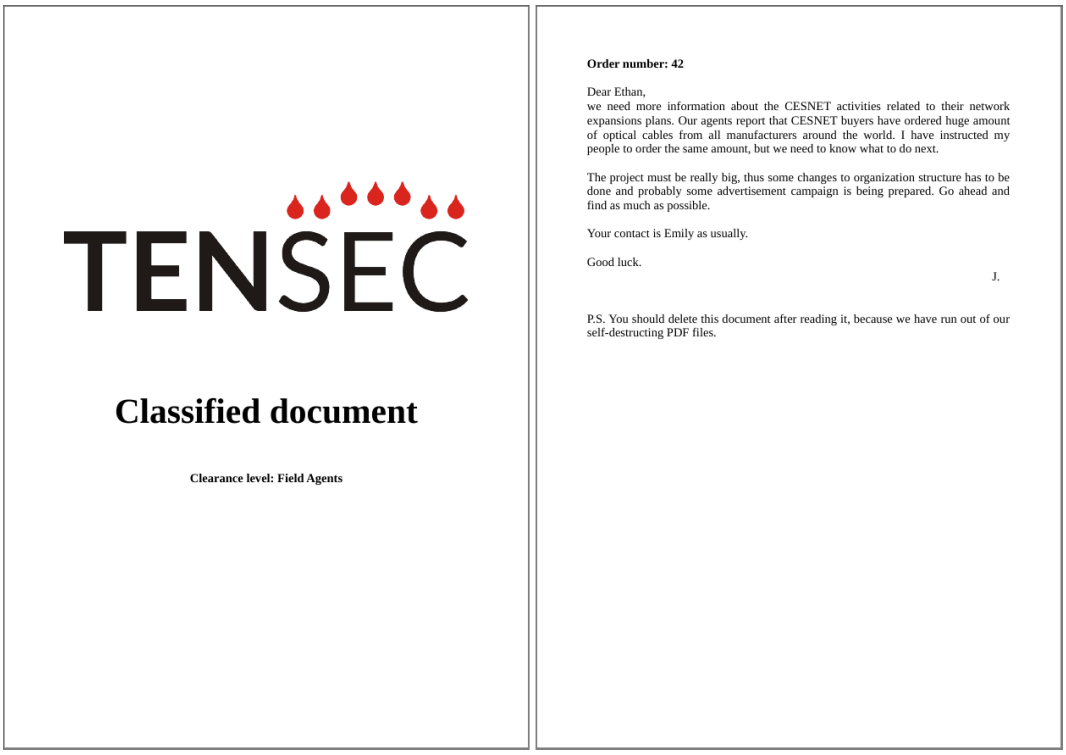
- `pdf-parser`, `xxd`, `base64`

3.5 Secret Archive Analysis

The archive found in a hidden PDF stream is extracted.

```
$ unzip object666.debase
Archive: object666.debase
  inflating: campaign_mars2019v8.pdf
  inflating: mars_scheme2017v36.pdf
  inflating: message.txt
  inflating: order42.pdf
  inflating: organization_scheme2018v2.pdf
```

The PDF file `order42.pdf` contains a TENSEC classified document – an order from J. for field agent Ethan to spy on CESNET.



TENSEC

Classified document

Clearance level: Field Agents

Order number: 42

Dear Ethan,
we need more information about the CESNET activities related to their network expansions plans. Our agents report that CESNET buyers have ordered huge amount of optical cables from all manufacturers around the world. I have instructed my people to order the same amount, but we need to know what to do next.

The project must be really big, thus some changes to organization structure has to be done and probably some advertisement campaign is being prepared. Go ahead and find as much as possible.

Your contact is Emily as usually.

Good luck.

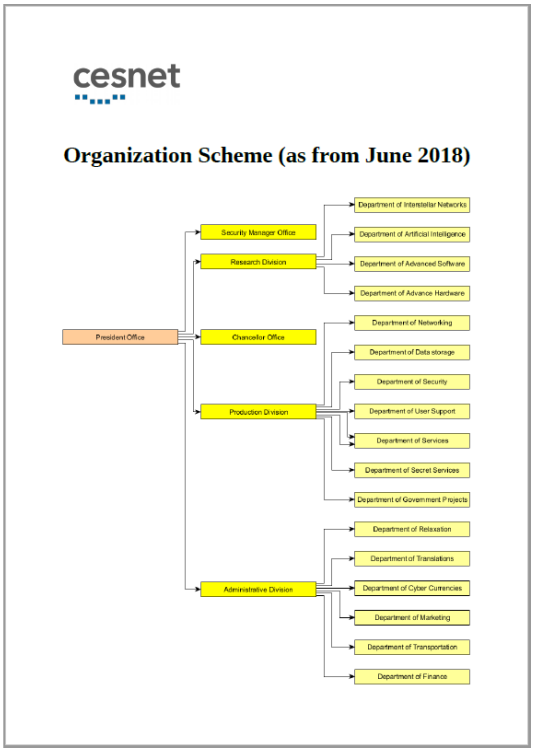
J.

