INSTITUTO SUPERIOR TÉCNICO

Departamento de Engenharia Informática

Forensics Cyber Security

MEIC / METI 2018-2019 – 1st Semester

Digital Forensics Report

Group 13 – 83531 – Miguel Belém, 83567 – Tiago Gonçalves, 83576 – Vítor Nunes

1 Objectives of the investigation

The purpose of this investigation is to find evidences of industrial espionage.

2 Artifacts for analysis

We received a copy of John's flash drive content, we create a read only copy (`chmod 444 *`) and calculate the md5 sum, which we stored on md5.txt file.

3 Evidence to look for

The team will try to find evidences that support or contradicts the industrial espionage suspect activity.

The first approach will be to look what we can gather from the present files;

The second approach will be focus on what maybe hidden.

4 Examination details

We started by looking at the text-readable files such as: `munich.txt` and `compose.py`. The `file` command tool had been used to confirm that the extensions are along with the content of the file.

Filename	Content Type (using *file* command tool)
cathedral.png	PNG image data, 696 x 462, 8-bit/color RGBA, non-interlaced
compress.py	python 2.7 byte-compiled
munich.txt	UTF-8 Unicode text, with very long lines
okotoberfest.png	PNG image data, 1200 x 524, 8-bit/color RGBA, non-interlaced
online_banking.zip	Zip archive data, at least v2.0 to extract
snow.bmp	PC bitmap, Windows 3.x format, 448 x 336 x 24
street.png	PNG image data, 945 x 630, 8-bit/color RGBA, non-interlaced
wursten.png	PNG image data, 640 x 484, 8-bit/color RGBA, non-interlaced

Table 1: File extension tampering analysis

A close look on **online_banking.zip** reveals that the file is encrypted, nevertheless we can dig into the file names and have a remote idea[1] of what may be archived in this file.

We suspect that the file `munich.txt` may contain some clue about the zip password. Most of the files that are encrypt uses a weak password, so we first approach the solution by brute-forcing the zip archive.

We started to extract the words from file using:

```
egrep -o '\w+' munich.txt | sort -u > words.txt
```

To the first approach, we used the **fcrackzip** tool already present in Kali Distribution:

```
fcrackzip -u -D -p 'words.txt' online_banking.zip
```

This hint leads to an unlocked zip archive, which password is: **`Stadelheim`** `(word present in line 82 of munich.txt)`

File Name	File Type	MD5 Hash
online_banking.docx	Microsoft Word 2007+	b70702822417bd39a7997a0f8c73941f
drone-A.bmp	data	05029f0ae6af62ca3350f5b094584b22

Table 2: Content of zip archive

The file `online_banking.docx` contains the following text:

> Password: 51782

Artifact 1: Content of online_banking.docx Word Document

We suspect that this password may unlock some encrypted file.

The **`drone-A.bmp`** is suspicious, the extensions don't match with the content, this may be the result of:

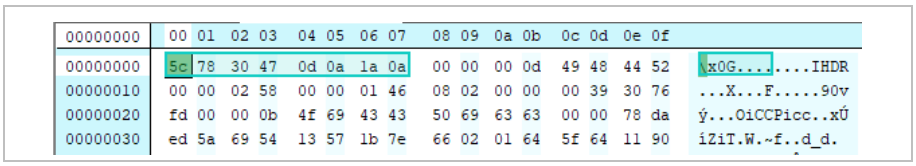
- Someone who tries to hide the real extension of the file;
- The file may be corrupted, in this specific case, the header of the file.

The first approach was check for the strings inside `drone-A.bmp`, which result in the following content:

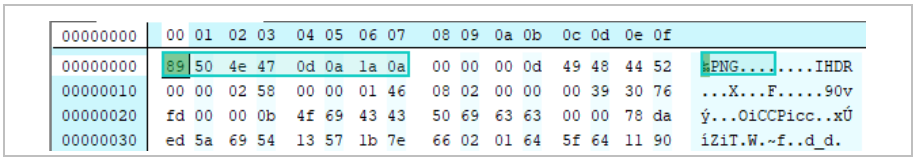
```
%tEXtdate:create
2014-12-09T14:18:22+01:00
%tEXtdate:modify
2014-12-09T14:18:22+01:00y
IEND
```

Table 3: String of suspicious file drone-A.bmp

This file is very similar to PNG files, because of a bunch of chucks (IHDR, IDATx, etc…) we found when dig in using a hex editor. The team decided to try to change the header to match a PNG file and see if we can recover the file. *(The magic number of a PNG file is: 89 50 4e 47 0d 0a 1a 0a)*



[1] Just because a file has certain kind of extension doesn't mean the content match it. Only a content file investigation will clarify this (when we extract the files).



