



**UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH**

**Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona**

SCRIPT-BASED TOOL FOR REMOTE DIGITAL FORENSIC ANALYSIS

A Master's Thesis

**Submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de
Barcelona**

Universitat Politècnica de Catalunya

by

Jordi Oliveras Boada

**In partial fulfillment
of the requirements for the degree of
MASTER IN TELECOMMUNICATIONS ENGINEERING**

Advisor: Josep Pegueroles

Barcelona, May 2017

Title of the thesis: Script-based tool for remote digital forensic analysis

Author: Jordi Oliveras Boada

Advisor: Josep Peguerols

Abstract

In this Master Thesis it has been working in digital forensics and its aim is to create a tool with a GUI that will allow people that are not specialist in digital forensics to perform an analysis at the same time that preserve the chain of custody of the artifacts being analyzed. This is done to make able digital forensics investigations at any part of the world where there it may not be any specialist.

The tools that have been scripted and analyzed in this project are the following: binary copy, file carving, timeline, OCR and eDiscovery. There are already some tools with GUI but our aim is to create a new one so easy to use that if in one country there is no specialist, the analysis can also be performed.

Acknowledgements

I would like to thank Josep Peguerols to help me when making the project and guiding and orientating me in the way it would be better to make this project and the contents and tools it should contain. I would also like to thank Arnau Estabanell to help me in developing one of the tools of this project.

Revision history and approval record

Revision	Date	Purpose
0	19/05/2017	Document creation
1	22/05/2017	Document revision

Written by:		Reviewed and approved by:	
Date	19/05/2017	Date	22/05/2017
Name	Jordi Oliveras	Name	Josep Peguerols
Position	Project Author	Position	Project Supervisor

Table of contents

Abstract.....	4
Acknowledgements.....	5
Revision history and approval record.....	6
Table of contents	7
List of Figures	9
List of Tables.....	11
1. Introduction	12
1.1. Requirements and specifications.....	12
1.2. Purpose	12
1.3. Methods.....	13
1.4. Work plan	13
1.5. Deviations from the initial plan.....	14
2. State of the art of the technology used or applied in this thesis.....	15
2.1. Current forensics procedure	15
2.2. Forensic tools with GUI	16
3. Project development	17
3.1. Binary copy.....	17
3.2. Recover (File Carving).....	24
3.3. Timeline	28
3.4. OCR.....	34
3.5. eDiscovery.....	42
4. Results	48
5. Budget.....	49
6. Conclusions and future development.....	51
Bibliography	53
Appendices	54
Appendix 1. Binary copy python code to work from the terminal	54
Appendix 2. Binary copy python code to work with the GUI.....	54
Appendix 3. Binary copy GUI glade code.....	56
Appendix 4. Binary copy desktop code	60
Appendix 5. File carving python code to work from the terminal.....	60

Appendix 6. File carving python code to work with the GUI	61
Appendix 7. File carving GUI glade code	64
Appendix 8. File carving desktop code.....	67
Appendix 9. Timeline python code to work from the terminal.....	68
Appendix 10. Timeline python code to work with the GUI.....	70
Appendix 11. Timeline GUI glade code	76
Appendix 12. Timeline desktop code	79
Appendix 13. OCR python code to work from the terminal	79
Appendix 14. OCR python code to work with the GUI	82
Appendix 15. OCR GUI glade code	87
Appendix 16. OCR desktop code	92
Appendix 17. eDiscovery starting window python code	93
Appendix 18. eDiscovery grep python code.....	94
Appendix 19. eDiscovery gawk python code.....	99
Appendix 20. eDiscovery starting window glade code	105
Appendix 21. eDiscovery operating window glade code	108
Appendix 22. eDiscovery desktop code	116
Glossary	117

List of Figures

Image 1: Gantt diagram of the project	14
Image 2: "dconf-editor" window.....	18
Image 3: Disabling the auto-mount of the USB devices using "dconf-editor".	18
Image 4: Example of how to install Python.	19
Image 5: Extract files from the compressed folder using the terminal.	19
Image 6: Look the full path to the directory of the scripts with the command "pwd"	19
Image 7: Opening Binary_Copy.desktop from gedit.	20
Image 8: Modify the document "Binary_Copy.desktop" with the new path.....	20
Image 9: Getting the name of the device.	21
Image 10: Double click to "Binary Copy" to run the script with the GUI.....	22
Image 11: Window to choose the path to the folder.....	22
Image 12: The text entry is filled automatically when clicking "Select".	23
Image 13: Sudo password asked at the terminal to allow the binary copy.	23
Image 14: Locate the folder with the scripts.	23
Image 15: Running the script from the terminal.	24
Image 16: Example of how to install Python.	25
Image 17: Extract files from the compressed folder using the terminal.	25
Image 18: Look the full path to the directory of the scripts with the command "pwd".	25
Image 19: Modify the document "Recover.desktop" with the location of the script.....	26
Image 20: Running the recover tool with the GUI just clicking on "Recover" icon.	26
Image 21: Window to choose the path to the binary copy.	27
Image 22: The text entry is filled automatically when clicking "Select".	27
Image 23: Locating the scripts with the terminal.....	28
Image 24: Using the recover tool from the terminal.	28
Image 25: Photorec performing the file carving from the terminal.	28
Image 26: Example of how to install Python.	29
Image 27: Extract files from the compressed folder using the terminal.	30
Image 28: Look the full path to the directory of the scripts with the command "pwd".	30
Image 29: Opening Timeline.desktop from gedit.	31
Image 30: Modify the "Timeline.desktop" with the new path.....	31
Image 31: Double click to "Timeline" to run the script with the GUI.....	32

Image 32: Window to choose the path to the folder.....	32
Image 33: Text entry is filled when clicking "Select".	33
Image 34: Document with the results of the analysis.....	33
Image 35: Owner and grup numbers.	33
Image 36: Locate the folder with the scripts.	34
Image 37: Running the script from the terminal.	34
Image 38: Example of how to install Python.	35
Image 39: Extract files from the compressed folder using the terminal.	35
Image 40: Look the full path to the directory of the scripts with the command "pwd".	36
Image 41: Opening OCR.desktop from gedit.	36
Image 42: Modify the document "OCR.desktop" with the new path.....	37
Image 43: Double click to "OCR" to run the script with the GUI.	37
Image 44: Window to choose the path to the folder.....	38
Image 45: The text entry is filled automatically when clicking "Select".	38
Image 46: Preparing the folder names by clicking on "PREPARE".	39
Image 47: Dividing the pdf files into pages and saving the results into "all_pdf_text" by clicking on "DIVIDE".....	39
Image 48: Performing the OCR.	40
Image 49: Locate the folder with the scripts.	40
Image 50: Running the script from the terminal.	41
Image 51: Example of how to install Python.	42
Image 52: Extract files from the compressed folder using the terminal.	43
Image 53: Look the full path to the directory of the scripts with the command "pwd"	43
Image 54: Modify the document "eDiscovery.desktop" with the location of the script.	44
Image 55: Running the search tool with the GUI just clicking on "eDiscovery" icon.	44
Image 56: Main window of the search tool once the search command has been selected.	45
Image 57: Window to choose the path to the folder containing the txt files to analyze.....	46
Image 58: The text entry is filled automatically when clicking "Select".	46
Image 59: Typing the word to search (in this example "computer") and pressing the filter button to start the search.	46
Image 60: Results shown after the search has finished.	47
Image 61: Document with the results saved.	47

List of Tables

Table 1: Time required for the tools to analyse 1GB of information.....	48
Table 2: Grep and gawk time searching into 100 documents.....	48
Table 3: Cost of the project.	49
Table 4: Cost of the use of the tool.	50

1. Introduction

Digital forensics is a science that is in charge of investigating suspicious digital artifacts in order to obtain some evidences in crimes that may have relation with computers or other digital devices.

The aim of this Master Thesis is to study the different tools required when performing a digital forensic analysis. In this project, the different requirements are explained and analyzed.

Apart from that, this project also includes the elaboration of some scripts providing a GUI (Graphical User Interface) to a user without much knowledge about digital forensics. The aim of that part is to allow that user to perform nearly automatically a digital forensic analysis without having to have the knowledge to do so.

This can be useful when the case we are working is located in another country. In order to avoid having to send a forensic specialist to the other country, since the evidences cannot be moved from the country, it can be used this tool so a non-specialist person can perform the analysis as well.

The first idea of this project was to continue a project called “Análisis de alternativas, desarrollo y puesta en marcha de una plataforma para análisis forense digital remoto” by Jordi Blanco. However, due to the huge amount of work required to make a whole platform working in those specifications, this project will just be focused on the scripts required to perform the analysis separately, which will allow to perform the digital forensic analysis but without the main structure estated in Jordi Blanco project.

1.1. Requirements and specifications

As it has been mentioned before, the requirements are simple: there must be some scripts with some kind of GUI that must allow to perform a digital forensic analysis.

The technical specifications are the same that would be for a normal digital forensic analysis: some tools that must allow to perform the actions required for the investigation without breaking the chain of custody of the artifact, since this is crucial in order to be presented as an evidence in a jury.

Apart from that technical requirements, there is also another requirement: this scripts must be user friendly and quite easy to use. The reason of that is to allow a person that has not much knowledge of digital forensics to be able to perform the analysis.

1.2. Purpose

As stated before, this Master Thesis will analyses the current requirements (talking about tools) required to perform a digital forensic analysis, and to create some scripts in order to provide a user friendly GUI to perform the analysis.

The purpose of doing so is to create a platform based on the project of Jordi Blanco, that would allow to perform a remote digital forensic analysis. The reason of those requirements is that the person that may be in charge of performing the analysis will not be a forensic specialist.

1.3. Methods

According to the beginning of the introduction, this project was meant to be the continuation of another project that already researched about all the problems encountered in remote digital forensic analysis.

To do so, in this project it has been done some part of research job in order to get the knowledge of the current requirements when performing a digital forensic analysis. Apart from that, also it has been searched other tools that allow to perform the same actions.

The software that has been used to program the scripts is mainly python, combined with glade tool to make the GUI easier to be created. All the scripts are prepared to be run in a Linux Operative System. Some of the scripts are based on other tools already existing whereas some of them are just providing a more friendly interface to simple Linux commands.

1.4. Work plan

The work plan of this project is structured in the following way: first of all, the first important task is to read the Jordi Blanco project and do some research about both digital forensics and also about scripting.

Once done that, the next important tasks will be working with the 5 tools that are included in this project. These tools are: binary copy, recover (file carving), timeline, OCR and eDiscovery. For each of the tool there are three important tasks to be done with: investigate and research about the use of the tool and the current options to perform that action, program the script according to the requirements of the tool and at last test the proper performance of the tool.

All those tasks are done for each tool, but tool by tool, first finish a tool before starting with the next one. The time required for each tool is around a month, however there may be some tools that are easier and will take less time to be finished whereas some others may last more than 2 month.

Apart from all those tasks, there is also the usual task on any project of documenting it and analyzing the results. This task is done during the whole project. In the pictures below you can find the gantt diagram of all the task of the project:

2. State of the art of the technology used or applied in this thesis

In this part of the Master Thesis we are going to talk about the current state of the art of the digital forensics. In this point we will focus specially on the already existing tools that nowadays are used to perform the analysis.

To make a quick remind of what has already been commented in the introduction, the digital forensics tools that are developed in this project and that will be commented here in this point are the binary copy, file carving, timeline, OCR and eDiscovery.

Apart from talking about the different tools we also will talk about tools that already exists and that also have a GUI, since creating tools like this is one of the main goals of the project.

2.1. Current forensics procedure

When performing a digital forensic analysis, doing a binary copy is one of the most important things to do during an investigation. It is required to get the information but without accessing to it (like putting the artifact into a plastic-bag to avoid leaving fingerprints but in a computer way). To do so what it has to be done is first of all to disable the automatic mount of the USB devices, and afterwards connect the device without mounting it and make a copy of all the 0 and 1 that the device contains. The command used to perform this action is "dd" which allow to perform binary copies. To proof that the binary copy is identically than the original artifact, it can be computed the hash of both things and check that it is the same.

Recovering deleted files is an essential part of the forensics analysis. This is called normally file carving. Once the binary copy is done, it can be used to recover files that has been deleted from there. There are different tools to perform this part of the forensics analysis, but in this project it will be used the Linux tool "photorec". This tool has its own interface in order to operate with it, but it can be used as well to be used in a script if it is provided with the proper parameters.

To get the timeline of the files in a specific directory is another tool that is sometimes used by forensics. There are already some scripts that does this functionality such as Quickfish. It consists on creating a document which contains the name of all the files located inside a certain directory and some useful information that can be used to filter the documents trying to encounter evidences. This information includes size, format of the file, day of creation, day of the last modification, user that created the document, user that has done the last modification...

Some of the problems that may be encountered when performing the investigation are that not all the documents will be text documents. There may be pictures, videos, PDFs among other different format files. Since the amount of information to analyze tends to be quite big, it would require to find some way to be able to analyze all this data. To do so it can be performed a massive OCR (Optical Character Recognition) To extract the text that appears I all those documents in a txt format to be able to perform searches on it. There are lots of tools to do so, since open source tools such as Tesseract (in that case it should be combined with other Linux tools like Imagemagick or Unpaper to prepare the

image before the OCR) to payment tools like Addove or Abby (this ones there is no need to prepare the image).

Another important tool really useful is to be able to find from all the documents, the ones that contain certain key words for the investigation. In digital forensics, this is called eDiscovery. To do so, there are some commands but in this project it will be used the Linux command "grep". Another tool that has being used in this project is "gawk". This command is quite slower than "grep" but it allows the user to do logical searches (looking for more that one word and deciding if both words have to be contained in the same document or if any of those appear decide to analyze the document).

2.2. Forensic tools with GUI

As it has been commented in other points of the project, one of the main goals of this project is to create some scripts with a GUI, in order to help to make this type of investigations available at any part of the world even without a forensic specialist there. There are different tools that already do some of the parts with GUI, the most used ones are Autopsy and EnCase.

Autopsy is one of the most known forensic tools. It has a GUI that is run from an HTML browser, and it is really intuitive to use. This tool allows you doing most of the tools that are treated on this project so it is important to take a look into it. It allows to make timeline analyses, keyword search (eDiscovery), file carving, and also some other tools that has not been scripted in this project, such as hash filtering or diferent tools to extract information about the properties of multimedia files.

However, even this tool is really intuitive to use, it does not have an OCR in order to convert images and pdfs into txt files before performing the search of keywords. Another problem is that it does not have the possibility to perform the binary copy. Since this is a really delicate part of the process since it is vital to preserve the chain of custody, if this part has to be done manually, this must be done by a forensic specialist.

Another tool that should be mentioned is EnCase. This tool also have its own GUI and allows the user to perform some basic forensic tasks. It focuses on the investigation of mobile phones. It helps to acquire the information from any phone of a wide diversity of OS. It has also tools such as eDiscovery to analyze the results.

The problem with this tool is also the same one with Autopsy. It is a really powerful tool if the user is a forensic specialist but it is delicate if someone who is not tries to use it since it may break the chain of custody. One of the aims of this project is preparing the tools in order that the user will preserve untouched the chain of custody.

3. Project development

In this point of the project, there are stated all the different 5 tools that have been analyzed and scripted in this project. This tools are the following: binary copy, recover (file carving), timeline, OCR and eDiscovery.

All of those are really useful tools in digital forensics. In this point of the project, they are explained a little bit, explained the main tools that normally are used to perform those actions and it is explained as well how the scripts work. It also contains an administrator guide (containing how to prepare the tool in order to be ready to use) and a user guide (containing instructions of how to use the tool). There are both guides for each of the 5 tools.

3.1. Binary copy

In forensics, when the police arrives at the scene of the crime, it puts the evidences into a plastic bag to avoid leaving fingerprints. In digital forensics it has to be done the same. In order to do so, there is an option that consists in not to mount the evidence but connect to the computer and perform a binary copy of it.

The binary copy consists on copying the 1 and 0 that there are inside the evidence, that from now we will call it artifact. The aim of doing a binary copy of the artifact is to be able to analyze its content but without modifying anything from the original content. Doing so it is crucial in digital forensics, since this artifact will not be able to be presented as an evidence if the chain of custody is not maintained untouched.

To make easy this process we have created a tool which will allow performing a binary copy of an artifact automatically. To use this tool it is important to have the computer that is going to be used configured in a way that has disabled the auto-mount of the USB devices before connecting the artifact that is going to be analyzed.

This tool what mainly does is to compute the hash of the device, it makes a binary copy of the device (which will be saved into the path provided by the user of the tool) and it makes a hash of the copy (to ensure that is the same and that nothing has been modified). There are several tools to perform those actions but in this scripts it is used the Linux commands "md5sum" to compute the hash and "dd" to perform the binary copy.

If you are interested in the code of the files to run this tool, in the appendix from 1 to 4 you can found all the codes of the scripts and the graphical interface to run this scripts.

ADMINISTRATOR GUIDE:

When preparing a computer to use this tool, first of all it is interesting to look at the system requirements. This tool, as well as all the others, is prepared to be run in a Linux Operative System, however, if the computer has another OS, it can be run as well using a virtual machine like VirtualBox with a Linux system installed.

This Operative system must have the USB auto-mount disabled. In order to disable the USB auto-mount, it is necessary to have a tool installed called "dconf-editor". This tool can be installed using the Linux command "sudo apt-get install dconf-editor" in a terminal. Once this is done this program has to be run by typing in the terminal "dconf-editor". This will show up the following window:

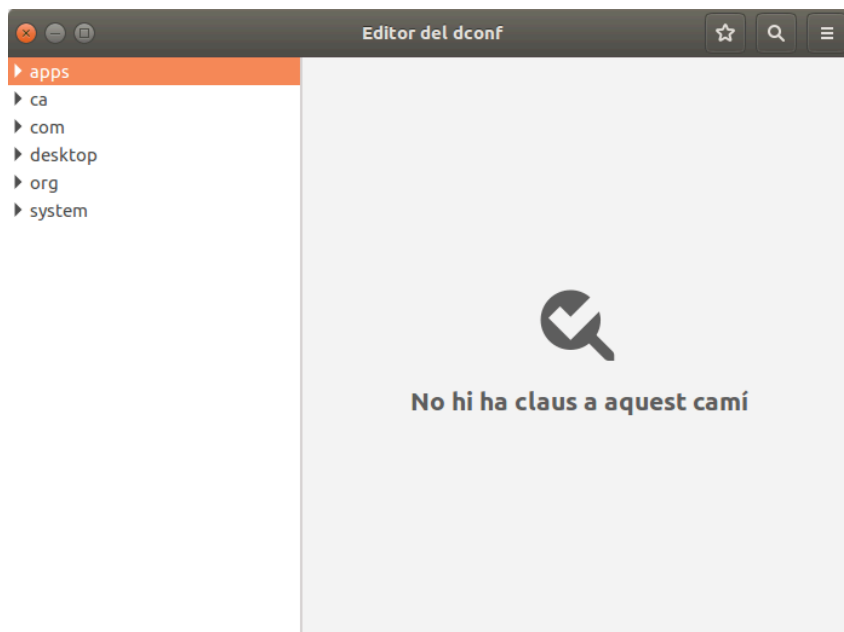


Image 2: "dconf-editor" window

Now you have to modify some parameters of it. To do so you have to click on "org", then on "gnome", then on "desktop" and then on "media-handling". Once there the parameters "automount" and "automount-open" must be set to false. This will disable the auto-mount of the USB devices.

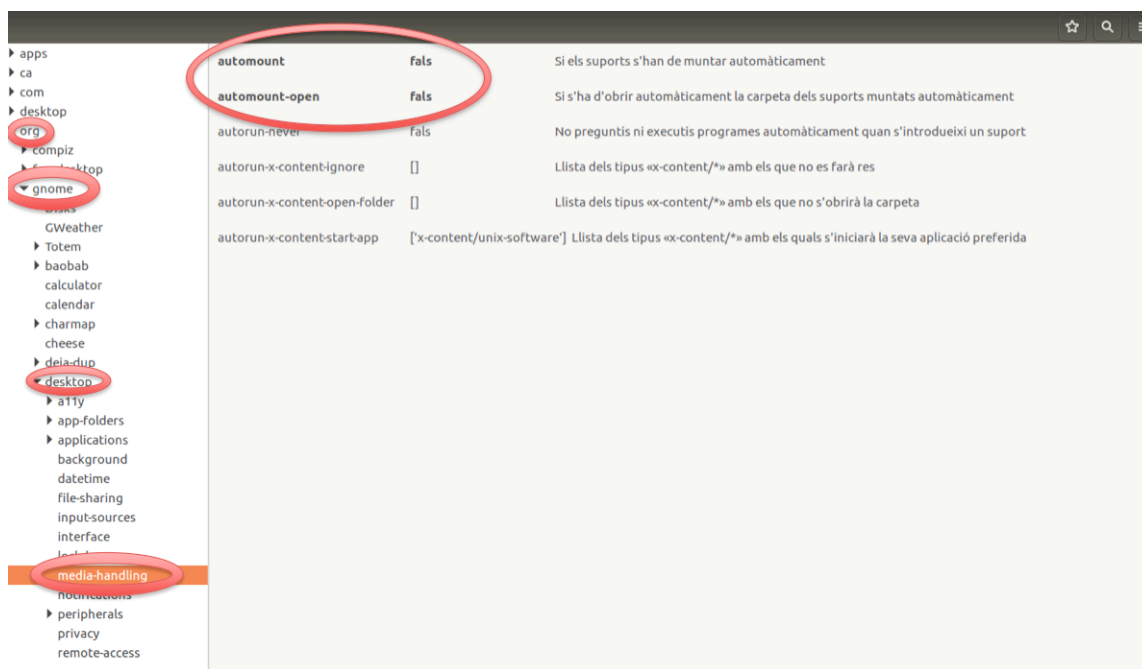


Image 3: Disabling the auto-mount of the USB devices using "dconf-editor".