P.S. You should delete this document after reading it, because we have run out of our self-destructing PDF files.

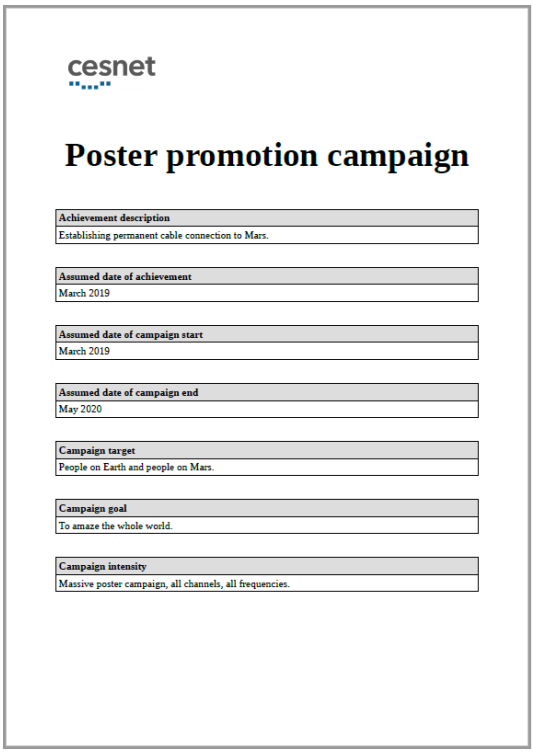
The text file `message.txt` contains a personal message from Ethan to John (probably John Walker of TENSEC).

```
Dear John,
I have the information you want to know, it is really big! You can look at the new
organizational scheme (notice the new Department of Interstellar Networking!!!),
details about poster advertisement campaign (it will start in few months), and the
technical scheme how the CESNET's engineers want to connect Mars to CESNET2 Network.
It looks like pure madness at first sight ... but it is NOT!!! I have also gathered
some leaflets about CESNET services - no news there.

I'm afraid they've suspected or even revealed me. That guy in my office, he is
watching me everyday. The gate-keeper, he asked me yesterday if I smuggled anything!
They certailny go after me!

This is the last data transfer for several weeks, I will have to pretend some
contagious disease.

E.
```

The PDF file `organization_scheme2018v2.pdf` is the first document matching the task – to find the CESNET classified documents. The content is the new organization scheme.



The PDF file `campaign_mars2019v8.pdf` is the second document matching the task – to find the CESNET classified documents. It contains plans for a marketing campaign and prepared posters (two pages as an example of content are shown below).



CESNET, a. l. e.
Forensic Laboratory
Zikova 4, 160 00
Prague 6, Czech Republic

The PDF file `mars_scheme2017v36.pdf` is the last document matching the task – to find the CESNET classified documents. It contains instructions on how to claim the prize.

Used tools

- `unzip`, `xpdf`, `less`

4 The Timeline

The main story is clear – the suspect Ethan Evilboy is a spy of TENSEC, Ltd. and he was instructed to get information about CESNET's last big project. He gathered some information and tried to exfiltrate it, but has been caught. The timeline of the whole story of data exfiltration attempt can be also created. Just a couple of details have yet to be investigated (these steps are not described in this document).

Date	Time	Event(s)
March 7th	20:11:54	Agent Ethan receives the order from J. to gather information.
March 13th	22:57:05 to 22:57:18	The additional content was prepared and added to the USB drive (to hide the important document).
March 14th	06:34:50	Initial Git commit was realized (*master* branch was created, carrier file has been already finished).
March 14th	07:03:40 to 07:29:32	The leaflets about services were added to the VeraCrypt container.
March 14th	07:22:24	The file *test.txt* was added to the VeraCrypt container.
March 14th	08:44:14 to 08:44:54	The stolen classified documents were gathered (a part of the hidding procedure).
March 14th	08:48:10	The message from Ethan to John was added to the CESNET classified documents.
March 14th	09:05:26	The *transport ready* Git commit was realized (*transport* branch was created, payload has already been implated).
March 14th	10:31:27 to 10:31:28	The VeraCrypt container, the video file containing the password, and the text hint were added to the USB flash drive.
March 14th	11:55:05	Ethan told his coworker to go have lunch without him.
March 14th	11:57:27	CESNET security team was alarmed by Ethan's coworker.
March 14th	12:15:10	Ethan has been stopped by the security team, the USB flash drive has been seized.

5 That's All Folks

This capture the flag contest is over. We hope that you have enjoyed the investigation of an extremely despicable security incident commited on CESNET grounds and that the rewards from our friends of the QuBit Conference made the fifty fastest successful investigators happy.

See you next time!

Team of CESNET forensic laboratory