Evidence 1: The original drone-A.bmp (above) and the recovered one (bellow)

We recovered the image successfully, which appears to be a drone plan.

On a first look the Word Document appears to be a regular file containing only text about some password.

In fact, it may contain hidden data. The team decides to unzip the word document and search for relevant evidence, and we didn't find any relevant information inside the file.

A close look to the `compress.py`, reveals strange behavior, so the team decide to give to the application a test image and a test file. The output appears to be a regular image, exactly the same look.

The python byte compiled file revels the following information using the strings command:

```
Ciber Securanca Forense - Instituto Superior Tecnico / Universidade LisboasL
LSB steganography tool: hide files within least significant bits of images.
Usage:s)
  %s <img_file> <payload_file> [password]s0
```

Artifact 2: Compress.py usage help

The purpose of the script is to hide files in pictures, so we decided to give it a try and see what would happen if we provide an image to this mysterious script.

The team suspects that the password found at the `online_banking` file may be the 3rd argument of this script.

File name	Size
test_image.png	614 191 bytes
test_image.png-stego.png	614 198 bytes

Table 4: Differences using compress.py

The images appear, on a first look, exactly the same, except the file size. The output image is slightly bigger.

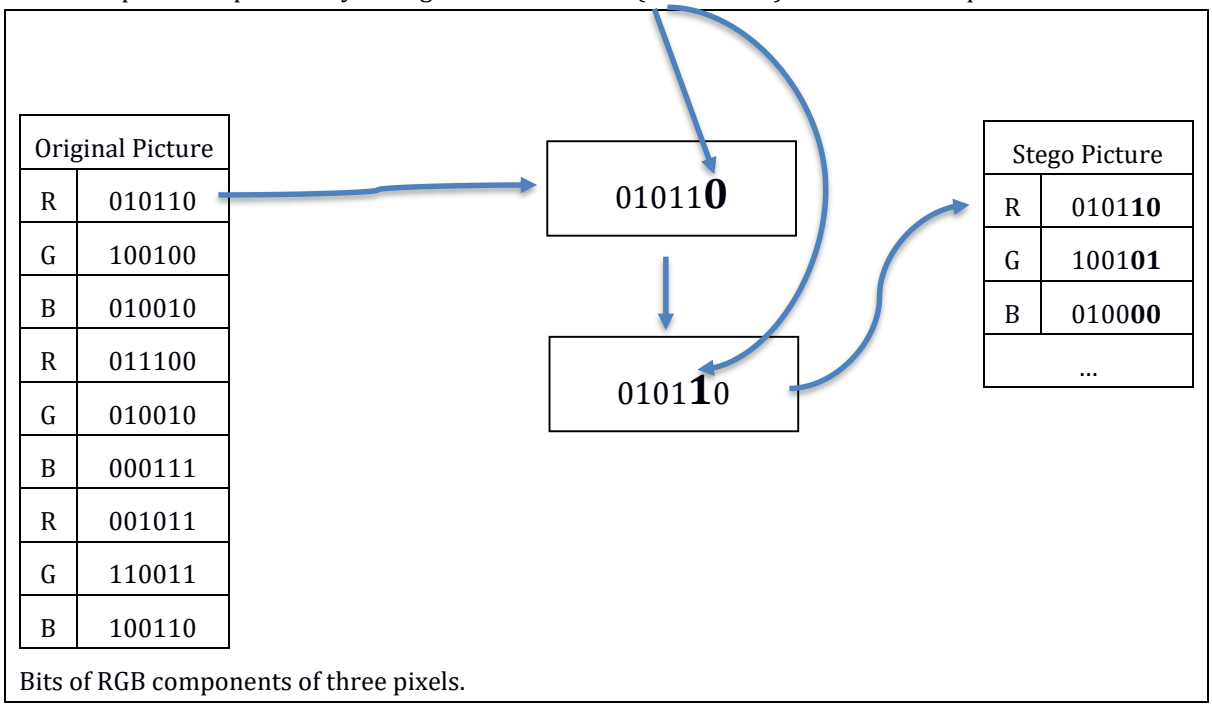
The team decided to decompile the python file using the tool **`uncompyle6`**[2] in order to retrieve more information about the program.