Take into account that if the tool is run in a virtual machine, both systems have to be disabled the USB auto-mount.

Apart from the operative system with the USB auto-mount disabled, this will have to have installed already some programs. In the case of this tool, the requirements are the

following: Python with its corresponding libraries (logging, time, sys, os, stat, time, hashlib, argparse, csv, gi with Gtk version 3). It is also recommended to have glade installed. All this tools can be installed using the command “sudo apt-get install xxxxxx” where xxxxxx is the name of the program required.

```
jordi@jordi-VirtualBox:~/Escriptori$ sudo apt-get install python
[sudo] contrasenya per a jordi:
S'està llegint la llista de paquets... Fet
S'està construint l'arbre de dependències
S'està llegint la informació de l'estat... Fet
```

Image 4: Example of how to install Python.

Once all of this is ready, the tool has to be prepared in order to be used. First of all, it is needed to get the compressed folder with all the scripts required to use the tool and unzip it into a known location. This can be done manually just by the default program on the operative system or to use the command “tar -xvf BINARY_COPY.tar.gz”.

```
jordi@jordi-VirtualBox:~/Documents$ ls
BINARY_COPY.tar.gz
jordi@jordi-VirtualBox:~/Documents$ tar -xvf BINARY_COPY.tar.gz
BINARY_COPY/Binary_Copy_gui.py
BINARY_COPY/Binary_Copy.desktop
BINARY_COPY/Binary_Copy.glade
BINARY_COPY/
BINARY_COPY/Binary_Copy_Terminal.py
jordi@jordi-VirtualBox:~/Documents$
```

Image 5: Extract files from the compressed folder using the terminal.

Make sure to know the exact path to the location where it has been extracted the documents. In case it is not known, using the terminal command “cd” to enter to the proper folders until reaching the folder where the scripts have been extracted, and once there use the command “pwd” to look for the current directory.

```
jordi@jordi-VirtualBox: ~/Documents/BINARY_COPY
jordi@jordi-VirtualBox:~$ ls
Baixades  Escriptori  glade  Música  Públic  test1.tiff
Documents examples.desktop Imatges Plantilles res.txt Vídeos
jordi@jordi-VirtualBox:~$ cd Documents/
jordi@jordi-VirtualBox:~/Documents$ ls
BINARY_COPY BINARY_COPY.tar.gz
jordi@jordi-VirtualBox:~/Documents$ cd BINARY_COPY/
jordi@jordi-VirtualBox:~/Documents/BINARY_COPY$ ls
Binary_Copy.desktop Binary_Copy_gui.py
Binary_Copy.glade   Binary_Copy_Terminal.py
jordi@jordi-VirtualBox:~/Documents/BINARY_COPY$ pwd
/home/jordi/Documents/BINARY_COPY
jordi@jordi-VirtualBox:~/Documents/BINARY_COPY$
```

Image 6: Look the full path to the directory of the scripts with the command “pwd”

This path it is necessary to make the script runnable from anywhere. To do so, it is necessary a text editor or if it is preferred to read it from the terminal. Open a text editor like gedit and once opened, oped the file called “Binary_Copy.desktop”. It is essential to do it that way since this file cannot be opened by double clicking on it (it is prepared to execute the scripts when clicking on it).

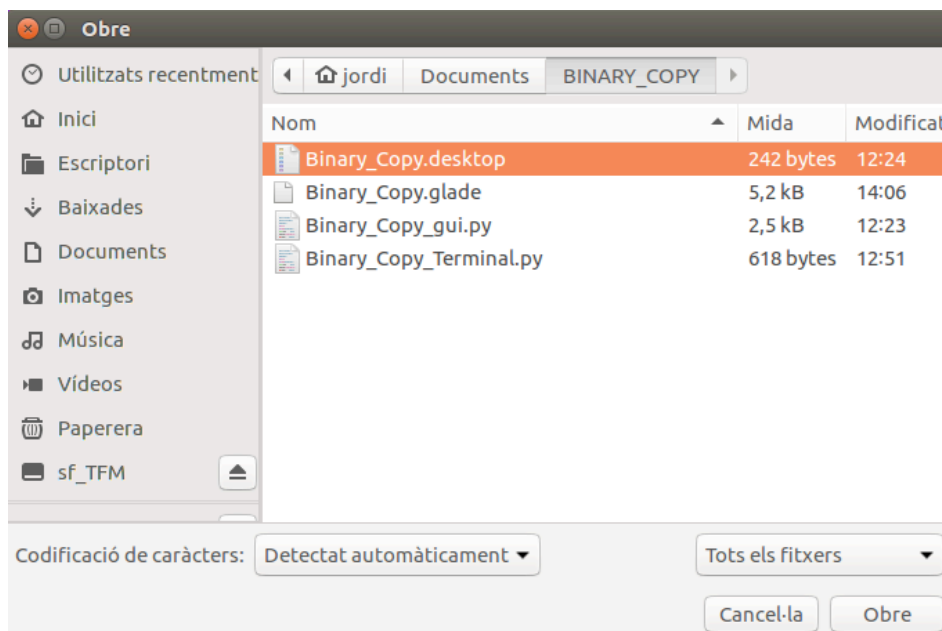


Image 7: Opening Binary_Copy.desktop from gedit.

Now, this document has to be modified using the proper path of the location of the scripts. The lines that have to be replaced are the ones that start with “Exec=” and “Path=”, and the path that there is afterwards has to be replaced with the new location obtained using “pwd”. It is important not to modify the name of the script at the end of the line exec. Once done that just save the file with the same name.

```
[Desktop Entry]
Name=Binary Copy
Exec=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/BINARY_COPY/Binary_Copy_gui.py
Path=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/BINARY_COPY/
Terminal=true
Type=Application
```

```
[Desktop Entry]
Name=Binary Copy
Exec=/home/jordi/Documents/BINARY_COPY/Binary_Copy_gui.py
Path=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/BINARY_COPY/
Terminal=true
Type=Application
```

```
[Desktop Entry]
Name=Binary Copy
Exec=/home/jordi/Documents/BINARY_COPY/Binary_Copy_gui.py
Path=/home/jordi/Documents/BINARY_COPY/
Terminal=true
Type=Application
```

Image 8: Modify the document "Binary_Copy.desktop" with the new path.

Once all of this is done, just move or copy the file “Binary_Copy.desktop” to anywhere in order to execute the scripts with the GUI (Graphical User Interface). Take into account that if this is moved to the account of another user, make sure it have the rights to execute scripts, otherwise it will not work. Consider as well that the icon of this file won’t show the entire name, just “Binary Copy”.

It is also important to look for the name of the USB in the folder “/dev” and set as default the name assigned by the computer. This can be done just by typing into a terminal “cd /dev”, then “ls”, then plugging the USB and typing “ls” again. The name that has appeared is the name you should provide to the person that is going to use the tool. Another way of

doing so is typing “lsblk” before and after plugging the USB and the one that has been added is the one that has to be provided to the user.

```

jordi@jordi-VirtualBox: /dev$ lsblk
cpu_dma_latency net          tty14  tty44  ttyS15  vcs2
cuse             network_latency tty15  tty45  ttyS16  vcs3
disk            network_throughput tty16  tty46  ttyS17  vcs4
dri             null          tty17  tty47  ttyS18  vcs5
dvd            port          tty18  tty48  ttyS19  vcs6
ecryptfs        ppp           tty19  tty49  ttyS2   vcs7
fb0            psaux        tty2   tty5   ttyS20  vcsa
fd             ptmx         tty20  tty50  ttyS21  vcsa1
full           pts          tty21  tty51  ttyS22  vcsa2
fuse           random       tty22  tty52  ttyS23  vcsa3
hidraw0        rfkill       tty23  tty53  ttyS24  vcsa4
hpet           rtc          tty24  tty54  ttyS25  vcsa5
hugepages      rtc0         tty25  tty55  ttyS26  vcsa6
hwrng          sda          tty26  tty56  ttyS27  vcsa7
initctl        sda1         tty27  tty57  ttyS28  vflo
input          sda2         tty28  tty58  ttyS29  vga_arbiter
kmsg           sda5         tty29  tty59  ttyS3   vhci
lightnvm       sg0          tty3   tty6   ttyS30  vhost-net
log            sg1          tty30  tty60  ttyS31  zero
loop0          snapshot     tty31  tty61  ttyS4   vcsa
loop1          snd          tty32  tty62  ttyS5   vcsa
loop2          sr0          tty33  tty63  ttyS6   vcsa
loop3          sdb          tty34  tty7   ttyS7   vcsa

jordi@jordi-VirtualBox: /dev$ lsblk
cpu_dma_latency net          tty11  tty41  ttyS12  vboxuser
cuse             network_latency tty12  tty42  ttyS13  vcs
disk            network_throughput tty13  tty43  ttyS14  vcs1
dri             null          tty14  tty44  ttyS15  vcs2
dvd            port          tty15  tty45  ttyS16  vcs3
ecryptfs        ppp           tty16  tty46  ttyS17  vcs4
fb0            psaux        tty17  tty47  ttyS18  vcs5
fd             ptmx         tty18  tty48  ttyS19  vcs6
full           pts          tty19  tty49  ttyS2   vcs7
fuse           random       tty2   tty5   ttyS20  vcsa
hidraw0        rfkill       tty20  tty50  ttyS21  vcsa1
hpet           rtc          tty21  tty51  ttyS22  vcsa2
hugepages      rtc0         tty22  tty52  ttyS23  vcsa3
hwrng          sda          tty23  tty53  ttyS24  vcsa4
initctl        sda1         tty24  tty54  ttyS25  vcsa5
input          sda2         tty25  tty55  ttyS26  vcsa6
kmsg           sda5         tty26  tty56  ttyS27  vcsa7
lightnvm       sdb          tty27  tty57  ttyS28  vflo
log            sdb1         tty28  tty58  ttyS29  vga_arbiter
loop0          sg0          tty29  tty59  ttyS3   vhci
loop1          sg1          tty3   tty6   ttyS30  vhost-net
loop2          sg2          tty30  tty60  ttyS31  zero
loop3          sr0          tty31  tty61  ttyS4   vcsa

```

Image 9: Getting the name of the device.

USER GUIDE

This tool can be used in two different formats: the first one is with the GUI and can be run just by double-clicking into the icon “Binary Copy”. And the second one is a version that does not need GUI and is completely run using just a terminal.

Using the tool with the GUI:

To run this tool it is really easy and intuitive. First of all make double click on the icon of the binary copy will open the main window of the tool (see the picture below). Apart from the main window it will also open a new terminal. This terminal will be needed later.

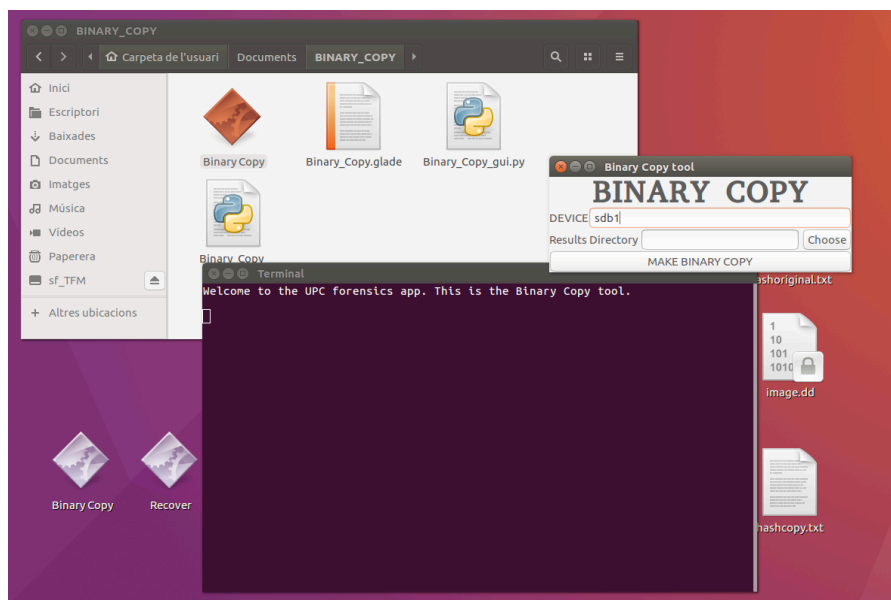


Image 10: Double click to "Binary Copy" to run the script with the GUI.

As it can be seen in the picture, there are two parameters required to perform the analysis. Do not touch the parameter of the field device unless you know exactly what you are doing. The second field is a path where the copy will be saved.

To choose it can be done into two different ways, one is just typing the path on the corresponding text entry (be careful of not making any miss-spell or it will not work) or clicking at the button next to it. When doing this second option it will appear a window where you will be able to navigate through the folders and select the proper one. Clicking "Select" will automatically fill the text entry with the selected path.

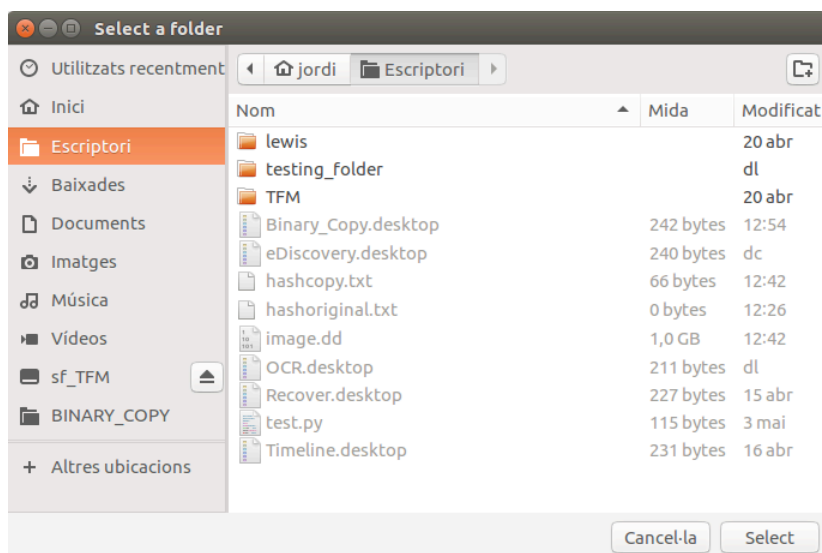


Image 11: Window to choose the path to the folder.



Image 12: The text entry is filled automatically when clicking “Select”.

It is important to take into account that the parameters that the script will use are the ones written there, so if the user selects a folder and then modify its name, the second name will be the one used.

Once it is selected just press the button “MAKE BINARY COPY” in order to start the binary copy. This is the point where the terminal is needed. It will ask the sudo password in order to perform the copy. Just type it on the terminal and press enter.

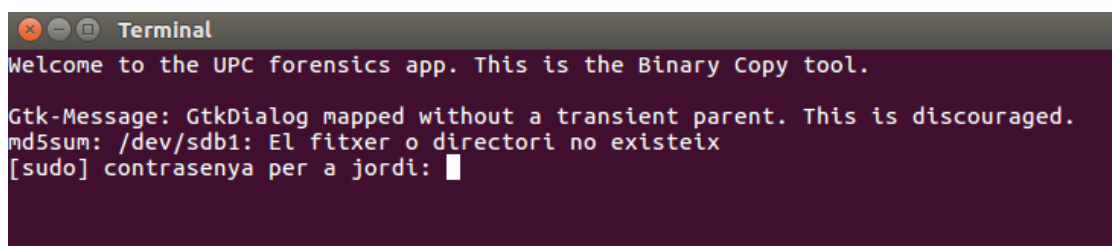


Image 13: Sudo password asked at the terminal to allow the binary copy.

Using the tool with the Terminal:

Running this tool from the terminal is quite simple if you are used to work with the terminal. What you first need to do is to locate the folder where the scripts are located as it can be seen in the picture below:

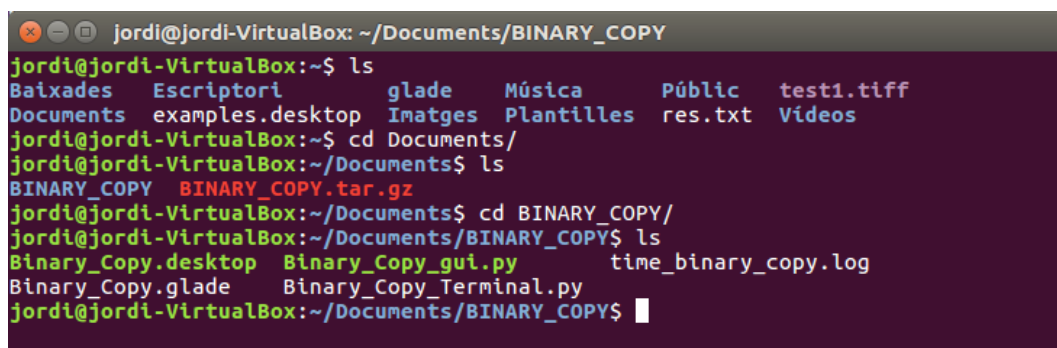


Image 14: Locate the folder with the scripts.

Once there just type the command “python Binary_Copy_Terminal.py”. This will automatically run the script at the terminal. To use this tool from here, you just need to answer all the parameters that the script asks you. And after doing so it will start performing the binary copy automatically. Be careful when using this tool and inserting the device name parameter. The image below shows an example of how to use it:


```
Enter the device name:
sdb1

Write the destination route:
/home/jordi/Escriptori
md5sum: /dev/sdb1: El fitxer o directori no existeix
[sudo] contrasenya per a jordi: █
```

Image 15: Running the script from the terminal.

The results in both cases will be three documents, 2 ones .txt that will contain the hash of the original device and the hash of the binary copy, and the third document will be the .dd file which is the binary copy of the device. All three files will be saved at the location indicated during the script.

3.2. Recover (File Carving)

In digital forensics, one of the most important tools is to recover files that has been corrupted or just that have been deleted. This tool is known with the name of “file carving” or just “carving”. This is a tool that is really often used on the forensics analysis, so we have considered that our tool must have a script to perform so.

In this case we have created a python script that allows the user to perform the file carving from a binary copy that has already been created from a device. The parameters required as inputs are only the location of the binary copy as well as the folder to put the results after the recover. There are several tools to perform the file carving, however, based on the results obtained by the project of Jordi Blanco, in this script we have used the free tool photorec.

The code of the tool I really simple, it just asks for the location of the image as well as the folder to save the results, and when the users runs it it will automatically perform the file carving process. The duration of this process can be different depending of the size of the binary image.

This tool is normally used after performing the binary copy of the artifact the user wants to analyze, so it can recover the deleted files of the artifact without touching it, which is really important to preserve the chain of custody.

If you are interested in the code of this tool check the appendix files from 5 to 8. There you will be able to find all the scripts used to run the tool as well as the graphical interface.

ADMINISTRATOR GUIDE:

In order to use this tool, as well as all the tools in this project, the operative system of the computer must be Linux. If the computer that the user is going to use has another operative system, it will have to use a virtual machine with a Linux system installed in order to run this tool.

This tool apart from the operative system also requires other programs to be installed: Python with its corresponding libraries (os, time, logging and gi with Gtk version 3). It is also recommended to have glade. All this tools can be installed using the command “sudo apt-get install xxxxxx” where xxxxxx is the name of the program required.


```
jordi@jordi-VirtualBox:~/Escriptori$ sudo apt-get install python
[sudo] contrasenya per a jordi:
S'està llegint la llista de paquets... Fet
S'està construint l'arbre de dependències
S'està llegint la informació de l'estat... Fet
```

Image 16: Example of how to install Python.

If the computer accomplishes all of this requirements, it is ready to prepare the tool for working. First of all, it is needed to get the compressed folder with all the scripts required to use the tool and unzip it into a known location. This can be done manually just by the default program on the operative system or to use the command “tar -xvf RECOVER.tar.gz”.

```
jordi@jordi-VirtualBox:~/Documents$ ls
RECOVER.tar.gz
jordi@jordi-VirtualBox:~/Documents$ tar -xvf RECOVER.tar.gz
RECOVER/recover.glade
RECOVER/Recover_terminal.py
RECOVER/Recover.desktop
RECOVER/Recover_gui.py
RECOVER/
jordi@jordi-VirtualBox:~/Documents$
```

Image 17: Extract files from the compressed folder using the terminal.

You have to know the exact location where the scripts have been unzipped. If you do not know it, you can use the terminal command “cd” to enter to the proper folders until reaching the folder where the scripts have been extracted, and once there use the command “pwd” to look for the current directory.

```
jordi@jordi-VirtualBox:~$ ls
Baixades  Escriptori  Imatges  Plantilles  Vídeos
Documents examples.desktop  Música  Públic
jordi@jordi-VirtualBox:~$ cd Documents/
jordi@jordi-VirtualBox:~/Documents$ ls
RECOVER RECOVER.tar.gz
jordi@jordi-VirtualBox:~/Documents$ cd RECOVER/
jordi@jordi-VirtualBox:~/Documents/RECOVER$ ls
Recover.desktop recover.glade Recover_gui.py Recover_terminal.py
jordi@jordi-VirtualBox:~/Documents/RECOVER$ pwd
/home/jordi/Documents/RECOVER
jordi@jordi-VirtualBox:~/Documents/RECOVER$
```

Image 18: Look the full path to the directory of the scripts with the command “pwd”.

The path to the location of the scripts is required to make the script runnable from any location with the file “.desktop”. To make it ready, just open the file “Recover.desktop” with a text editor. First start the text editor and open the document from there, since if you try to open it by clicking on it it will just try to run the script.

Once you have the document opened it has to be modified using the proper path of the location of the scripts. The lines that have to be replaced are the ones that start with “Exec=” and “Path=”, and the path that there is afterwards has to be replaced with the new location obtained using “pwd”. It is important not to modify the name of the script at the end of the line exec. Once done that just save the file with the same name.

```

[Desktop Entry]
Name=Recover
Exec=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/RECOVER/Recover_gui.py
Path=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/RECOVER/
Terminal=false
Type=Application

[Desktop Entry]
Name=Recover
Exec=/home/jordi/Documents/RECOVER/Recover_gui.py
Path=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/RECOVER/
Terminal=false
Type=Application

[Desktop Entry]
Name=Recover
Exec=/home/jordi/Documents/RECOVER/Recover_gui.py
Path=/home/jordi/Documents/RECOVER/
Terminal=false
Type=Application

```

Image 19: Modify the document "Recover.desktop" with the location of the script.

Once all of this is done, just move or copy the file "Recover.desktop" to anywhere in order to execute the scripts with the GUI (Graphical User Interface). Take into account that if this is moved to the account of another user, make sure it have the rights to execute scripts, otherwise it will not work. Consider as well that the icon of this file won't show the entire name, just "Recover".

USER GUIDE

The recover tool can be used with two different formats, one with a GUI that is really user friendly for somebody that is not used to work from a terminal, and a second option that is an script that is run from the terminal.

Using the tool with the GUI:

To run this tool it is really simple. In order to start using the tool just make double click on the icon of the "Recover" and this will start the tool (see the picture below).

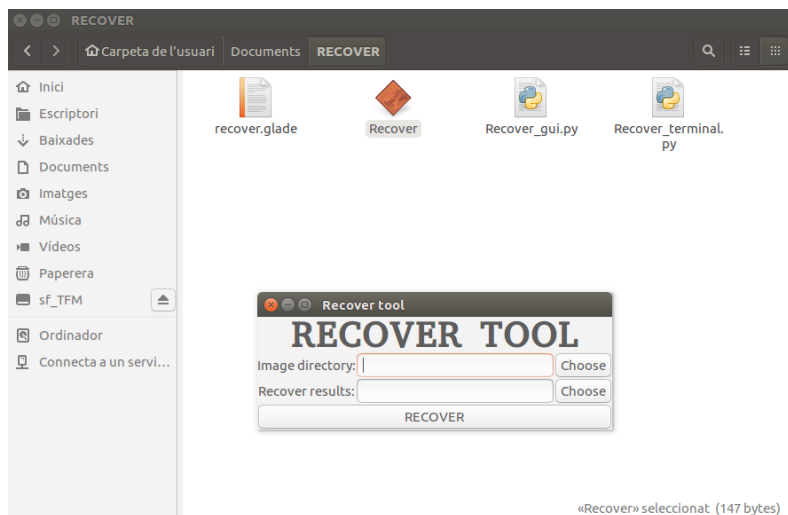


Image 20: Running the recover tool with the GUI just clicking on "Recover" icon.

As it has been explained before, this tool requires two input parameters. The first one is the location of the binary copy of the artifact that you want to analyze. Take into account

that this must include the name of the image. The second parameter is just the path to a folder where the script will save the results that it has been able to recover.

To choose them, it can be done just by typing the path on the corresponding text entry (be careful of not making any miss-spell or it will not work), or clicking at the button next to it. When doing this second option it will appear a window where you will be able to navigate through the folders and select the proper one. Clicking “Select” will automatically fill the text entry with the selected path.

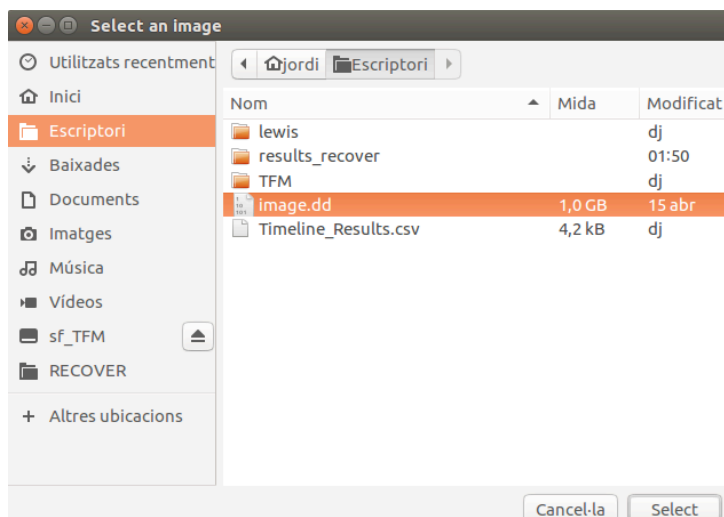


Image 21: Window to choose the path to the binary copy.



Image 22: The text entry is filled automatically when clicking "Select".

It is important to take into account that the parameters that the script will use are the ones written there, so if the user selects a folder and then modify its name, the second name will be the one used.

When both parameters have been selected, you can just click on the button that says “RECOVER” in order to start the file carving of the binary image. This may take quite some time, depending mainly on the size of the binary image. The results will be stored in another folder located inside the folder of the path named Recovered_files.# where # is a numeral (in that example will be 1).

Using the tool with the Terminal:

To run this tool from the terminal, first of all open a terminal and go to the location of the scripts.

```
jordi@jordi-VirtualBox: ~/Documents/RECOVER
jordi@jordi-VirtualBox:~$ ls
Baixades  Escriptori  Imatges  Plantilles  Videos
Documents examples.desktop  Música  Públic
jordi@jordi-VirtualBox:~$ cd Documents/
jordi@jordi-VirtualBox:~/Documents$ ls
RECOVER RECOVER.tar.gz
jordi@jordi-VirtualBox:~/Documents$ cd RECOVER/
jordi@jordi-VirtualBox:~/Documents/RECOVER$ ls
Recover.desktop recover.glade Recover_gui.py Recover_terminal.py
jordi@jordi-VirtualBox:~/Documents/RECOVER$
```

Image 23: Locating the scripts with the terminal.

Once the terminal is on that folder, just run the command “python Recover_terminal.py”. This will start the script but asking the parameters at the terminal without any type of GUI. To use the tool in this way just keep answering the parameters that the script asks you and after all are filled it will start automatically. In the image below can be seen an example of how to use the recover tool with the terminal:

```
jordi@jordi-VirtualBox: ~/Escriptori
jordi@jordi-VirtualBox:~$ cd Escriptori/
jordi@jordi-VirtualBox:~/Escriptori$ ls
image.dd lewis results_recover TFM Timeline_f
jordi@jordi-VirtualBox:~/Escriptori$ pwd
/home/jordi/Escriptori
jordi@jordi-VirtualBox:~/Escriptori$

jordi@jordi-VirtualBox: ~/Documents/RECOVER
jordi@jordi-VirtualBox:~/Documents/RECOVER$ python Recover_terminal.py
Welcome to the UPC forensics app. This is the Recovery Files tool.

Write the name of the file (ex. image.dd):
image.dd

Write the file route (ex. /root/docs):
/home/jordi/Escriptori

Write the destination route (ex. /root/results):
/home/jordi/Escriptori/results_recover
PhotoRec 7.0, Data Recovery Utility, April 2015
Christophe GRENIER <grenier@cgsecurity.org>
http://www.cgsecurity.org
jordi@jordi-VirtualBox:~/Documents/RECOVER$
```

Image 24: Using the recover tool from the terminal.

As it has been stated at the beginning, this tool uses the free tool photorec to perform the file carving. When the script uses this, the terminal will change a little. You do not need to touch anything, just wait until it finishes and it will return to the other terminal again.

```
jordi@jordi-VirtualBox: ~/Documents/RECOVER
PhotoRec 7.0, Data Recovery Utility, April 2015
Christophe GRENIER <grenier@cgsecurity.org>
http://www.cgsecurity.org

Disk /home/jordi/Escriptori/image.dd - 1048 MB / 999 MiB (R0)
Partition      Start      End      Size in sectors
P FAT16         0 0 1    127 121 59    2047937 [USB DISK]

Pass 1 - Reading sector      595320/2047937, 80 files found
Elapsed time 0h00m23s - Estimated time to completion 0h00m56
jpg: 73 recovered
gz: 2 recovered
tx?: 2 recovered
dbf: 1 recovered
dta: 1 recovered
sqlite: 1 recovered

Stop
```

Image 25: Photorec performing the file carving from the terminal.

3.3. Timeline

In digital forensics, talking about getting the Timeline of a certain directory is to get all the information about all the files inside that directory. The information normally it is

interesting to include in a timeline are the name of the file, the creation time, the modification time, the access time, the size...

Looking for the timeline of a directory is a tool frequently used in forensics. That's a motivation to create a tool that will automatically perform an analysis of a directory and all of the files that it contains and will create a document with all the interesting information.

To do so, we have created a python script that the input parameters will be the path of the folder that we want to analyze and the directory of where to save the results of the analysis. We have used some part of the code of the tool "Quickfish" that allows performing timelines among other things. Since the aim of this project is to create a really easy to use tool, it is not necessary to use all the functionalities that this program allows so it has just been used certain part of the code.

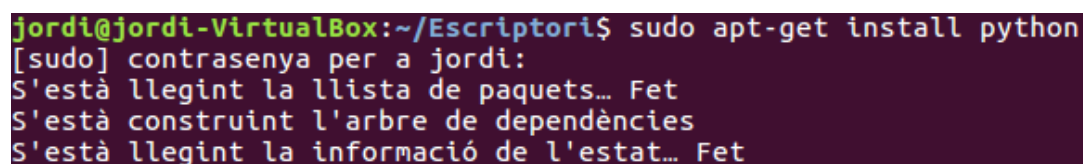
This code what mainly does is it enters to the directory and looks for all the files that are inside of it. For each of the files it checks if it is able to open and if so looks for all the information that it might be interesting to get. All this information is stored into a ".csv" file that will be saved at the location that has been introduced.

If you are interested in the code of the files to run this tool, in the appendix from 9 to 12 you can found all the codes of the scripts and the graphical interface to run this scripts.

ADMINISTRATOR GUIDE:

When preparing a computer to use this tool, first of all it is interesting to look at the system requirements. This tool is prepared to run in the operative system Linux. In case the computer has another operative system, it will have to have a virtual machine with a Linux version installed on it.

Apart from the operative system, this will have to have installed already some programs. In the case of this tools, the requirements are the following: Python with its corresponding libraries (logging, time, sys, os, stat, time, hashlib, argparse, csv, gi with Gtk version 3). It is also recommended to have glade installed as well as a program able to read ".csv" files like OppenOffice Calc. All this tools can be installed using the command "sudo apt-get install xxxxxx" where xxxxxx is the name of the program required.



```
jordi@jordi-VirtualBox:~/Escriptori$ sudo apt-get install python
[sudo] contrasenya per a jordi:
S'està lligint la llista de paquets... Fet
S'està construint l'arbre de dependències
S'està lligint la informació de l'estat... Fet
```

Image 26: Example of how to install Python.

Once all of this is ready, the tool has to be prepared in order to be used. First of all, it is needed to get the compressed folder with all the scripts required to use the tool and unzip it into a known location. This can be done manually just by the default program on the operative system or to use the command "tar -xvf TIMELINE.tar.gz".

```
jordi@jordi-VirtualBox:~/Documents$ ls
TIMELINE.tar.gz
jordi@jordi-VirtualBox:~/Documents$ tar -xvf TIMELINE.tar.gz
TIMELINE/Timeline.desktop
TIMELINE/Timeline.glade
TIMELINE/Timeline_gui.py
TIMELINE/
TIMELINE/Timeline_terminal.py
jordi@jordi-VirtualBox:~/Documents$
```

Image 27: Extract files from the compressed folder using the terminal.

Make sure to know the exact path to the location where it has been extracted the documents. In case it is not known, using the terminal command “cd” to enter to the proper folders until reaching the folder where the scripts have been extracted, and once there use the command “pwd” to look for the current directory.

```
jordi@jordi-VirtualBox: ~/Documents/TIMELINE
jordi@jordi-VirtualBox:~$ ls
Baixades  Escriptori  Imatges  Plantilles  Videos
Documents examples.desktop  Música  Públic
jordi@jordi-VirtualBox:~$ cd Documents/
jordi@jordi-VirtualBox:~/Documents$ ls
TIMELINE  TIMELINE.tar.gz
jordi@jordi-VirtualBox:~/Documents$ cd TIMELINE/
jordi@jordi-VirtualBox:~/Documents/TIMELINE$ ls
Timeline.desktop  Timeline.glade  Timeline_gui.py  Timeline_terminal.py
jordi@jordi-VirtualBox:~/Documents/TIMELINE$ pwd
/home/jordi/Documents/TIMELINE
jordi@jordi-VirtualBox:~/Documents/TIMELINE$
```

Image 28: Look the full path to the directory of the scripts with the command “pwd”.

This path it is necessary to make the script runnable from anywhere. To do so, it is necessary a text editor or if it is preferred to read it from the terminal. Open a text editor like gedit and once opened, oped the file called “timeline.desktop”. It is essential to do it that way since this file cannot be opened by double clicking on it (it is prepared to execute the scripts when clicking on it).

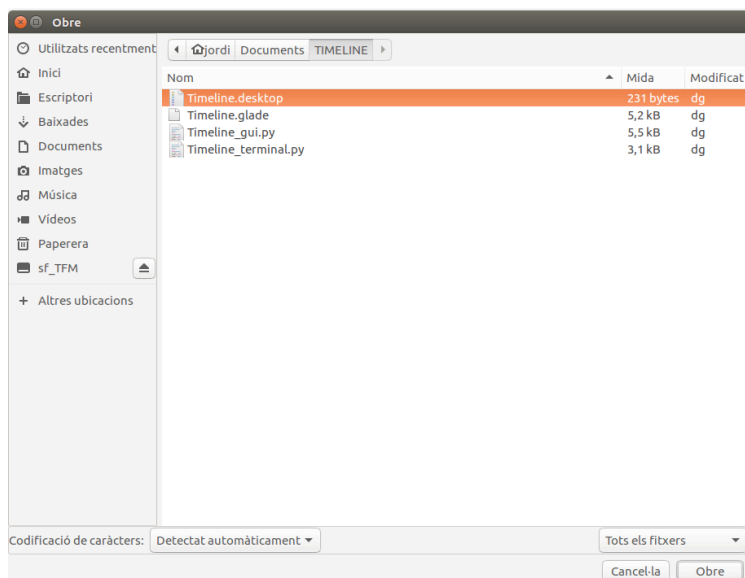


Image 29: Opening Timeline.desktop from gedit.

Now, this document has to be modified using the proper path of the location of the scripts. The lines that have to be replaced are the ones that start with “Exec=” and “Path=”, and the path that there is afterwards has to be replaced with the new location obtained using “pwd”. It is important not to modify the name of the script at the end of the line exec. Once done that just save the file with the same name.

```
[Desktop Entry]
Name=Timeline
Exec=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/TIMELINE/Timeline_gui.py
Path=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/TIMELINE/
Terminal=false
Type=Application

[Desktop Entry]
Name=Timeline
Exec=/home/jordi/Documents/TIMELINE/Timeline_gui.py
Path=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/TIMELINE/
Terminal=false
Type=Application

[Desktop Entry]
Name=Timeline
Exec=/home/jordi/Documents/TIMELINE/Timeline_gui.py
Path=/home/jordi/Documents/TIMELINE/
Terminal=false
Type=Application
```

Image 30: Modify the "Timeline.desktop" with the new path.

Once all of this is done, just move or copy the file “Timeline.desktop” to anywhere in order to execute the scripts with the GUI (Graphical User Interface). Take into account that if this is moved to the account of another user, make sure it have the rights to execute scripts, otherwise it will not work. Consider as well that the icon of this file won’t show the entire name, just “Timeline”.

USER GUIDE

This tool can be used in two different formats: the first one is with the GUI and can be run just by double-clicking into the icon “Timeline”. And the second one is a version that does not need GUI and is completely run using just a terminal.

Using the tool with the GUI:

To run this tool it is really easy and intuitive. First of all make double click on the icon of the timeline will open the main window of the tool (see the picture below).

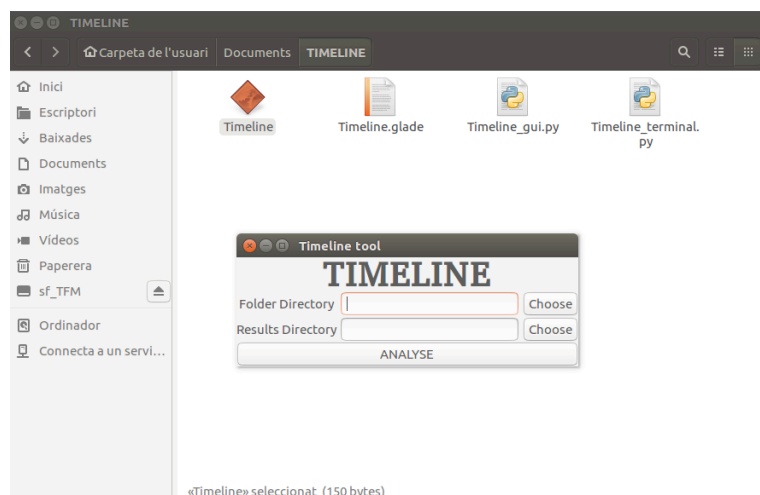


Image 31: Double click to "Timeline" to run the script with the GUI.

As it can be seen in the picture, there are two parameters required to perform the analysis: the folder directory and the results directory. The folder directory is the path to the folder that the user wants to analyze. The results directory is the path of the place where the user wants the results of the analysis.

To choose them, it can be done just by typing the path on the corresponding text entry (be careful of not making any miss-spell or it will not work), or clicking at the button next to it. When doing this second option it will appear a window where you will be able to navigate through the folders and select the proper one. Clicking "Select" will automatically fill the text entry with the selected path.

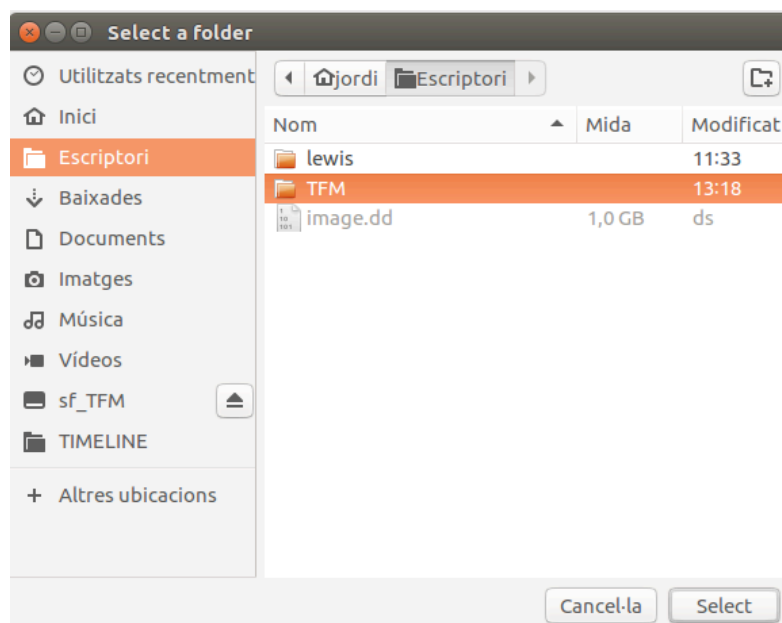


Image 32: Window to choose the path to the folder.



Image 33: Text entry is filled when clicking "Select".

It is important to take into account that the parameters that the script will use are the ones written there, so if the user selects a folder and then modify its name, the second name will be the one used.