We successfully get the source code and the team put efforts to try to understand how this compress process works. After several hours the team discover that:

- The payload data is hidden using the **first** and **second** latest significant bytes (LSB) of a pixel data. (See the example bellow);
- The password argument is used to define the offset and to select in which pixel we start injecting the message.

[2] Available at: https://pypi.org/project/uncompyle6/2.2.0/

We will explain this process by hiding one character 'a' (01100001) inside one PNG picture.



Example 1: Compress process

A closer look to strings inside snow.bmp file, revels the following information:

hCIch sende Ihnen f nf Dateien: (1) Drohne A Pl ne, (2) Drohne B Pl ne, (3) technische Spezifikationen, (4) Passw rter von DroneX Dateiservern	I'm sending you five files: (1) Drone APl ne, (2) drone B pl ne, (3) technical specifications, (4) passw of DroneX file servers

Evidence 2: German information found in snow.bmp tail, and English translation (right).

In order to retrieve the hidden information, the team create a script to extract the LSB bits from the stego pictures. *(This script can be found in zip archive with name: **script/decompress.py**)*

The decompress.py script can be used using the following command in bash:

```
$ python decompress.py input.png output-file [password]
```

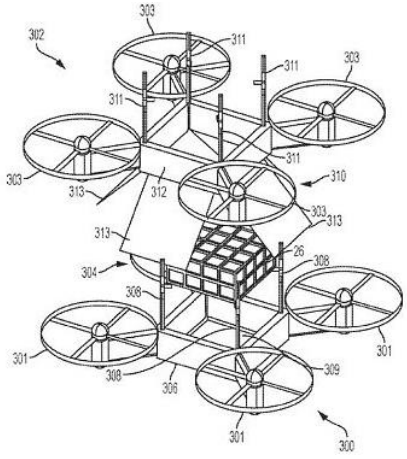
Note: Run this script with python 2 and **not** with python 3 versions.

Using the python script in all files, we found the following evidence in wursten.png file without any password:

```
ACCESS PASSWORDS FOR DRONEX SERVERS
SERVER 1: Sr!_01llxt
SERVER 2: p_GEtKl4dA
```

Evidence 3: Hidden data found in wursten.png

And we get a picture of, what appears to be, the drone's technical specs inside the `oktoberfest.png`:



Evidence 4: Drone's technical specs

And the following picture hidden inside `street.png`:

Artifact 3: A suspicious picture of a castle

At this point, the team asks itself: "What is the purpose of hiding a picture of a castle inside another picture?". This was suspicious enough so let's run the script again on this castle and see what is inside, behind those walls.

```
OPERATING SPECIFICATIONS
DRONE A
Operational Altitude: 5,000 – 12,000ft WGS, no lift loss
(FAA limited at this time)
Sensor agnostic system
…
```

Evidence 5: Drone's technical specs (partial file)

We found what appears to be the drone's technical specifications which can be viewed fully in `specs.txt`.

Evidence 6 is now confirmed, all of 4 files were found.

5 Analysis results

The pen was found inside John's case, nevertheless there was no evidence that support that he is the owner. Only a further investigation to his computer will determine this question, which is to look for some evidence that this flash drive (with this serial number) was connected to his computer.

The team **found five artifacts**, **two** drone **pictures** (specifications for both models), **one text document** describing drone's specifications and **two plain text messages:** Passwords for the Drone-X servers and a message that suggests someone (the perpetrator) was sending multiple objects that contain information about the drones (all the other artifacts that were found).

File Name	MD5 Hash
drone-A.png	d99f500968d444b5e0a1c9fd1dd69274
specs.txt	3ba4ca7f05bbf65083360e455fa8ea8a
drone_schema.png	cbe4c039f3fa2b312bb95a0964ffba4d
passwords.txt	3cb3f3162e4cf990168d904d3bb300b9
message.txt	7665a2071050bda07960e58c06e79705
castle.png	d770b66b4f5833b0be194362f440e494
script/source_compress.py	39e01a20232d92376496b224382368a1

Table 5: List of recovered files from John's flash drive

The team could not understand the purpose of the password stored on `online_banking` Word Document. We tried using it in `decompress` script but it provides no relevant evidence.

6 Conclusions

We can conclude that someone sent critical information about Drone-X's new drone models putting effort on hiding the track (e.g. files inside pictures).

Lisbon, October 21st 2018, Miguel Belém, Tiago Gonçalves, Vítor Nunes.