Once both of the folders are selected just clicking the button "ANALYZE" will perform the examination of the files and will automatically save a document called Timeline_Results.csv with all the results. This document will look like the image below:

	A	B	C	D	E	F	G	H	I	J
	File	Path	Size	Modified Time	Access Time	Created Time	Owner	Group	Mode	
1	logfile.log	/home/jordi/Esriptori/TFM/Forensics_Scripts/logfile.log	164	Sun Apr 16 03:10:27 2017	Sun Apr 16 05:18:25 2017	Sun Apr 16 03:10:27 2017	1000	1000	33204	
2	Timeline.py	/home/jordi/Esriptori/TFM/Forensics_Scripts/Timeline.py	7467	Wed Apr 12 11:51:54 2017	Sun Apr 16 03:10:27 2017	Fri Apr 14 12:18:35 2017	1000	1000	33204	
3	names.py	/home/jordi/Esriptori/TFM/Forensics_Scripts/names.py	553	Wed Apr 12 11:51:48 2017	Sun Apr 16 03:10:27 2017	Fri Apr 14 12:18:35 2017	1000	1000	33204	
4	OCR_script.py	/home/jordi/Esriptori/TFM/Forensics_Scripts/OCR_script.py	566	Wed Apr 12 11:51:52 2017	Sun Apr 16 03:10:27 2017	Fri Apr 14 12:18:35 2017	1000	1000	33204	
5	Timeline_v1.py	/home/jordi/Esriptori/TFM/Forensics_Scripts/Timeline_v1.py	3820	Sun Apr 16 03:29:38 2017	Sun Apr 16 03:36:27 2017	Sun Apr 16 03:29:38 2017	1000	1000	33204	
6	prepare_pdf.py	/home/jordi/Esriptori/TFM/Forensics_Scripts/prepare_pdf.py	1229	Wed Apr 12 11:51:54 2017	Sun Apr 16 03:10:27 2017	Fri Apr 14 12:18:35 2017	1000	1000	33204	
7	Binary_Copy.py	/home/jordi/Esriptori/TFM/Forensics_Scripts/Binary_Copy.py	620	Sat Apr 15 00:27:45 2017	Sun Apr 16 03:10:27 2017	Sat Apr 15 00:27:45 2017	1000	1000	33204	
8	Recover.py	/home/jordi/Esriptori/TFM/Forensics_Scripts/Recover.py	505	Wed Apr 12 11:51:48 2017	Sun Apr 16 03:10:27 2017	Fri Apr 14 12:18:35 2017	1000	1000	33204	
9	recover_glade	/home/jordi/Esriptori/TFM/Forensics_Scripts/Scripts_with/	5041	Sun Apr 16 04:32:55 2017	Sun Apr 16 04:33:04 2017	Sun Apr 16 04:32:55 2017	1000	1000	33204	
10	Recover_terminal.py	/home/jordi/Esriptori/TFM/Forensics_Scripts/Scripts_with/	837	Sun Apr 16 04:42:41 2017	Sun Apr 16 04:45:12 2017	Sun Apr 16 04:42:41 2017	1000	1000	33277	
11	Recover_desktop	/home/jordi/Esriptori/TFM/Forensics_Scripts/Scripts_with/	227	Sat Apr 15 14:32:09 2017	Sat Apr 15 14:32:09 2017	Sat Apr 15 14:32:09 2017	1000	27	33279	
12	Recover_gui.py	/home/jordi/Esriptori/TFM/Forensics_Scripts/Scripts_with/	3087	Sun Apr 16 04:43:42 2017	Sun Apr 16 04:45:32 2017	Sun Apr 16 04:43:42 2017	1000	1000	33277	
13	Timeline_desktop	/home/jordi/Esriptori/TFM/Forensics_Scripts/Scripts_with/	231	Sun Apr 16 05:23:30 2017	Thu Apr 20 11:37:58 2017	Sun Apr 16 05:24:56 2017	1000	1000	33279	
14	Timeline_terminal.py	/home/jordi/Esriptori/TFM/Forensics_Scripts/Scripts_with/	3063	Sun Apr 16 05:19:29 2017	Sun Apr 16 05:51:01 2017	Sun Apr 16 05:25:18 2017	1000	1000	33277	
15	Timeline_gui.py	/home/jordi/Esriptori/TFM/Forensics_Scripts/Scripts_with/	5473	Sun Apr 16 05:19:35 2017	Thu Apr 20 11:38:02 2017	Sun Apr 16 05:19:35 2017	1000	1000	33277	
16	Timeline_glade	/home/jordi/Esriptori/TFM/Forensics_Scripts/Scripts_with/	5207	Sun Apr 16 04:32:53 2017	Sun Apr 16 04:33:17 2017	Sun Apr 16 04:32:53 2017	1000	1000	33204	
17	OCR_directory3.py	/home/jordi/Esriptori/TFM/Forensics_Scripts/Scripts_with/	2261	Wed Apr 12 11:51:54 2017	Sun Apr 16 03:10:27 2017	Fri Apr 14 12:18:35 2017	1000	1000	33204	
18	State_of_the_art_v1.docx	/home/jordi/Esriptori/TFM/Memory/State_of_the_art_v1.docx	10053	Wed Apr 12 11:51:52 2017	Sat Apr 15 16:46:13 2017	Fri Apr 14 12:18:35 2017	1000	1000	33204	
19	timeline.odt	/home/jordi/Esriptori/TFM/Memory/timeline.odt	24788	Thu Apr 20 12:58:31 2017	Thu Apr 20 12:58:31 2017	Thu Apr 20 12:58:31 2017	1000	1000	33204	
20	~lock.timeline.odt#	/home/jordi/Esriptori/TFM/Memory/~lock.timeline.odt#	82	Thu Apr 20 12:58:31 2017	Thu Apr 20 12:58:31 2017	Thu Apr 20 12:58:31 2017	1000	1000	33204	
21	Project_development.odt	/home/jordi/Esriptori/TFM/Memory/Project_development.odt	157967	Sun Apr 16 06:03:55 2017	Sun Apr 16 06:03:55 2017	Sun Apr 16 06:03:55 2017	1000	1000	33204	
22	plantilla_tfm_v3.doc	/home/jordi/Esriptori/TFM/Memory/plantilla_tfm_v3.doc	224768	Wed Apr 12 11:51:48 2017	Sun Apr 16 03:10:27 2017	Fri Apr 14 12:18:35 2017	1000	1000	33204	
23	Introduction.docx	/home/jordi/Esriptori/TFM/Memory/Introduction.docx	107059	Wed Apr 12 11:51:52 2017	Sun Apr 16 03:10:27 2017	Fri Apr 14 12:18:35 2017	1000	1000	33204	
24										
25										
26										

Image 34: Document with the results of the analysis.

The information recorded in the order that it appears to the document is the following: name of the file, path to reach the file, size of the file in bytes, time when the last modification has been done, time of the last time this file has been accessed, time of the creation of the document, owner of the document (its a number associated with the user, to check them type id in the terminal), group of the document (the same as the owner but referring to the group), and the last is the mode which is a number that makes reference to the permissions (read, read-write....).

```
jordi@jordi-VirtualBox:~/Documents/TIMELINE$ id
uid=1000(jordi) gid=1000(jordi) groups=1000(jordi),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
```

Image 35: Owner and grup numbers.

Using the tool with the Terminal:

Running this tool from the terminal is quite simple if you are used to work with the terminal. What you first need to do is to locate the folder where the scripts are located as it can be seen in the picture below:

```
jordi@jordi-VirtualBox: ~/Documents/TIMELINE
jordi@jordi-VirtualBox:~$ ls
Baixades  Escriptori  Imatges  Plantilles  Videos
Documents examples.desktop  Música  Públic
jordi@jordi-VirtualBox:~$ cd Documents/
jordi@jordi-VirtualBox:~/Documents$ ls
TIMELINE  TIMELINE.tar.gz
jordi@jordi-VirtualBox:~/Documents$ cd TIMELINE/
jordi@jordi-VirtualBox:~/Documents/TIMELINE$ ls
Timeline.desktop  Timeline.glade  Timeline_gui.py  Timeline_terminal.py
jordi@jordi-VirtualBox:~/Documents/TIMELINE$
```

Image 36: Locate the folder with the scripts.

Once there just type the command “python Timeline_terminal.py”. This will automatically run the script at the terminal. To use this tool from here, you just need to answer all the parameters that the script asks you. And after doing so it will start performing the timeline automatically. The image below shows an example of how to use it:

```
jordi@jordi-VirtualBox: ~/Escriptori
jordi@jordi-VirtualBox:~$ cd Escriptori/
jordi@jordi-VirtualBox:~/Escriptori$ ls
image.dd  lewis  results_recover  TFM  Timeline_Results.csv
jordi@jordi-VirtualBox:~/Escriptori$ pwd
/home/jordi/Escriptori
jordi@jordi-VirtualBox:~/Escriptori$

jordi@jordi-VirtualBox: ~/Documents/TIMELINE
jordi@jordi-VirtualBox:~/Documents/TIMELINE$ python Timeline_terminal.py
Welcome to the UPC forensics app. This is the Timeline tool.

Write the root folder path (ex. /root/docs):
/home/jordi/Escriptori/TFM

Write the destination path (ex. /root/results):
/home/jordi/Escriptori
jordi@jordi-VirtualBox:~/Documents/TIMELINE$
```

Image 37: Running the script from the terminal.

3.4. OCR

Optical Character Recognition (OCR), is one tool that is frequently used in digital forensics. As its name says, this tool is in charge of reading the text of pictures or pdfs or other type of documents to allow a search of them.

This is so important because normally, when there is a case required to analyze a lot of information, the people in charge of the investigation can not afford the time of looking through all the documents (also it is not totally legal to do so). So what is done in these cases is to perform a mass search of certain key words through all the documents. The problem is that this search can be performed if the documents are text documents, but cannot read text that is inside images or pdf. This is the motivation why OCR is commonly used in digital forensics.

There are several tools to perform an OCR, there are options for any operative system as well as there are some that are free whereas some are paid to do so. In this project, we have been working with several tools and testing them. The first one we looked at was tesseract, a free tool that performs quite well the OCR, however the main problem is that the user had to “prepare” the image before trying to perform the OCR. Another tools that we have tested are ABBY and Adobe. These tools are prepared for the operative system Windows whereas the first one is for Linux. Abby and Adobe perform much better the OCR and also is not required to prepare the images to perform it. The main problem is that are not free.

This tool has been divided into three parts to allow performing a mass OCR to a huge directory of files without losing its structure. The three parts are prepare (with replaces white spaces in the name for “_s_” to avoid having problems with some of the commands

used), the second part is divide (some OCR tools do not allow converting more than one image at the same time for example pdf of more than 1 page, so this part of the tool gets all the pdf and divide them into pages, and all of them are saved into the same folder (the name of the path is kept at the name of this new documents created replacing the “/” for “_b_”) and the third part is the OCR.

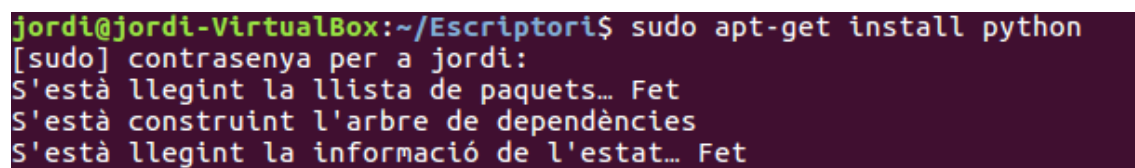
The OCR included in this tool is tesseract, because it is free and because it is for Linux. However, this tool is prepared for, if the user wants, the first steps can be performed without doing the OCR and uses an external tool to perform it.

If you are interested in the code of the files to run this tool, in the appendix from 13 to 16 you can found all the codes of the scripts and the graphical interface to run this scripts.

ADMINISTRATOR GUIDE:

As all the other tools in this project, the operative system required is Linux. It can be just installed in the computer or having a virtual machine with this operative system in case the computer does not have a partition with Linux.

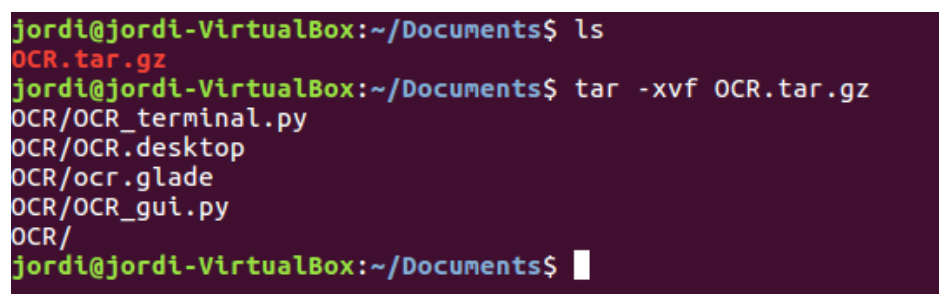
Apart from the operative system, this will have to have installed already some programs. In the case of this tools, the requirements are the following: Python with its corresponding libraries (os, time, logging, pyPdf, gi with Gtk version 3). Apart from that it is also required to have some other tools installed: ImageMagic, Tesseract and pdfseparate. It is also recommended to have glade installed. All this tools can be installed using the command “sudo apt-get install xxxxxx” where xxxxxx is the name of the program required.



```
jordi@jordi-VirtualBox:~/Escriptori$ sudo apt-get install python
[sudo] contrasenya per a jordi:
S'està llegint la llista de paquets... Fet
S'està construïnt l'arbre de dependències
S'està llegint la informació de l'estat... Fet
```

Image 38: Example of how to install Python.

Once all of this is ready, the tool has to be prepared in order to be used. First of all, it is needed to get the compressed folder with all the scripts required to use the tool and unzip it into a known location. This can be done manually just by the default program on the operative system or to use the command “tar -xvf OCR.tar.gz”.



```
jordi@jordi-VirtualBox:~/Documents$ ls
OCR.tar.gz
jordi@jordi-VirtualBox:~/Documents$ tar -xvf OCR.tar.gz
OCR/OCR_terminal.py
OCR/OCR.desktop
OCR/ocr.glade
OCR/OCR_gui.py
OCR/
jordi@jordi-VirtualBox:~/Documents$
```

Image 39: Extract files from the compressed folder using the terminal.

Make sure to know the exact path to the location where it has been extracted the documents. In case it is not known, using the terminal command “cd” to enter to the proper folders until reaching the folder where the scripts have been extracted, and once there use the command “pwd” to look for the current directory.

```
jordi@jordi-VirtualBox:~$ ls
Baixades  Escriptori  glade  Música  Públic  test1.tiff
Documents examples.desktop Imatges Plantilles res.txt Videos
jordi@jordi-VirtualBox:~$ cd Documents/
jordi@jordi-VirtualBox:~/Documents$ ls
OCR OCR.tar.gz
jordi@jordi-VirtualBox:~/Documents$ cd OCR/
jordi@jordi-VirtualBox:~/Documents/OCR$ ls
OCR.desktop ocr.glade OCR_gui.py OCR_terminal.py
jordi@jordi-VirtualBox:~/Documents/OCR$ pwd
/home/jordi/Documents/OCR
jordi@jordi-VirtualBox:~/Documents/OCR$
```

Image 40: Look the full path to the directory of the scripts with the command "pwd".

This path it is necessary to make the script runnable from anywhere. To do so, it is necessary a text editor or if it is preferred to read it from the terminal. Open a text editor like gedit and once opened, oped the file called "OCR.desktop". It is essential to do it that way since this file cannot be opened by double-clicking on it (it is prepared to execute the scripts when clicking on it).

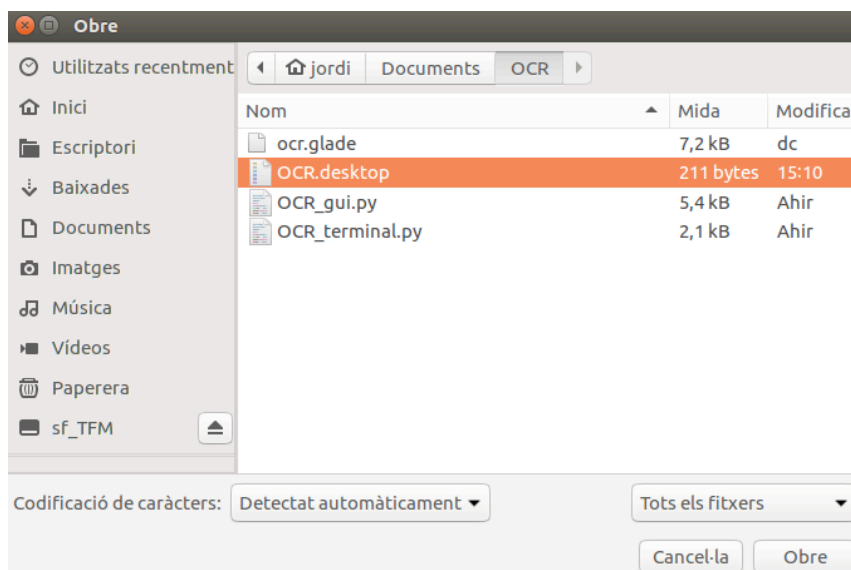


Image 41: Opening OCR.desktop from gedit.

Now, this document has to me modified using the proper path of the location of the scripts. The lines that have to be replaced are the ones that start with "Exec=" and "Path=", and the path that there is afterwards has to be replaced with the new location obtained using "pwd". It is important not to modify the name of the script at the end of the line exec. Once done that just save the file with the same name.

```

[Desktop Entry]
Name=OCR
Exec=/home/jordi/Escriptori/TFM/Forensics Scripts/Scripts with GUI/OCR/OCR_gui.py
Path=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/OCR/
Terminal=false
Type=Application

[Desktop Entry]
Name=OCR
Exec=/home/jordi/Documents/OCR/OCR_gui.py
Path=/home/jordi/Escriptori/TFM/Forensics Scripts/Scripts with GUI/OCR/
Terminal=false
Type=Application

[Desktop Entry]
Name=OCR
Exec=/home/jordi/Documents/OCR/OCR_gui.py
Path=/home/jordi/Documents/OCR/
Terminal=false
Type=Application

```

Image 42: Modify the document "OCR.desktop" with the new path.

Once all of this is done, just move or copy the file "OCR.desktop" to anywhere in order to execute the scripts with the GUI (Graphical User Interface). Take into account that if this is moved to the account of another user, make sure it have the rights to execute scripts, otherwise it will not work. Consider as well that the icon of this file won't show the entire name, just "OCR".

USER GUIDE

This tool can be used in two different formats: the first one is with the GUI and can be run just by doubleclicking into the icon "OCR". And the second one is a version that does not need GUI and is completely run using just a terminal.

Using the tool with the GUI:

To run this tool it is really easy and intuitive. First of all make double click on the icon of the OCR will open the main window of the tool (see the picture below).

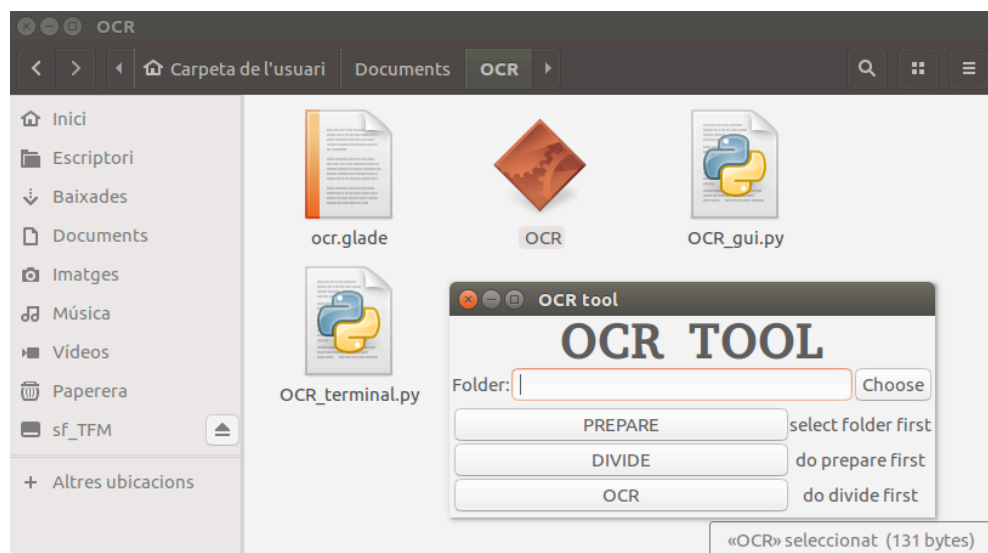


Image 43: Double click to "OCR" to run the script with the GUI.

As it can be observed, the window that appears has 4 options, these options have to be done in the correct order, otherwise the program may have problems. First of all select the folder that contains the directory to analyze. It can be just typed into the text entry or just clicking into the button “choose” will open another window to select the directory. If it has been used the second option the name will be written there automatically.

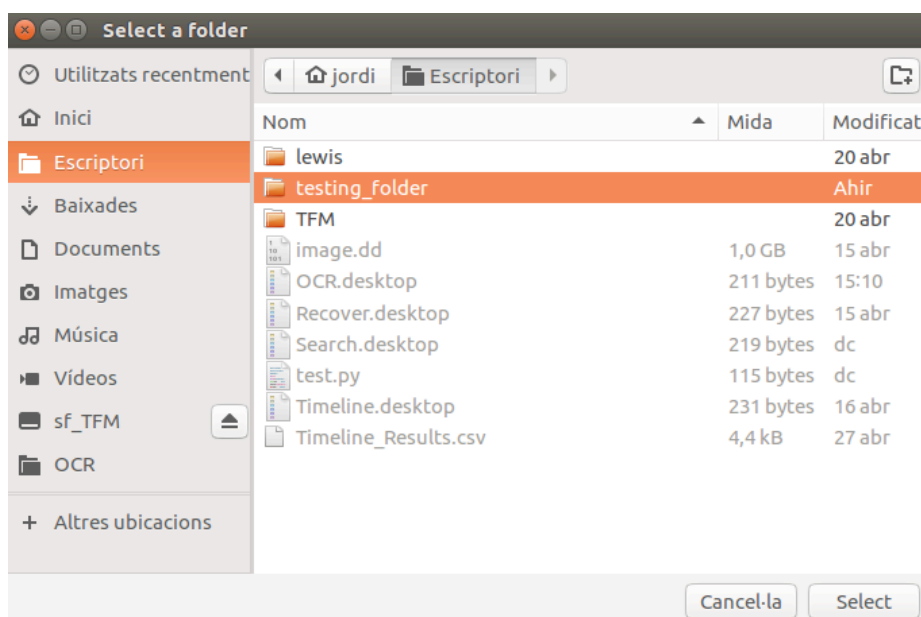


Image 44: Window to choose the path to the folder.

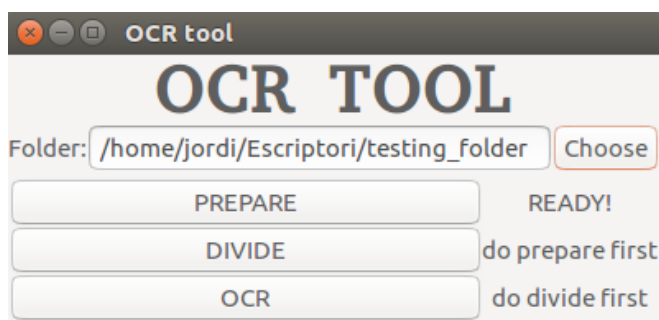


Image 45: The text entry is filled automatically when clicking "Select".

Once this is done it is important not to modify this path while performing the other operations, since doing so may crash the program. Once this is done, as can be observe at the image, the option prepare is ready. What does this option is to replace all the white spaces in the names of the folders by “_s_”. It is necessary to do so to perform the other actions. Just by clicking to the button in will automatically do so.

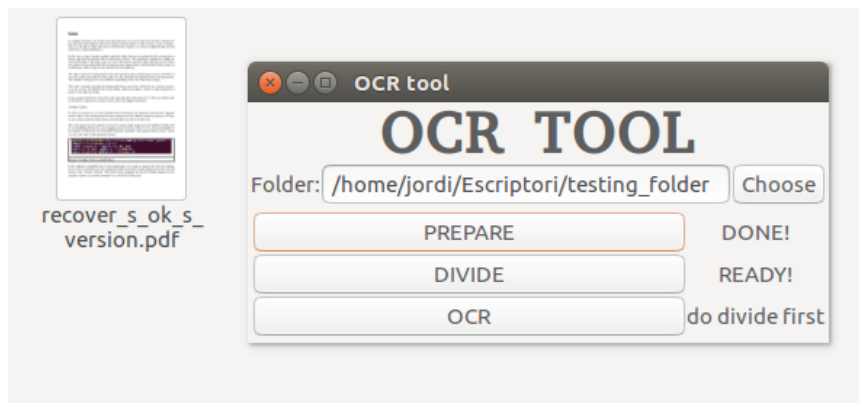


Image 46: Preparing the folder names by clicking on "PREPARE".

As it can be seen in the picture, all the names with spaces have been modified. Now it comes another step. Getting all the pdf files, divide them and put them in the same folder to make them ready to pass an OCR. Clicking the button "DIVIDE" automatically does this.

All of these pages will be saved into a new folder that will be created called "all_pdf_text". The names of this files will contain the path of the original document into its name replacing the "/" for "_" in order to be able to connect this results to the original ones.

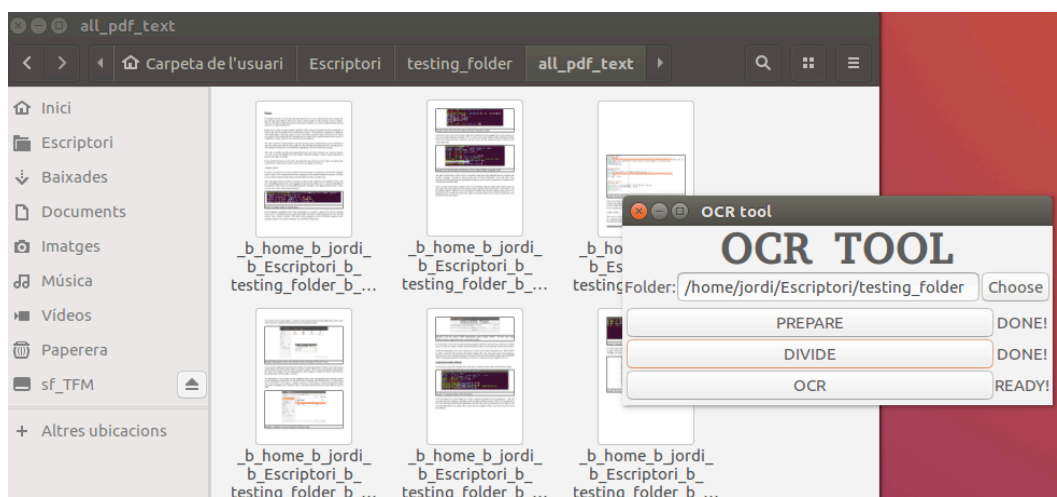


Image 47: Dividing the pdf files into pages and saving the results into "all_pdf_text" by clicking on "DIVIDE".

Now there is just one last step: performing the OCR. If at this point the user prefers to use another program to perform the OCR, just do it but conserving the name of the files exactly as it is (in order to be able to use the search tool).

If that is not the case, the user can also use the OCR tool provided here, "tesseract". Clicking on the last button "OCR" will do 2 things. First what it will do is to convert the pdf pages into images ".tiff" in order to enhance the quality of the OCR, and to allow using tesseract since it does not work with pdf files. The second thing it does is the OCR. The results are saved into the "all_pdf_text" folder and all the temporary files (pdf and tiff) are deleted, so the folder will only contain the .txt files.

This last action may last longer than the others. The time of all this operations will mainly depend basically in the amount of files that the user wants to analyze.

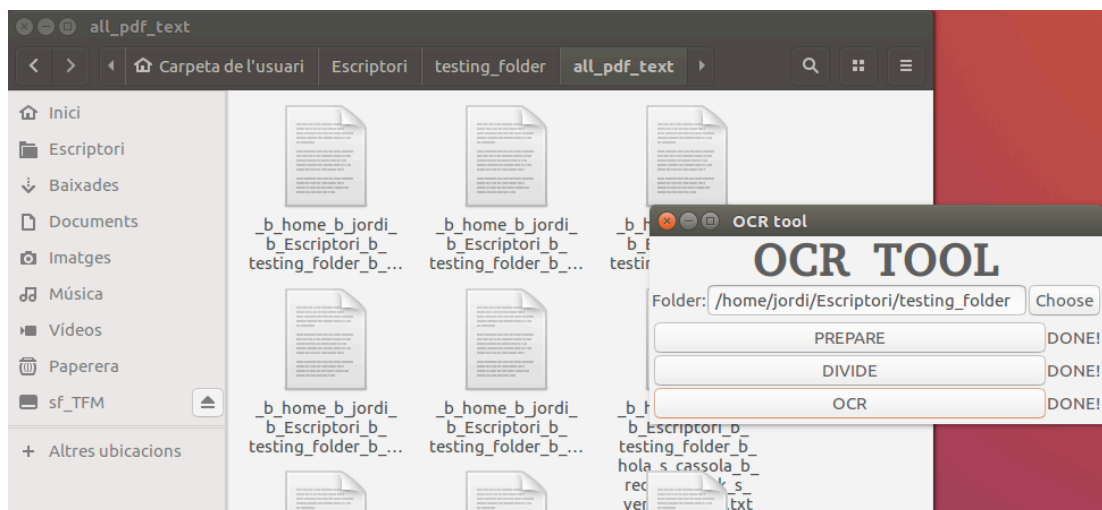


Image 48: Performing the OCR.

Using the tool with the Terminal:

Running this tool from the terminal is quite simple if you are used to work with the terminal. What you first need to do is to locate the folder where the scripts are located as it can be seen in the picture below:

```
jordi@jordi-VirtualBox: ~/Documents/OCR
jordi@jordi-VirtualBox:~$ ls
Baixades  Escriptori  glade  Música  Públic  test1.tiff
Documents examples.desktop Imatges Plantilles res.txt Videos
jordi@jordi-VirtualBox:~$ cd Documents/
jordi@jordi-VirtualBox:~/Documents$ ls
OCR  OCR.tar.gz
jordi@jordi-VirtualBox:~/Documents$ cd OCR/
jordi@jordi-VirtualBox:~/Documents/OCR$ ls
OCR.desktop  ocr.glade  OCR_gui.py  OCR_terminal.py
```

Image 49: Locate the folder with the scripts.

Once there just type the command “python OCR_terminal.py”. This will automatically run the script at the terminal. To use this tool from here, you just need to answer all the parameters that the script asks you. And after doing so it will start performing the timeline automatically. The image below shows an example of how to use it:


```

jordi@jordi-VirtualBox:~/Documents/OCRS$ python OCR_terminal.py
Welcome to the UPC forensics app. This is the Massive OCR tool.

Write the directory path (ex. /root/docs):
/home/jordi/Escriptori/testing_folder

Type an order (options: PREPARE, DIVIDE, OCR, EXIT):
PREPARE

PREPARATION DONE!

Type an order (options: PREPARE, DIVIDE, OCR, EXIT):
DIVIDE

DIVISION DONE!

Type an order (options: PREPARE, DIVIDE, OCR, EXIT):
OCR
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
Detected 128 diacritics
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
Detected 184 diacritics
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
Tesseract Open Source OCR Engine v3.04.01 with Leptonica
Page 1
OCR DONE!

Type an order (options: PREPARE, DIVIDE, OCR, EXIT):
EXIT
Goodbye
jordi@jordi-VirtualBox:~/Documents/OCRS$ █

```

Image 50: Running the script from the terminal.

Take into account that running the script from the terminal will allow the user to perform just one action as well, but in this case it can skip some steps or modify the order. It is strongly recommended not to do so unless the user really knows what it is doing since it may have some problems with it.

3.5. eDiscovery

Once all the operations of the forensics have been performed, it remains the last step, which consists on analyzing the results in order to get the evidences of the case. This action is called eDiscovery and consists on searching key-words among all the documents that are being investigated, in order to find those ones of our interest.

In our case, we have created a GUI (graphical user interface) in order to make simpler this task. In the case of this tool we have not created an script to use the program from the terminal since the use of the GUI is to avoid exactly so (since if there are a lot of files it may be difficult to work with them from a terminal).

This script is really simple, as input it is required to select the directory of documents .txt to analyze. and just type a word to search. This script will automatically check all the content of the txt files and will show up a list with some information about each document which contains the word: the name of the document, the path and the page number where the word has been encountered.

It is really important when using this tool that the names of the txt files in the folder is the one obtained by our other tool (the OCR tool). This tool is prepared to analyze. the output of the other tool, if the name of the documents do not contain the path to the original file, the program may crash.

Once the search has been performed there are some options to do with the current results before starting another search. This options are 3: open the txt file, open the original pdf file and save the list with the results into a .csv document.

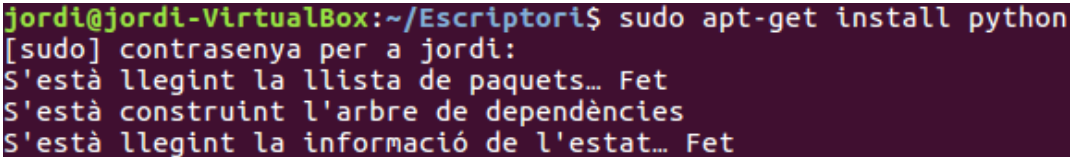
At the start of the tool it will ask which search tool to you want to use: “grep” or “gawk”. This are the Linux commands used to perform the analysis. Each one with its own advantages and drawbacks explained below.

If you are interested in the code of this tool check the appendix files from 17 to 22. There you will be able to find all the scripts used to run the tool as well as the graphical interface.

ADMINISTRATOR GUIDE:

In order to use this tool, as well as all the tools in this project, the operative system of the computer must be Linux. If the computer that the user is going to use has another operative system, it will have to use a virtual machine with a Linux. system installed in order to run this tool.

This tool apart from the operative system also requires other programs to be installed: Python with its corresponding libraries (subprocess, os, tempfile, ntpathos, time, logging and gi with Gtk version 3). To use the “gawk” tool it has to be installed as well. It is also recommended to have glade. All this tools can be installed using the command “sudo apt-get install xxxxxx” where xxxxxx is the name of the program required.



```
jordi@jordi-VirtualBox:~/Escriptori$ sudo apt-get install python
[sudo] contrasenya per a jordi:
S'està llegint la llista de paquets... Fet
S'està construint l'arbre de dependències
S'està llegint la informació de l'estat... Fet
```

Image 51: Example of how to install Python.

Once the computer has all the requirements installed, it is ready to prepare the tool for working. First of all, it is needed to get the compressed folder with all the scripts required to use the tool and unzip it into a known location. This can be done manually just by the default program on the operative system or to use the command “tar -xvf EDISCOVERY.tar.gz”.

```
jordi@jordi-VirtualBox:~/Documents$ ls
EDISCOVERY.tar.gz
jordi@jordi-VirtualBox:~/Documents$ tar -xvf EDISCOVERY.tar.gz
EDISCOVERY/addvante_logo.png
EDISCOVERY/Search.py
EDISCOVERY/logo_petit.png
EDISCOVERY/time_grep.log
EDISCOVERY/eDiscovery.png
EDISCOVERY/eDiscovery.desktop
EDISCOVERY/Search.glade
EDISCOVERY/icone_AddVANTE.jpg
EDISCOVERY/logo_AddVANTE.png
EDISCOVERY/time_gawk.log
EDISCOVERY/eDiscovery_petit.png
EDISCOVERY/Grep_eDiscovery.py
EDISCOVERY/mini.png
EDISCOVERY/
EDISCOVERY/eDiscovery.glade
EDISCOVERY/Gawk_eDiscovery.py
jordi@jordi-VirtualBox:~/Documents$
```

Image 52: Extract files from the compressed folder using the terminal.

You have to know the exact location where the scripts have been unzipped. If you do not know it, you can use the terminal command “cd” to enter to the proper folders until reaching the folder where the scripts have been extracted, and once there use the command “pwd” to look for the current directory.

```
jordi@jordi-VirtualBox:~$ ls
Baixades  Escriptori  glade  Música  Públic  test1.tiff
Documents examples.desktop Imatges Plantilles res.txt Videos
jordi@jordi-VirtualBox:~$ cd Documents/
jordi@jordi-VirtualBox:~/Documents$ ls
EDISCOVERY EDISCOVERY.tar.gz
jordi@jordi-VirtualBox:~/Documents$ cd EDISCOVERY/
jordi@jordi-VirtualBox:~/Documents/EDISCOVERY$ ls
addvante_logo.png      eDiscovery.png      logo_AddVANTE.png  Search.py
eDiscovery.desktop    Gawk_eDiscovery.py  logo_petit.png     time_gawk.log
eDiscovery.glade       Grep_eDiscovery.py  mini.png           time_grep.log
eDiscovery_petit.png  icone_AddVANTE.jpg  Search.glade
jordi@jordi-VirtualBox:~/Documents/EDISCOVERY$ pwd
/home/jordi/Documents/EDISCOVERY
jordi@jordi-VirtualBox:~/Documents/EDISCOVERY$
```

Image 53: Look the full path to the directory of the scripts with the command “pwd”

The path to the location of the scripts is required to make the script runnable from any location with the file “.desktop”. To make it ready just open the file “eDiscovery.desktop” with a text editor. First start the text editor and open the document from there, since if you try to open it by clicking on it it will just try to run the script.

Once you have the document opened it has to be modified using the proper path of the location of the scripts. The lines that have to be replaced are the ones that start with

“Exec=” and “Path=”, and the path that there is afterwards has to be replaced with the new location obtained using “pwd”. It is important not to modify the name of the script at the end of the line exec. Once done that just save the file with the same name.

```
[Desktop Entry]
Name=eDiscovery
Exec=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/EDISCOVERY/Grep_eDiscovery.py
Path=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/EDISCOVERY/
Terminal=false
Type=Application

[Desktop Entry]
Name=eDiscovery
Exec=/home/jordi/Documents/EDISCOVERY/Grep_eDiscovery.py
Path=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/EDISCOVERY/
Terminal=false
Type=Application

[Desktop Entry]
Name=eDiscovery
Exec=/home/jordi/Documents/EDISCOVERY/Grep_eDiscovery.py
Path=/home/jordi/Documents/EDISCOVERY/
Terminal=false
Type=Application
```

Image 54: Modify the document "eDiscovery.desktop" with the location of the script.

Once all of this is done, just move or copy the file “eDiscovery.desktop” to anywhere in order to execute the scripts with the GUI (Graphical User Interface). Take into account that if this is moved to the account of another user, make sure it have the rights to execute scripts, otherwise it will not work. Consider as well that the icon of this file won’t show the entire name, just “eDiscovery”.

USER GUIDE

To run this tool it is really simple. In order to start using the tool just make double click on the icon of the “eDiscovery” and this will start the tool (see the picture below). It can also be run if you are in the terminal at the folder with the scripts by typing the bash command “python search.py”.

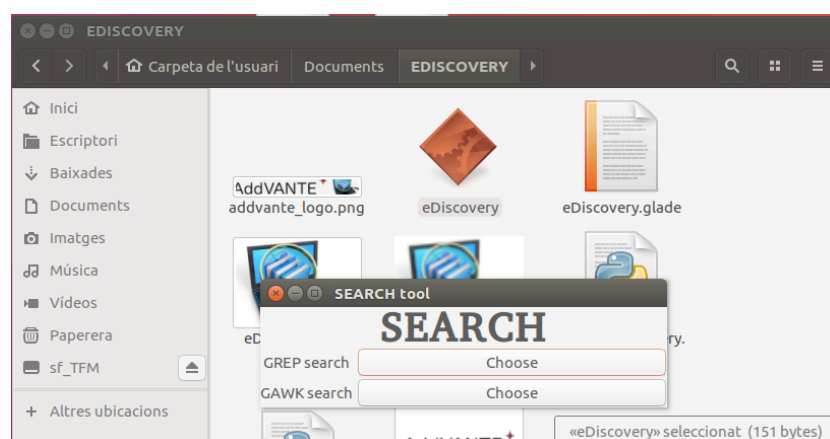


Image 55: Running the search tool with the GUI just clicking on "eDiscovery" icon.

The two options that appear are the 2 tools to perform the search. The first one “grep” is faster and more efficient, however it just accept one single word, so if the user wants to search more words will have to use the search several times. The second option “gawk” it is much slower but allows the user to perform logical searches so more than one word can be searched at the same time. The proper way of writing the words if the user wants

to use the logical search is the following: (/word1/ && /word2/) to perform a logical AND search with the 2 words and (/word1/ || /word2) to perform the search with a logical OR. More than one logical operator can be used but be careful to maintain the proper nomenclature.

Once selected the search tool, both tools look and work exactly the same way. The main window of the tool can be seen in the picture below:

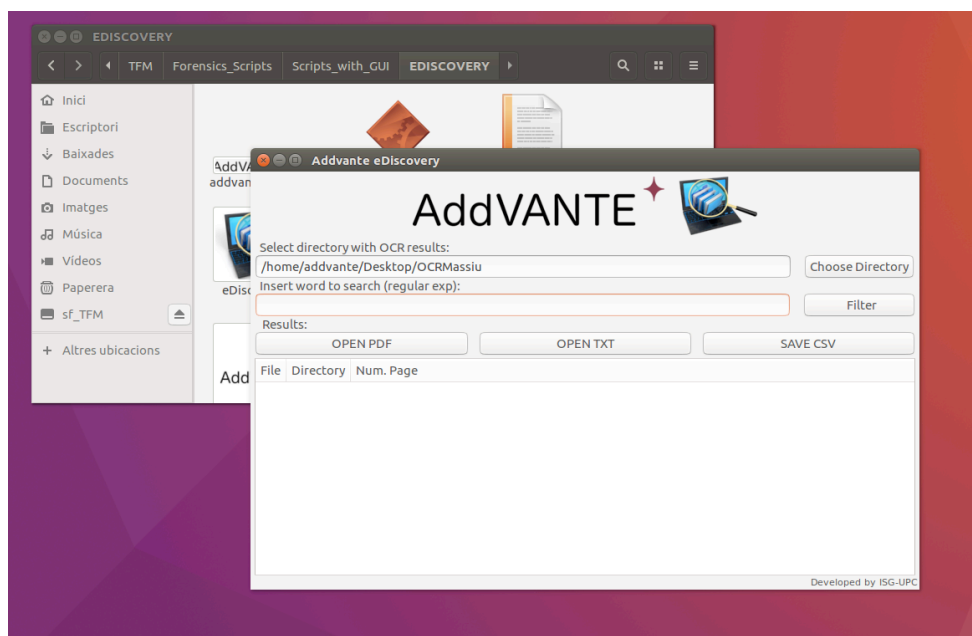


Image 56: Main window of the search tool once the search command has been selected.

As mentioned before, first of all the program requires to select an input folder with the txt files that the user wants to analyze. Remember to take into account that the names of those files must contain the path of the original file in the name (this is the folder that the OCR tool provides as its output).

To select this directory, it can be done in two ways: one simply type the path into the entry text (be careful of not miss-spelling the path if you use it like this) or the other is just clicking the button select, which will open another window to select the directory, and will fill the field automatically once the user clicks in the button "Select".

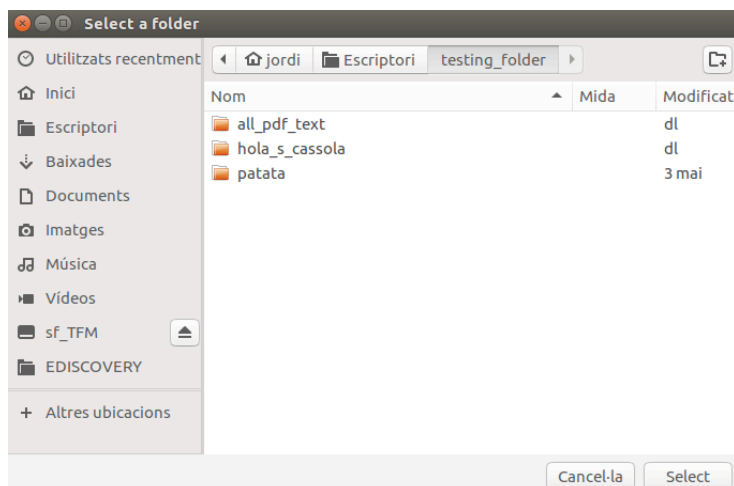


Image 57: Window to choose the path to the folder containing the txt files to analyze.

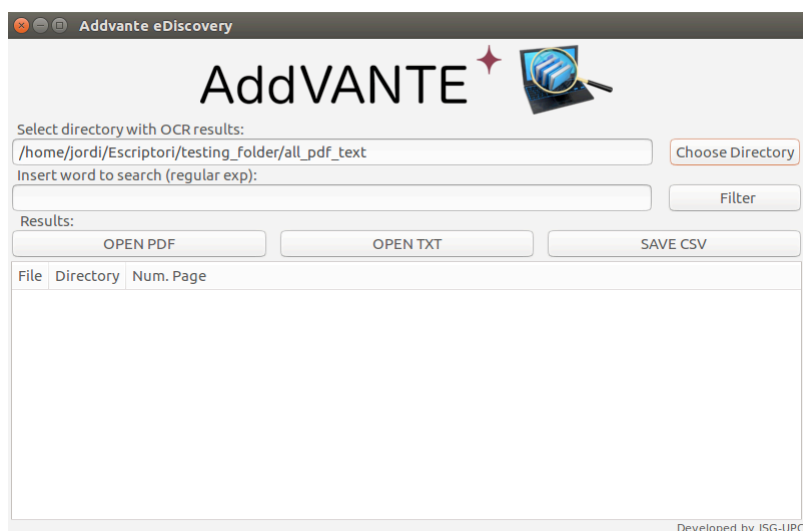


Image 58: The text entry is filled automatically when clicking "Select".

Once this is done, it can start to search which txt documents (and its associated pdf file before performing the OCR) contain certain key word. To do so it is really simple, just type the word you want to search into the field where it says "Insert the word to search (regular exp)" and press the button "Filter". This may take some time if there are lots of documents to analyze.



Image 59: Typing the word to search (in this example "computer") and pressing the filter button to start the search.

Once the search is done, the results will appear in a list below, where some information of them can be found: the name of the original document, the path to the original document and the number of the page where the word has been found.

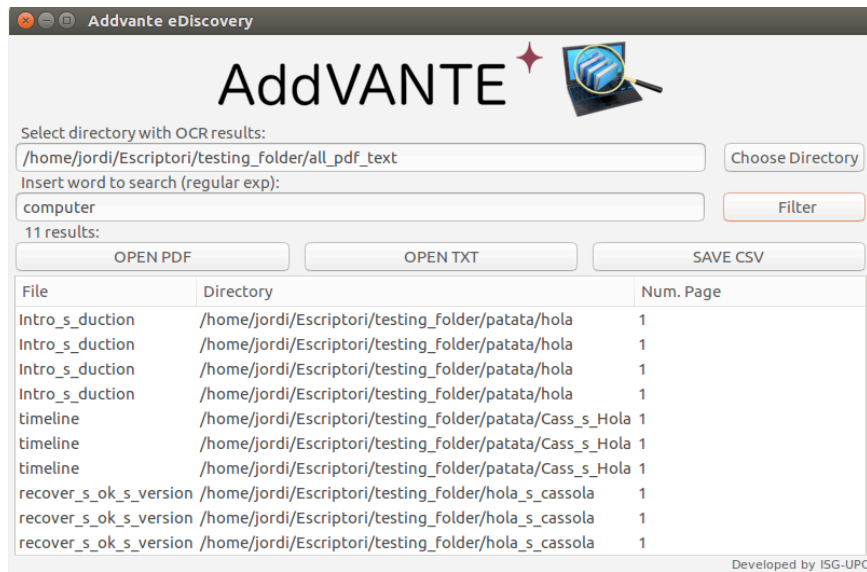


Image 60: Results shown after the search has finished.

As mentioned before, once the results are shown there are some options to do with them: clicking the button "open txt" will open with the default text editor the document where the word has been found, pressing the "open pdf" button will open the original pdf file and pressing the button "save csv" will create a document called "results_[the word searched]" and it will be saved at the folder containing the scripts. This document will contain the same information that is shown on the program.

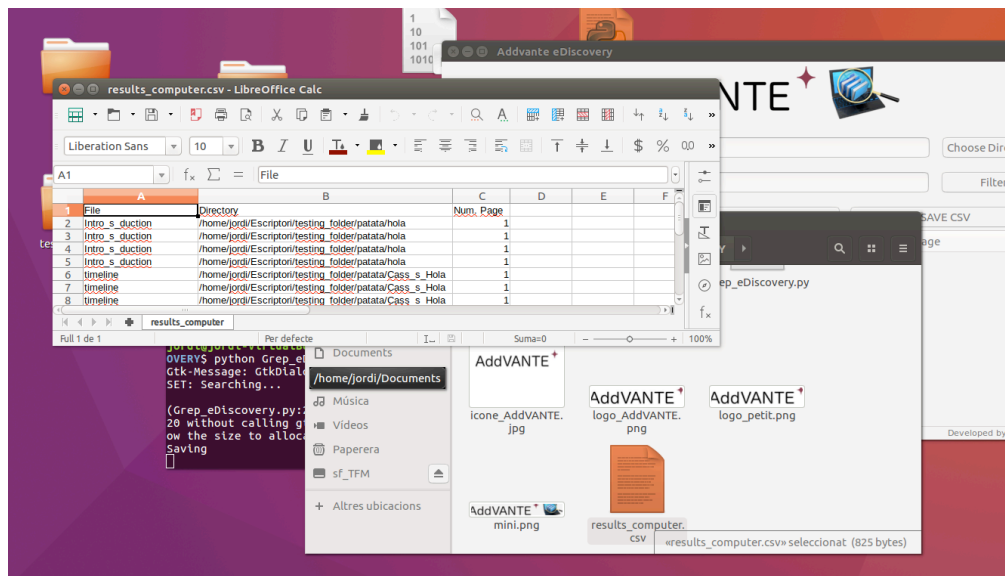


Image 61: Document with the results saved.

4. Results

In this part of the project we will evaluate the proper performance of each of the tools. To do so, we have decided to compute the time that each of the tools requires to analyze 1GB of information. We will compute the time of both using the GUI and using the terminal version.

	Binary Copy	Timeline	File Carving	OCR
GUI	~489s	~7s	~17s	Prepare ~0s Divide ~157s OCR ~427
Terminal	~473s	~7s	~17s	Prepare ~0s Divide ~147s OCR ~424s

Table 1: Time required for the tools to analyse 1GB of information.

As it can be observed, the difference between the time obtained using the script with the GUI is quite similar to the time obtained using the terminal. The main advantage of the first one is that it can be used by anybody without knowledge of digital forensic analysis. The advantage of the second one is that it can be run in background in the terminal while doing other things.

The eDiscovery tool has not been included in this table since it does not have the option of running it from the terminal. In this tool we have compared the two different ways if searching: using the Linux command “grep” or “gawk”. The time results are in the table below:

Grep	~0.07s
Gawk	~0.65s

Table 2: Grep and gawk time searching into 100 documents.

As it can be observed, using gawk instead of grep make the search 10 times slower. The advantage in using gawk instead of grep is that, as it has been mentioned during the project, it allows the user to perform logical searches with more than one word, whereas using grep it only allows to enter one single word.

5. Budget

In this point of the Master Thesis we are going to talk about the economic budget required to do this project taking into account the hours required to design and program all of the tools. We will talk as well of the economical cost of preparing the tool once all the scripts have been done and also the cost of the use of the tools.

First of all, lets see the costs of creating the tools. As it has been seen, there are 5 tools. In order to simplify the costs of designing and programming the tools, we will consider that the time needed in order to create each of the tools is 1 month. To get a brief idea of the cost that this might suppose we will consider that the salary of the telecommunications engineer is around 2000€/month. If we compute the cost of the creation of the tools this is: 5 tools * 1 tool/month * 2000€/month = 10000€. Since all of the tools use free software, there are no additional costs on this point.

Now lets talk about the installation cost. This part can also be done by another engineer but the time required to install the tools into a computer is really low since the procedure is really simple. This installation time is around 1-2 hours per computer. Assuming that the salary of this engineer may be between 15-20€/hour, we will neglect this cost taking into account the difference order of magnitude of this cost and the others of the project.

To end with the economical analysis, lets see the cost of operating this tools. In this point we will compute 2 costs in order to see the differences between the cost of a current forensic case and the cost using the tools by a person that is not a forensic specialist. The salary that a digital forensic specialist may earn is about 3000€/month whereas a normal employee may earn 1500€/moth. Taking that into account, if we consider that the required time to analyze the artifacts is "the same", in 7 months the money required to create the tool will be recovered.

In the table below, it can be observed the economic calculations that has been done to have a more clear sight of the amounts we are talking about:

Employees	Time	Salary	TOTAL	
Design and programming cost	1	5 month	2000€/month	10000€
Installation cost	1	1-2 hours	15€/hour	~30€
Use cost	1	-----	1500€/month	1500€/month

Table 3: Cost of the project.

# Month working	5	6	7	8
------------------------	---	---	---	---

This tool usr	7500€	9000€	10500€	12000€
Specialist	15000€	18000€	21000€	24000€
Difference	7500€	9000€	10500€	12000€

Table 4: Cost of the use of the tool.

So taking into consideration the results obtained from here, if the intention is that this tool will be used for more than 7 month, it will be economically viable.

IMPORTANT: these economical calculations have been done in order to prove the economic viability of the system. However, the aim of this project is not to replace the forensics specialist by people without knowledge plus this tool, but is to provide the possibility for places where there are no forensic specialist to perform investigations in a proper way.

6. Conclusions and future development

To end with this project, first of all let's make a brief summary of all the contents that have been commented on this project. We have started explaining what is digital forensics and the current tools that nowadays are being used to perform the analysis. Apart from that, we have taken a look into the different operations that have to be done when performing a forensic analysis. These tools that have been studied in this project are the binary copy, the timeline, the file carving, the OCR and the eDiscovery.

We have seen also the economic impact of implementing this tool. However, as it has already been commented, the aim of this project was to create some scripts with a GUI that will make the process easy in places where there are no digital forensic specialists.

Since all the tools that have been analyzed now, they all have its script with its corresponding GUI (which is friendly enough in order to be used by a non-specialist person that may need to perform the analysis), we can say that the first part of the project has been accomplished.

One of the results that has been observed is that this process is quite slow when the user is required to analyze or work with really huge amount of information and documents. These tools are prepared specially for this cases (since in a case that there is really few documents to analyze it could be done manually), however due to the really big amount of data that this tools may have to process, it can turn to be too slow.

Talking about the time required to perform these analysis, one recommendation if one day this project is continued could be to allow an option to allow the run in background of the terminal the script. This can be done easily with the Linux command "nohup". Actually that is the reason that there is a script to run the tools from the terminal without GUI. This has been tested with this script and it's really useful since the user can have a server performing the analysis in background while doing other things. The main problem with this is that the parameters like paths to the folder will have to be from the initial point in the script so it will be less flexible when having to operate with them.

Another recommendation that would be interesting to be done when continuing this project is to make more efficient the scripts. These scripts have been done in a simple way in order to be easy to use for the user, with the GUI, but it has not been taken much into account if this is the most efficient way of performing the actions. This may not save much time in small analyses but it will be noticed when working with really big amount of documents.

If we focus in specific scripts, some of them could also be enhanced such as the OCR, since now only converts pdf but with the same command it is possible to convert pictures so it may be another important thing to do when continuing the project. Apart from that, as it has been seen in the state of the art, there are some tools such as hash filter or tools for multimedia file analysis that has not been implemented and that could be interesting to be implemented in a future.

The second part of the project was to create a full platform into a virtual machine as it was proposed by Jordi Blanco in his project. As it was mentioned in the point 1.5 of this project (deviations from the initial plan), this part has not been done. All of those tools can

be used anyway without this platform, however creating it will make much more easy to operate with the tools as well as reducing the time needed from the administrator to prepare the tools.

Implementing that platform is one of the main important things to do when continuing the project. There are to reasons for it. The first one is that it will be much more easy to have all the files related to a certain case located, it could be done a full integrated tool with all the other tools integrated to it, and it will be much easier to investigate after having been analyzed. The second reason is that it would allow to put all the results on the folder according to its case and this folder will be related to the folder with the documents being investigated. This does not seem much important but it could allow creating scripts that just selecting the case, it could be run any of the tools in background since it will know the location of the documents and will not have to be entered by the user.

As Jordi Blanco suggested on his project, all of this tools could be integrated in the same platform. This platform could be a virtual machine with the Linux OS and could allow to use all these scripts that has been created in this project in order to create the platform as explained before in this point. Then this may solve the problem of remote digital forensics since distributing a virtual machine image is easy and will make the digital forensics accessible at any part of the world.

Bibliography

A thorough reference list such as that shown in the following examples: Conference paper [1], journal paper [2], book [3], standard-1 [4], standard-2 [5], online reference [6], patent [7], M.S. thesis [8] and Ph.D. dissertation [9].

- [1] J. Blanco, “Análisis de alternativas, desarrollo y puesta en marcha de una plataforma para análisis forense digital remoto”, Final Project, ETSETB, 2016.
- [2] Brian Carrier. “Open Source Digital Forensics Tools: The Legal Argument”. Stake inc, 2002.
- [3] “Python-Forensics”, 2017. [Online] Available: <http://python-forensics.org/>
- [4] Brian Carrier. “Autopsy”, 2017. [Online] Available: <https://www.sleuthkit.org/autopsy/>
- [5] “EnCase”. [Online] Available: <https://www.guidancesoftware.com/es/encase-forensic>
- [6] GitHub, “tesseract-ocr”, 2017. [Online] Available: <https://github.com/tesseract-ocr/tesseract/wiki>
- [7] CGSecurity, “PhotoRec”, 2016. [Online] Available: http://www.cgsecurity.org/wiki/PhotoRec_Data_Carving

Appendices

Appendix 1. Binary copy python code to work from the terminal

```
#!/usr/bin/env python
import os, logging, time

logging.basicConfig(filename='time_binarycopy_terminal.log', level=logging.DEBUG, format='%(asctime)s %(message)s')

print "Welcome to the UPC forensics app. This is the Binary Copy tool.\n"

devname=raw_input("\nEnter the device name (sdb1):\n")

destroute=raw_input("\nWrite the destination route (/home/jordi/escriptori):\n")

startTime = time.time()

#Create hash original device.
commandline="sudo md5sum /dev/"+devname+" > "+destroute+"/hashoriginal.txt"
os.system(commandline)

#Create binary copy.
commandline="sudo dd if=/dev/"+devname+" of="+destroute+"/image.dd"
os.system(commandline)

#Create hash binary copy.
commandline="sudo md5sum "+destroute+"/image.dd > "+destroute+"/hashcopy.txt"
os.system(commandline)

endTime = time.time()
duration = endTime - startTime

logging.info('Timeline Duration: ' + str(duration) + ' seconds')
```

Appendix 2. Binary copy python code to work with the GUI

```
#!/usr/bin/env python
import logging, time, sys, os, stat, time, hashlib, argparse, csv, gi
gi.require_version("Gtk", "3.0")
from gi.repository import Gtk
from gi.repository import Gdk

class binary_copy():
```

```

def __init__(self):
    self.builder=Gtk.Builder()
    self.builder.add_from_file("Binary_Copy.glade")
    self.win=self.builder.get_object("binary_copy_window")
    self.win.connect("delete-event",Gtk.main_quit)
    self.win.set_title("Binary Copy tool")
    self.win.set_resizable(True)
    self.win.set_default_size(400,50)
    self.win.move(400,200)
    copyButton = self.builder.get_object("but_copy")
    copyButton.connect("clicked", self.do_copy)
    button_res = self.builder.get_object("but_results")
    button_res.connect("clicked",self.get_result)

def run(self):
    self.win.show_all()
    Gtk.main()

def get_result(self, widget):
    self.open_dialog=Gtk.FileChooserDialog("Select a
folder",None,Gtk.FileChooserAction.SELECT_FOLDER,(Gtk.STOCK_CANCEL,
Gtk.ResponseType.CANCEL,"Select",Gtk.ResponseType.OK))
    self.open_dialog.set_default_size(800,400)
    self.TYPE="Results"

self.open_dialog.connect("response",self.open_dir_dialog)
    self.open_dialog.show()
    self.win.hide()

def do_copy(self, widget):
    logging.basicConfig(filename='time_binarycopy_gui.log',level=logging.DEBUG,format='%(asctime)s %(message)s')
    startTime = time.time()
    result_path = self.builder.get_object("text_results")
    destroute=result_path.get_text()
    dev = self.builder.get_object("text_dev")
    devname=dev.get_text()

```

```

        md5orcmd="sudo          md5sum          /dev/"+devname+"          >
"+destroute+"/hashoriginal.txt"

        os.system(md5orcmd)

        ddcmd="sudo          dd          if=/dev/"+devname+"
of="+destroute+"/image.dd"

        os.system(ddcmd)

        md5cpcmd="sudo          md5sum          "+destroute+"/image.dd          >
"+destroute+"/hashcopy.txt"

        os.system(md5cpcmd)

        endTime = time.time()

        duration = endTime - startTime

        logging.info('Binary Copy Terminal Duration: ' +
str(duration) + ' seconds')

    def open_dir_dialog(self, dialog, response_id):
        open_dialog=dialog

        if response_id==Gtk.ResponseType.OK:

self.directory=open_dialog.get_uri().replace("file://", "")

        self.open_dialog.hide()

        self.win.show()

        self.win.move(400,200)

        result_path =
self.builder.get_object("text_results")

        result_path.set_text(self.directory)

        elif response_id==Gtk.ResponseType.CANCEL:

        self.open_dialog.hide()

        self.win.show()

        self.win.move(400,200)

print "Welcome to the UPC forensics app. This is the Binary Copy
tool.\n"

binary_copy_script=binary_copy()

binary_copy_script.run()

```

Appendix 3. Binary copy GUI glade code

```
<?xml version="1.0" encoding="UTF-8"?>
```



```

<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.12"/>
  <object class="GtkWindow" id="binary_copy_window">
    <property name="can_focus">False</property>
    <child>
      <object class="GtkBox" id="box1">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <child>
          <object class="GtkLabel" id="T">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="label" translatable="yes">BINARY
COPY</property>
            <attributes>
              <attribute name="font-desc" value="Abyssinica SIL Bold
10"/>
              <attribute name="weight" value="ultraheavy"/>
              <attribute name="scale" value="3"/>
            </attributes>
          </object>
          <packing>
            <property name="expand">True</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkGrid">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <child>
              <object class="GtkLabel" id="label1">
                <property name="visible">True</property>
                <property name="can_focus">False</property>

```

```

        <property name="label"
translatable="yes">DEVICE</property>
    </object>
    <packing>
        <property name="left_attach">0</property>
        <property name="top_attach">0</property>
    </packing>
</child>
<child>
    <object class="GtkEntry" id="text_dev">
        <property name="visible">True</property>
        <property name="can_focus">True</property>
        <property name="hexpand">True</property>
        <property name="text"
translatable="yes">sdb1</property>
        <property name="placeholder_text"
translatable="yes">sdb1</property>
    </object>
    <packing>
        <property name="left_attach">1</property>
        <property name="top_attach">0</property>
    </packing>
</child>
</object>
<packing>
    <property name="expand">False</property>
    <property name="fill">True</property>
    <property name="position">1</property>
</packing>
</child>
<child>
    <object class="GtkGrid" id="grid1">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="row_spacing">1</property>
        <property name="column_spacing">4</property>
        <child>

```

```

        <object class="GtkLabel" id="label2">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="label" translatable="yes">Results
Directory</property>
        </object>
        <packing>
            <property name="left_attach">0</property>
            <property name="top_attach">0</property>
        </packing>
    </child>
    <child>
        <object class="GtkEntry" id="text_results">
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="hexpand">True</property>
        </object>
        <packing>
            <property name="left_attach">1</property>
            <property name="top_attach">0</property>
        </packing>
    </child>
    <child>
        <object class="GtkButton" id="but_results">
            <property name="label"
translatable="yes">Choose</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
        </object>
        <packing>
            <property name="left_attach">2</property>
            <property name="top_attach">0</property>
        </packing>
    </child>
</object>
<packing>

```

```

        <property name="expand">False</property>
        <property name="fill">True</property>
        <property name="position">2</property>
    </packing>
</child>
<child>
    <object class="GtkButton" id="but_copy">
        <property name="label" translatable="yes">MAKE BINARY COPY</property>
        <property name="visible">True</property>
        <property name="can_focus">True</property>
        <property name="receives_default">True</property>
    </object>
    <packing>
        <property name="expand">False</property>
        <property name="fill">True</property>
        <property name="position">3</property>
    </packing>
</child>
</object>
</child>
</object>
</interface>

```

Appendix 4. Binary copy desktop code

```

[Desktop Entry]
Name=Binary Copy
Exec=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/BINARY_COPY/Binary_Copy_gui.py
Path=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/BINARY_COPY/
Terminal=true
Type=Application

```

Appendix 5. File carving python code to work from the terminal

```
#!/usr/bin/env python
```

```

import os, logging, time

logging.basicConfig(filename='time_recover_terminal.log',level=logging.
DEBUG,format='%(asctime)s %(message)s')

print "Welcome to the UPC forensics app. This is the Recovery Files
tool.\n"

filename=raw_input("\nWrite the name of the file (ex. image.dd):\n")
fileroute=raw_input("\nWrite the file route (ex. /root/docs):\n")
destroute=raw_input("\nWrite the destination route (ex.
/root/results):\n")
#RECOVER FILES.
startTime = time.time()

commandline="photorec /debug /log /d "+destroute+"/Recovered_Files /cmd
"+fileroute+"/"+filename+"
partition_none,options,mode_ext2,fileopt,everything,enable,search"
os.system(commandline);

endTime = time.time()
duration = endTime - startTime
logging.info('Duration: ' + str(duration) + ' seconds')

```

Appendix 6. File carving python code to work with the GUI

```

#!/usr/bin/env python
import os, gi, time, logging
gi.require_version("Gtk", "3.0")
from gi.repository import Gtk
from gi.repository import Gdk

class recover_files():
    TYPE=""
    def __init__(self):
        self.builder=Gtk.Builder()
        self.builder.add_from_file("recover.glade")
        self.win=self.builder.get_object("Recover_window")
        self.win.connect("delete-event",Gtk.main_quit)
        self.win.set_title("Recover tool")
        self.win.set_resizable(True)

```

```

        self.win.set_default_size(400,50)
        self.win.move(400,200)
        filButton = self.builder.get_object("but_recover")
        filButton.connect("clicked", self.do_recover)

        button_dir =
self.builder.get_object("but_choose_image")
        button_dir.connect("clicked",self.get_image)

        button_res =
self.builder.get_object("but_choose_results")
        button_res.connect("clicked",self.get_result)

    def run(self):
        self.win.show_all()
        Gtk.main()

    def get_image(self, widget):
        self.open_dialog=Gtk.FileChooserDialog("Select an
image",None,Gtk.FileChooserAction.OPEN,(Gtk.STOCK_CANCEL,
Gtk.ResponseType.CANCEL,"Select",Gtk.ResponseType.OK))
        self.open_dialog.set_default_size(800,400)
        self.TYPE="IM"

self.open_dialog.connect("response",self.open_dir_dialog)
        self.open_dialog.show()
        self.win.hide()

    def get_result(self, widget):
        self.open_dialog=Gtk.FileChooserDialog("Select a
folder",None,Gtk.FileChooserAction.SELECT_FOLDER,(Gtk.STOCK_CANCEL,
Gtk.ResponseType.CANCEL,"Select",Gtk.ResponseType.OK))
        self.open_dialog.set_default_size(800,400)
        self.TYPE="RES"

self.open_dialog.connect("response",self.open_dir_dialog)
        self.open_dialog.show()
        self.win.hide()

    def do_recover(self, widget):

```

```

logging.basicConfig(filename='time_recover_gui.log', level=logging.DEBUG
,format='%(asctime)s %(message)s')

        startTime = time.time()

        result_path
self.builder.get_object("entry_result_path")
        txt_result_path=result_path.get_text()

        image_path
self.builder.get_object("entry_image_path")
        txt_image_path=image_path.get_text()

        commandline="photorec          /debug          /log          /d
"+txt_result_path+"/Recovered_Files          /cmd          "+txt_image_path+"
partition_none,options,mode_ext2,fileopt,everything,enable,search"

        os.system(commandline)

        endTime = time.time()

        duration = endTime - startTime

        logging.info('Duration: ' + str(duration) + ' seconds')


def open_dir_dialog(self,dialog,response_id):
    open_dialog=dialog

    if response_id==Gtk.ResponseType.OK:

self.directory=open_dialog.get_uri().replace("file://","")

        self.open_dialog.hide()

        self.win.show()

        self.win.move(400,200)

        if self.TYPE=="RES":

                result_path
self.builder.get_object("entry_result_path")
                result_path.set_text(self.directory)

                elif self.TYPE=="IM":

                        image_path
self.builder.get_object("entry_image_path")
                        image_path.set_text(self.directory)

                elif response_id==Gtk.ResponseType.CANCEL:

                        self.open_dialog.hide()

                        self.win.show()

                        self.win.move(400,200)

print "Welcome to the UPC forensics app. This is the Recovery Files
tool.\n"

```

```
recover_script=recover_files()  
recover_script.run()
```

Appendix 7. File carving GUI glade code

```
<?xml version="1.0" encoding="UTF-8"?>  
<!-- Generated with glade 3.18.3 -->  
<interface>  
  <requires lib="gtk+" version="3.12"/>  
  <object class="GtkWindow" id="Recover_window">  
    <property name="can_focus">False</property>  
    <child>  
      <object class="GtkBox">  
        <property name="visible">True</property>  
        <property name="can_focus">False</property>  
        <property name="orientation">vertical</property>  
        <child>  
          <object class="GtkLabel">  
            <property name="visible">True</property>  
            <property name="can_focus">False</property>  
            <property name="label" translatable="yes">RECOVER  
TOOL</property>  
            <attributes>  
              <attribute name="font-desc" value="Abyssinica SIL Bold  
10"/>  
              <attribute name="weight" value="ultraheavy"/>  
              <attribute name="scale" value="3"/>  
            </attributes>  
          </object>  
          <packing>  
            <property name="expand">False</property>  
            <property name="fill">True</property>  
            <property name="position">0</property>  
          </packing>  
        </child>  
        <child>  
          <object class="GtkGrid">
```



```

    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <child>
      <object class="GtkLabel">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label" translatable="yes">Image
directory:</property>
      </object>
      <packing>
        <property name="left_attach">0</property>
        <property name="top_attach">0</property>
      </packing>
    </child>
    <child>
      <object class="GtkLabel">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label" translatable="yes">Recover
results:</property>
      </object>
      <packing>
        <property name="left_attach">0</property>
        <property name="top_attach">1</property>
      </packing>
    </child>
    <child>
      <object class="GtkEntry" id="entry_image_path">
        <property name="visible">True</property>
        <property name="can_focus">True</property>
        <property name="hexpand">True</property>
      </object>
      <packing>
        <property name="left_attach">1</property>
        <property name="top_attach">0</property>
      </packing>
    </child>

```

```

    <child>
      <object class="GtkEntry" id="entry_result_path">
        <property name="visible">True</property>
        <property name="can_focus">True</property>
      </object>
      <packing>
        <property name="left_attach">1</property>
        <property name="top_attach">1</property>
      </packing>
    </child>
    <child>
      <object class="GtkButton" id="but_choose_image">
        <property name="label"
translatable="yes">Choose</property>
        <property name="visible">True</property>
        <property name="can_focus">True</property>
        <property name="receives_default">True</property>
      </object>
      <packing>
        <property name="left_attach">2</property>
        <property name="top_attach">0</property>
      </packing>
    </child>
    <child>
      <object class="GtkButton" id="but_choose_results">
        <property name="label"
translatable="yes">Choose</property>
        <property name="visible">True</property>
        <property name="can_focus">True</property>
        <property name="receives_default">True</property>
      </object>
      <packing>
        <property name="left_attach">2</property>
        <property name="top_attach">1</property>
      </packing>
    </child>
  </object>

```

```

        <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">1</property>
        </packing>
    </child>
    <child>
        <object class="GtkButton" id="but_recover">
            <property name="label"
translatable="yes">RECOVER</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
        </object>
        <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">2</property>
        </packing>
    </child>
</object>
</child>
</object>
</interface>

```

Appendix 8. File carving desktop code

```

[Desktop Entry]
Name=Recover
Exec=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/RECO
VER/Recover_gui.py
Path=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/RECO
VER/
Terminal=false
Type=Application

```

Appendix 9. Timeline python code to work from the terminal

```
#!/usr/bin/env python
import logging, time, sys, os, stat, time, hashlib, argparse, csv

def WalkPath(rootPath, reportPath):
    processCount = 0
    errorCount = 0
    reportPath = os.path.join(reportPath, "Timeline_Results.csv")
    oCVS = _CSVWriter(reportPath)
    if rootPath.endswith('\\') or rootPath.endswith('/'):
        rootPath = rootPath
    else:
        rootPath = rootPath+'/'
    for root, dirs, files in os.walk(rootPath):
        for file in files:
            fname = os.path.join(root, file)
            result = getInfo(fname, file, oCVS)
            if result is True:
                processCount += 1
            else:
                errorCount += 1
    oCVS.writerClose()
    return(processCount)

def getInfo(theFile, simpleName, o_result):
    if os.path.exists(theFile):
        if not os.path.islink(theFile):
            if os.path.isfile(theFile):
                try:
                    f = open(theFile, 'rb')
                except IOError:
                    logging.warning('Open Failed: '
+ theFile)
                    return
            else:
                try:
                    theFileStats =
```

```

os.stat(theFile)
                                (mode, ino, dev, nlink,
uid, gid, size, atime, mtime, ctime) = os.stat(theFile)
                                rd = f.read()
                                except IOError:
                                    f.close()
                                    logging.warning('File
Access Error: ' + theFile)

                                return

                                else:
                                    fileSize = str(size)
                                    modifiedTime =
time.ctime(mtime)
                                    accessTime =
time.ctime(atime)
                                    createdTime =
time.ctime(ctime)
                                    ownerID = str(uid)
                                    groupID = str(gid)
                                    fileMode = bin(mode)
                                    resultList =
[simpleName, theFile, fileSize, modifiedTime, accessTime, createdTime,
ownerID, groupID, str(mode)]

o_result.writeCSVRow(resultList)

                                return True

                                else:
                                    logging.warning('[' + repr(simpleName)
+ ', Skipped NOT a File' + ']')
                                    return False

                                else:
                                    logging.warning('[' + repr(simpleName) + ',
Skipped Link NOT a File' + ']')
                                    return False

                                else:
                                    logging.warning('[' + repr(simpleName) + ', Path does
NOT exist' + ']')
                                    return False

class _CSVWriter:

```

```

def __init__(self, fileName):
    try:
        if (sys.version_info > (3, 0)):
            self.csvFile = open(fileName,
'w',newline="\r\n")

        else:
            self.csvFile = open(fileName, 'w')

        tempList = ['File', 'Path', 'Size', 'Modified
Time', 'Access Time', 'Created Time', 'Owner', 'Group', 'Mode']
        outStr = ",".join(tempList)
        self.csvFile.write(outStr)
        self.csvFile.write("\n")

    except:
        logging.error('CSV File Open Failure')
        quit()

def writeCSVRow(self, outList):
    outStr = ",".join(outList)
    self.csvFile.write(outStr)
    self.csvFile.write("\n")

def writerClose(self):
    self.csvFile.close()

logging.basicConfig(filename='time_timeline_terminal.log',level=logging
.DEBUG,format='%(asctime)s %(message)s')

print "Welcome to the UPC forensics app. This is the Timeline tool.\n"
rootPath=raw_input("\nWrite the root folder path (ex. /root/docs):\n")
reportPath=raw_input("\nWrite the destination path (ex.
/root/results):\n")

startTime = time.time()
filesProcessed = WalkPath(rootPath, reportPath)
endTime = time.time()
duration = endTime - startTime
logging.info('Timeline Duration: ' + str(duration) + ' seconds')

```

Appendix 10. Timeline python code to work with the GUI

```
#!/usr/bin/env python
```

```

import logging, time, sys, os, stat, time, hashlib, argparse, csv, gi
gi.require_version("Gtk", "3.0")
from gi.repository import Gtk
from gi.repository import Gdk

class timeline():
    TYPE=""

    def __init__(self):
        self.builder=Gtk.Builder()
        self.builder.add_from_file("Timeline.glade")
        self.win=self.builder.get_object("timeline_window")
        self.win.connect("delete-event",Gtk.main_quit)
        self.win.set_title("Timeline tool")
        self.win.set_resizable(True)
        self.win.set_default_size(400,50)
        self.win.move(400,200)

        filtButton = self.builder.get_object("but_analyse")
        filtButton.connect("clicked", self.do_analyse)
        button_dir = self.builder.get_object("but_folder")
        button_dir.connect("clicked",self.get_folder)
        button_res = self.builder.get_object("but_results")
        button_res.connect("clicked",self.get_result)

    def run(self):
        self.win.show_all()
        Gtk.main()

    def get_folder(self, widget):
        self.open_dialog=Gtk.FileChooserDialog("Select folder",None,Gtk.FileChooserAction.SELECT_FOLDER, (Gtk.STOCK_CANCEL,
Gtk.ResponseType.CANCEL,"Select",Gtk.ResponseType.OK))
        self.open_dialog.set_default_size(800,400)
        self.TYPE="Path"

self.open_dialog.connect("response",self.open_dir_dialog)
        self.open_dialog.show()
        self.win.hide()

```

```

    def get_result(self, widget):
        self.open_dialog=Gtk.FileChooserDialog("Select a
        folder",None,Gtk.FileChooserAction.SELECT_FOLDER,(Gtk.STOCK_CANCEL,
        Gtk.ResponseType.CANCEL,"Select",Gtk.ResponseType.OK))
        self.open_dialog.set_default_size(800,400)
        self.TYPE="Results"

self.open_dialog.connect("response",self.open_dir_dialog)

        self.open_dialog.show()
        self.win.hide()

    def do_analyse(self, widget):

logging.basicConfig(filename='time_timeline_gui.log',level=logging.DEBU
G,format='%(asctime)s %(message)s')

        startTime = time.time()
        result_path = self.builder.get_object("text_results")
        txt_result_path=result_path.get_text()
        folder_path = self.builder.get_object("text_folder")
        txt_folder_path=folder_path.get_text()
        filesProcessed      =      self.WalkPath(txt_folder_path,
txt_result_path)

        endTime = time.time()
        duration = endTime - startTime
        logging.info('Duration: ' + str(duration) + ' seconds')

    def open_dir_dialog(self,dialog,response_id):
        open_dialog=dialog

        if response_id==Gtk.ResponseType.OK:

self.directory=open_dialog.get_uri().replace("file://","")
        self.open_dialog.hide()
        self.win.show()
        self.win.move(400,200)
        if self.TYPE=="Path":
            folder_path
            =
self.builder.get_object("text_folder")
            folder_path.set_text(self.directory)

```



```

        elif self.TYPE=="Results":
            result_path =
self.builder.get_object("text_results")
            result_path.set_text(self.directory)
        elif response_id==Gtk.ResponseType.CANCEL:
            self.open_dialog.hide()
            self.win.show()
            self.win.move(400,200)

    def WalkPath(self, rootPath, reportPath):
        processCount = 0
        errorCount = 0
        reportPath = os.path.join(reportPath,
"Timeline_Results.csv")
        oCVS = _CSVWriter(reportPath)
        if rootPath.endswith('\\') or rootPath.endswith('/'):
            rootPath = rootPath
        else:
            rootPath = rootPath+'/'
        for root, dirs, files in os.walk(rootPath):
            for file in files:
                fname = os.path.join(root, file)
                result = self.getInfo(fname, file,
oCVS)

                if result is True:
                    processCount += 1
                else:
                    errorCount += 1

        oCVS.writerClose()
        return(processCount)

    def getInfo(self, theFile, simpleName, o_result):
        if os.path.exists(theFile):
            if not os.path.islink(theFile):
                if os.path.isfile(theFile):
                    try:
                        f = open(theFile, 'rb')

```

```

                                except IOError:
                                    logging.warning('Open
Failed: ' + theFile)

                                return

                                else:

                                    try:

                                        theFileStats =
os.stat(theFile)

                                        (mode, ino,
dev, nlink, uid, gid, size, atime, mtime, ctime) = os.stat(theFile)

                                        rd = f.read()

                                except IOError:
                                    f.close()

                                logging.warning('File Access Error: ' + theFile)

                                return

                                else:

                                    fileSize =
str(size)

                                    modifiedTime =
time.ctime(mtime)

                                    accessTime =
time.ctime(atime)

                                    createdTime =
time.ctime(ctime)

                                    ownerID =
str(uid)

                                    groupID =
str(gid)

                                    fileMode =
bin(mode)

                                    resultList =
[simpleName, theFile, fileSize, modifiedTime, accessTime, createdTime,
ownerID, groupID, str(mode)]

                                o_result.writeCSVRow(resultList)

                                return True

                                else:

                                    logging.warning('[' +
repr(simpleName) + ', Skipped NOT a File' + ']')

                                return False

```

```

        else:
            logging.warning([' + repr(simpleName)
+ ', Skipped Link NOT a File' + ''])
            return False

    else:
        logging.warning([' + repr(simpleName) + ',
Path does NOT exist' + ''])
        return False

class _CSVWriter:
    def __init__(self, fileName):
        try:
            if (sys.version_info > (3, 0)):
                self.csvFile = open(fileName,
'w',newline="\r\n")
            else:
                self.csvFile = open(fileName, 'w')

            tempList = ['File', 'Path', 'Size', 'Modified
Time', 'Access Time', 'Created Time', 'Owner', 'Group', 'Mode']
            outStr = ",".join(tempList)
            self.csvFile.write(outStr)
            self.csvFile.write("\n")

        except:
            logging.error('CSV File Open Failure')
            quit()

    def writeCSVRow(self, outList):
        outStr = ",".join(outList)
        self.csvFile.write(outStr)
        self.csvFile.write("\n")

    def writerClose(self):
        self.csvFile.close()

print "Welcome to the UPC forensics app. This is the Timeline tool.\n"
timeline_script=timeline()
timeline_script.run()

```

Appendix 11. Timeline GUI glade code

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.18.3 -->
<interface>
  <requires lib="gtk+" version="3.12"/>
  <object class="GtkWindow" id="timeline_window">
    <property name="can_focus">False</property>
    <child>
      <object class="GtkBox" id="box1">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <child>
          <object class="GtkLabel" id="T">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="label"
translatable="yes">TIMELINE</property>
            <attributes>
              <attribute name="font-desc" value="Abyssinica SIL Bold
10"/>
              <attribute name="weight" value="ultraheavy"/>
              <attribute name="scale" value="3"/>
            </attributes>
          </object>
          <packing>
            <property name="expand">True</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkGrid" id="grid1">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="row_spacing">1</property>
            <property name="column_spacing">4</property>
```

```

<child>
  <object class="GtkEntry" id="text_folder">
    <property name="visible">True</property>
    <property name="can_focus">True</property>
    <property name="hexpand">True</property>
  </object>
  <packing>
    <property name="left_attach">1</property>
    <property name="top_attach">0</property>
  </packing>
</child>
<child>
  <object class="GtkEntry" id="text_results">
    <property name="visible">True</property>
    <property name="can_focus">True</property>
    <property name="hexpand">True</property>
  </object>
  <packing>
    <property name="left_attach">1</property>
    <property name="top_attach">1</property>
  </packing>
</child>
<child>
  <object class="GtkButton" id="but_folder">
    <property name="label"
translatable="yes">Choose</property>
    <property name="visible">True</property>
    <property name="can_focus">True</property>
    <property name="receives_default">True</property>
  </object>
  <packing>
    <property name="left_attach">2</property>
    <property name="top_attach">0</property>
  </packing>
</child>
<child>
  <object class="GtkButton" id="but_results">

```

```

        <property name="label"
translatable="yes">Choose</property>
        <property name="visible">True</property>
        <property name="can_focus">True</property>
        <property name="receives_default">True</property>
    </object>
    <packing>
        <property name="left_attach">2</property>
        <property name="top_attach">1</property>
    </packing>
</child>
<child>
    <object class="GtkLabel" id="label1">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label" translatable="yes">Folder
Directory</property>
    </object>
    <packing>
        <property name="left_attach">0</property>
        <property name="top_attach">0</property>
    </packing>
</child>
<child>
    <object class="GtkLabel" id="label2">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label" translatable="yes">Results
Directory</property>
    </object>
    <packing>
        <property name="left_attach">0</property>
        <property name="top_attach">1</property>
    </packing>
</child>
</object>
<packing>

```

```

        <property name="expand">False</property>
        <property name="fill">True</property>
        <property name="position">1</property>
    </packing>
</child>
<child>
    <object class="GtkButton" id="but_analyse">
        <property name="label"
translatable="yes">ANALYSE</property>
        <property name="visible">True</property>
        <property name="can_focus">True</property>
        <property name="receives_default">True</property>
    </object>
    <packing>
        <property name="expand">False</property>
        <property name="fill">True</property>
        <property name="position">2</property>
    </packing>
</child>
</object>
</child>
</object>
</interface>

```

Appendix 12. Timeline desktop code

```

[Desktop Entry]
Name=Timeline
Exec=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/TIME
LINE/Timeline_gui.py
Path=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/TIME
LINE/
Terminal=false
Type=Application

```

Appendix 13. OCR python code to work from the terminal

```
#!/usr/bin/env python
```

```

import os, time, logging, pyPdf
from pyPdf import PdfFileReader, PdfFileWriter

def check_order(dir):
    order=raw_input("\nType an order (options: PREPARE, DIVIDE,
OCR, EXIT):\n")
    if order=="PREPARE":
        startTime = time.time()
        replace_space(dir)
        endTime = time.time()
        duration = endTime - startTime
        logging.info('PREPARE Duration: ' + str(duration) + '
seconds')

        print "\nPREPARATION DONE!\n"
        check_order(dir)
    elif order=="DIVIDE":
        startTime = time.time()
        divide(dir)
        endTime = time.time()
        duration = endTime - startTime
        logging.info('DIVIDE Duration: ' + str(duration) + '
seconds')

        print "\nDIVISION DONE!\n"
        check_order(dir)
    elif order=="OCR":
        startTime = time.time()
        ocr(dir+"/all_pdf_text/")
        endTime = time.time()
        duration = endTime - startTime
        logging.info('OCR Duration: ' + str(duration) + '
seconds')

        print "\nOCR DONE!\n"
        check_order(dir)
    elif order=="EXIT":
        print "Goodbye"
    else:
        print "\nERROR!\n"
        check_order(dir)

```



```

def replace_space(dir):
    for root, dirs, files in os.walk(dir):
        for dire in dirs:
            if " " in dire:
                os.rename(os.path.join(root, dire),
os.path.join(root, dire.replace(" ", "_s_")))
                dire=dire.replace(" ", "_s_")
                self.replace_space(dir)
                break
        for file in files:
            if " " in file:
                os.rename(os.path.join(root, file),
os.path.join(root, file.replace(" ", "_s_")))
                file=file.replace(" ", "_s_")

def divide(dir):
    cmd="mkdir "+dir+"/all_pdf_text"
    os.system(cmd)
    for root, dirs, files in os.walk(dir):
        for file in files:
            if file.endswith(".pdf"):
                fname = os.path.join(root, file)

py_pdf_file=PdfFileReader(open(fname,'rb'))

                if py_pdf_file.isEncrypted:
                    numpages=-1

                else:

numpages=py_pdf_file.getNumPages()

                if numpages!=-1:
                    cvcmd="pdfseparate "+fname+"
"+dir+"/all_pdf_text/"+(fname.replace("/", "_b_"))+"-%d.pdf"
                    os.system(cvcmd)

def ocr(dir):
    for root, dirs, files in os.walk(dir):

```

```

        for file in files:
            if file.endswith(".pdf"):
                fname=os.path.join(root,file)
                cv_cmd="convert -monochrome -density
300 "+fname+" "+fname[:-3]+"tiff"
                os.system(cv_cmd)
                ocr_cmd="tesseract "+fname[:-3]+"tiff
"+fname[:-4]
                os.system(ocr_cmd)
                rmpdf_cmd="rm "+fname
                os.system(rmpdf_cmd)
                rmtiff_cmd="rm "+fname[:-3]+"tiff"
                os.system(rmtiff_cmd)

logging.basicConfig(filename='time_ocr_terminal.log',level=logging.DEBU
G,format='%(asctime)s %(message)s')

print "Welcome to the UPC forensics app. This is the Massive OCR
tool.\n"

dir=raw_input("\nWrite the directory path (ex. /root/docs):\n")
check_order(dir)

```

Appendix 14. OCR python code to work with the GUI

```

#!/usr/bin/env python
import os, gi, time, logging, pyPdf
gi.require_version("Gtk","3.0")
from gi.repository import Gtk
from gi.repository import Gdk
from pyPdf import PdfFileReader, PdfFileWriter

class ocr():
    def __init__(self):
        self.builder=Gtk.Builder()
        self.builder.add_from_file("ocr.glade")
        self.win=self.builder.get_object("Ocr_window")
        self.win.connect("delete-event",Gtk.main_quit)
        self.win.set_title("OCR tool")

```

```

        self.win.set_resizable(True)
        self.win.set_default_size(400,50)
        self.win.move(400,200)
        chooseButton = self.builder.get_object("but_choose")
        chooseButton.connect("clicked",self.get_folder)
        prepareButton = self.builder.get_object("but_prepare")
        prepareButton.connect("clicked", self.do_prepare)
        divideButton = self.builder.get_object("but_divide")
        divideButton.connect("clicked",self.do_divide)
        ocrButton = self.builder.get_object("but_ocr")
        ocrButton.connect("clicked",self.do_ocr)

    def run(self):
        self.win.show_all()
        Gtk.main()

    def get_folder(self, widget):
        self.open_dialog=Gtk.FileChooserDialog("Select a
        folder",None,Gtk.FileChooserAction.SELECT_FOLDER, (Gtk.STOCK_CANCEL,
        Gtk.ResponseType.CANCEL,"Select",Gtk.ResponseType.OK))
        self.open_dialog.set_default_size(800,400)
self.open_dialog.connect("response",self.open_dir_dialog)
        self.open_dialog.show()
        self.win.hide()

    def open_dir_dialog(self,dialog,response_id):
        open_dialog=dialog
        if response_id==Gtk.ResponseType.OK:
self.directory=open_dialog.get_uri().replace("file://","")
            self.open_dialog.hide()
            self.win.show()
            self.win.move(400,200)
            folder_path =
self.builder.get_object("entry_folder")
            folder_path.set_text(self.directory)
            text_prepare =

```

```

self.builder.get_object("text_prepare")
    text_prepare.set_text("READY!")
    text_divide
self.builder.get_object("text_divide")
    text_divide.set_text("do prepare first")
    text_ocr = self.builder.get_object("text_ocr")
    text_ocr.set_text("do divide first")

    elif response_id==Gtk.ResponseType.CANCEL:
        self.open_dialog.hide()
        self.win.show()
        self.win.move(400,200)

def do_prepare(self, widget):
    text_prepare = self.builder.get_object("text_prepare")
    state=text_prepare.get_text()
    folder_path = self.builder.get_object("entry_folder")

    if folder_path.get_text()=="":
        state=""
    else:
        state="READY!"

    if state=="READY!":
        startTime = time.time()
        self.replace_space(folder_path.get_text())
        endTime = time.time()
        duration = endTime - startTime
        logging.info('PREPARE      Duration:      ' +
str(duration) + ' seconds')
        text_prepare.set_text("DONE!")
        text_divide
self.builder.get_object("text_divide")
        text_divide.set_text("READY!")
    else:

error_dialog=Gtk.MessageDialog(None,0,Gtk.MessageType.ERROR,Gtk.Buttons
Type.CANCEL,"ERROR")

        error_dialog.format_secondary_text("Select      a
folder first.")

```

```

        error_dialog.run()
        error_dialog.destroy()

    def do_divide(self, widget):
        text_divide = self.builder.get_object("text_divide")
        state=text_divide.get_text()
        folder_path = self.builder.get_object("entry_folder")
        dire=folder_path.get_text()
        if state=="READY!":
            startTime = time.time()
            self.divide(dire)
            endTime = time.time()
            duration = endTime - startTime
            logging.info('DIVIDE      Duration:      ' +
str(duration) + ' seconds')
            text_divide.set_text("DONE!")
            text_ocr = self.builder.get_object("text_ocr")
            text_ocr.set_text("READY!")
        else:

error_dialog=Gtk.MessageDialog (None,0,Gtk.MessageType.ERROR,Gtk.Buttons
Type.CANCEL,"ERROR")

        error_dialog.format_secondary_text("Select      a
folder first.")

        error_dialog.run()
        error_dialog.destroy()

    def do_ocr(self, widget):
        text_ocr = self.builder.get_object("text_ocr")
        state=text_ocr.get_text()
        folder_path = self.builder.get_object("entry_folder")
        dire=folder_path.get_text()
        if state=="READY!":
            startTime = time.time()
            self.ocr(dire+"/all_pdf_text/")
            endTime = time.time()
            duration = endTime - startTime

```

```

        logging.info('OCR Duration: ' + str(duration) +
' seconds')

        text_ocr.set_text("DONE!")

    else:

error_dialog=Gtk.MessageDialog (None,0,Gtk.MessageType.ERROR,Gtk.Buttons
Type.CANCEL,"ERROR")

        error_dialog.format_secondary_text("Select      a
folder first.")

        error_dialog.run()

        error_dialog.destroy()


    def replace_space(self, dir):

        for root, dirs, files in os.walk(dir):

            for dire in dirs:

                if " " in dire:

                    os.rename(os.path.join(root,
dire), os.path.join(root, dire.replace(" ", "_s_")))

                    dire=dire.replace(" ", "_s_")

                    self.replace_space(dire)

                    break

            for file in files:

                if " " in file:

                    os.rename(os.path.join(root,
file), os.path.join(root, file.replace(" ", "_s_")))

                    file=file.replace(" ", "_s_")


    def divide(self, dir):

        cmd="mkdir "+dir+"/all_pdf_text"

        os.system(cmd)

        for root, dirs, files in os.walk(dir):

            for file in files:

                if file.endswith(".pdf"):

                    fname      =      os.path.join(root,
file)

py_pdf_file=PdfFileReader(open(fname,'rb'))

                    if py_pdf_file.isEncrypted:

                        numpages=-1

```

```

else:

numpages=py_pdf_file.getNumPages()

    if numpages!=-1:
        cvcmd="pdfseparate
"+fname+" "+dir+"/all_pdf_text/"+(fname.replace("/","_b_"))+"-%d.pdf"
        os.system(cvcmd)

def ocr(self, dir):
    for root, dirs, files in os.walk(dir):
        for file in files:
            if file.endswith(".pdf"):
                fname=os.path.join(root,file)
                cv_cmd="convert -monochrome -
density 300 "+fname+" "+fname[:-3]+"tiff"
                os.system(cv_cmd)
                ocr_cmd="tesseract "+fname[:-
3]+"tiff "+fname[:-4]
                os.system(ocr_cmd)
                rmpdf_cmd="rm "+fname
                os.system(rmpdf_cmd)
                rmtiff_cmd="rm "+fname[:-
3]+"tiff"
                os.system(rmtiff_cmd)

logging.basicConfig(filename='time_ocr_gui.log',level=logging.DEBUG,for
mat='% (asctime)s %(message)s')

print "Welcome to the UPC forensics app. This is the Massive OCR
tool.\n"

ocr_script=ocr()
ocr_script.run()

```

Appendix 15. OCR GUI glade code

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
    <requires lib="gtk+" version="3.12"/>

```

```

<object class="GtkWindow" id="Ocr_window">
  <property name="can_focus">False</property>
  <child>
    <object class="GtkBox">
      <property name="visible">True</property>
      <property name="can_focus">False</property>
      <property name="margin_left">1</property>
      <property name="margin_right">1</property>
      <property name="margin_top">1</property>
      <property name="margin_bottom">1</property>
      <property name="orientation">vertical</property>
      <child>
        <object class="GtkLabel">
          <property name="visible">True</property>
          <property name="can_focus">False</property>
          <property name="label" translatable="yes">OCR
TOOL</property>
          <attributes>
            <attribute name="font-desc" value="Abyssinica SIL Bold
10"/>
            <attribute name="weight" value="ultraheavy"/>
            <attribute name="scale" value="3"/>
          </attributes>
        </object>
        <packing>
          <property name="expand">False</property>
          <property name="fill">True</property>
          <property name="position">0</property>
        </packing>
      </child>
    </child>
    <child>
      <object class="GtkGrid">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="margin_left">1</property>
        <property name="margin_right">1</property>
        <child>

```



```

        <object class="GtkLabel">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="label"
translatable="yes">Folder:</property>
        </object>
        <packing>
            <property name="left_attach">0</property>
            <property name="top_attach">0</property>
        </packing>
    </child>
    <child>
        <object class="GtkEntry" id="entry_folder">
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="hexpand">True</property>
        </object>
        <packing>
            <property name="left_attach">1</property>
            <property name="top_attach">0</property>
        </packing>
    </child>
    <child>
        <object class="GtkButton" id="but_choose">
            <property name="label"
translatable="yes">Choose</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
        </object>
        <packing>
            <property name="left_attach">2</property>
            <property name="top_attach">0</property>
        </packing>
    </child>
</object>
<packing>

```

```

        <property name="expand">False</property>
        <property name="fill">True</property>
        <property name="padding">1</property>
        <property name="position">1</property>
    </packing>
</child>
<child>
    <object class="GtkGrid">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="margin_left">2</property>
        <property name="margin_right">2</property>
        <property name="margin_top">1</property>
        <property name="margin_bottom">1</property>
        <child>
            <object class="GtkButton" id="but_prepare">
                <property name="label"
translatable="yes">PREPARE</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <property name="hexpand">True</property>
            </object>
            <packing>
                <property name="left_attach">0</property>
                <property name="top_attach">0</property>
            </packing>
        </child>
        <child>
            <object class="GtkButton" id="but_divide">
                <property name="label"
translatable="yes">DIVIDE</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
            </object>
            <packing>

```

```

        <property name="left_attach">0</property>
        <property name="top_attach">1</property>
    </packing>
</child>
<child>
    <object class="GtkButton" id="but_ocr">
        <property name="label"
translatable="yes">OCR</property>
        <property name="visible">True</property>
        <property name="can_focus">True</property>
        <property name="receives_default">True</property>
    </object>
    <packing>
        <property name="left_attach">0</property>
        <property name="top_attach">2</property>
    </packing>
</child>
<child>
    <object class="GtkLabel" id="text_prepare">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label" translatable="yes">select folder
first</property>
    </object>
    <packing>
        <property name="left_attach">1</property>
        <property name="top_attach">0</property>
    </packing>
</child>
<child>
    <object class="GtkLabel" id="text_divide">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label" translatable="yes">do prepare
first</property>
    </object>
    <packing>

```

```

        <property name="left_attach">1</property>
        <property name="top_attach">1</property>
    </packing>
</child>
<child>
    <object class="GtkLabel" id="text_ocr">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label" translatable="yes">do divide
first</property>
    </object>
    <packing>
        <property name="left_attach">1</property>
        <property name="top_attach">2</property>
    </packing>
</child>
</object>
<packing>
    <property name="expand">False</property>
    <property name="fill">True</property>
    <property name="padding">2</property>
    <property name="position">2</property>
</packing>
</child>
</object>
</child>
</object>
</interface>

```

Appendix 16. OCR desktop code

```

[Desktop Entry]
Name=OCR
Exec=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/OCR/
OCR_gui.py
Path=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/OCR/
Terminal=false

```

```
Type=Application
```

Appendix 17. eDiscovery starting window python code

```
#!/usr/bin/env python
import logging, time, sys, os, stat, time, hashlib, argparse, csv, gi
gi.require_version("Gtk", "3.0")
from gi.repository import Gtk
from gi.repository import Gdk

class search():
    def __init__(self):
        self.builder=Gtk.Builder()
        self.builder.add_from_file("Search.glade")
        self.win=self.builder.get_object("search_window")
        self.win.connect("delete-event",Gtk.main_quit)
        self.win.set_title("SEARCH tool")
        self.win.set_resizable(True)
        self.win.set_default_size(400,50)
        self.win.move(400,200)
        grepButton = self.builder.get_object("but_grep")
        grepButton.connect("clicked", self.run_grep)
        gawkButton = self.builder.get_object("but_gawk")
        gawkButton.connect("clicked",self.run_gawk)

    def run(self):
        self.win.show_all()
        Gtk.main()

    def run_grep(self, widget):
        dire=os.path.realpath(__file__)
        dire=dire[:-9]
        cmd="python "+dire+"Grep_eDiscovery.py"
        os.system(cmd)

    def run_gawk(self, widget):
```

```

        dire=os.path.realpath(__file__)
        dire=dire[:-9]
        cmd="python "+dire+"Gawk_eDiscovery.py"
        os.system(cmd)

print "Welcome to the UPC forensics app. This is the Search tool.\n"
search_script=search()
search_script.run()

```

Appendix 18. eDiscovery grep python code

```

#!/usr/bin/env python
import subprocess, os, tempfile, ntpath, time, logging, gi, csv
gi.require_version("Gtk", "3.0")
from gi.repository import Gtk

class search_words():
    directory=""
    word_f=""
    num_results=0
    results=Gtk.TextBuffer()
    columns=["File", "Directory"]
    def __init__(self):
        self.builder=Gtk.Builder()
        self.builder.add_from_file("eDiscovery.glade")
        self.win=self.builder.get_object("main_window")
        self.win.connect("delete-event", Gtk.main_quit)
        self.win.set_title("Addvante eDiscovery")
        self.win.set_resizable(True)
        self.win.set_default_size(800,500)
        self.win.move(400,200)

        #make the filter button work by default when you press enter
        filtButton = self.builder.get_object("filter_but")
        filtButton.connect("clicked", self.do_filter)

```

```

word_text = self.builder.get_object("filter_text")
word_text.connect("activate", self.do_filter)
word_text.grab_focus_without_selecting()

button_dir = self.builder.get_object("choosedir_but")
button_dir.connect("clicked", self.get_dir)

button_filt = self.builder.get_object("pdf_but")
button_filt.connect("clicked", self.open_pdf)

button_filt = self.builder.get_object("txt_but")
button_filt.connect("clicked", self.open_txt)

button_csv = self.builder.get_object("csv_but")
button_csv.connect("clicked", self.save_csv)

def run(self):
    self.win.show_all()
    Gtk.main()

def get_dir(self, widget):
    self.open_dialog=Gtk.FileChooserDialog("Select folder", None, Gtk.FileChooserAction.SELECT_FOLDER, (Gtk.STOCK_CANCEL,
Gtk.ResponseType.CANCEL, "Select", Gtk.ResponseType.OK))
    self.open_dialog.set_default_size(800,400)
    self.open_dialog.connect("response", self.open_dir_dialog)
    self.open_dialog.show()
    self.win.hide()

def do_filter(self, widget):
    results_list=self.builder.get_object("results_list")
    info=self.builder.get_object("results_label")
    info.set_text("Searching... Please wait.")
    print "SET: Searching..."
    results_list.clear()
    word_text=self.builder.get_object("filter_text")
    dire=self.builder.get_object("choosedir_text")

```

```

        self.word_f=word_text.get_text()
        self.directory=dire.get_text()

        if self.word_f=="":

error_dialog=Gtk.MessageDialog (None,0,Gtk.MessageType.ERROR,Gtk.Buttons
Type.CANCEL,"ERROR")

            error_dialog.format_secondary_text("MISSING WORD. Insert a
word please.")

            error_dialog.run()

            error_dialog.destroy()

        elif self.directory=="":

error_dialog=Gtk.MessageDialog (None,0,Gtk.MessageType.ERROR,Gtk.Buttons
Type.CANCEL,"ERROR")

            error_dialog.format_secondary_text("MISSING          DIRECTORY.
Insert a directory please.")

            error_dialog.run()

            error_dialog.destroy()

        else:

            startTime = time.time()

            scmd="grep -r "+self.directory+" -e "+self.word_f+" -i"

proc=subprocess.Popen (scmd,shell=True,stdout=subprocess.PIPE)

            x=0

            for line in proc.stdout.readlines():

                x=x+1

                self.prepare(line)

            if x==0:

error_dialog=Gtk.MessageDialog (None,0,Gtk.MessageType.ERROR,Gtk.Buttons
Type.CANCEL,"ERROR")

                error_dialog.format_secondary_text("Not found any file
matching this word in this folder.")

                error_dialog.run()

                error_dialog.destroy()

                info.set_text("0 results:")

            else:

                self.num_results=x

                info.set_text(str(self.num_results)+" results:")

            endTime = time.time()

```



```

        duration = endTime - startTime
        logging.info('Grep Duration: ' + str(duration) + '
seconds')

def open_dir_dialog(self, dialog, response_id):
    open_dialog=dialog
    if response_id==Gtk.ResponseType.OK:
        self.directory=open_dialog.get_uri().replace("file://", "")
        self.open_dialog.hide()
        self.win.show()
        self.win.move(400,200)
        dir_text = self.builder.get_object("choosedir_text")
        dir_text.set_text(self.directory)
    elif response_id==Gtk.ResponseType.CANCEL:
        self.open_dialog.hide()
        self.win.show()
        self.win.move(400,200)

def prepare(self, line):
    line=line.rstrip()
    results_list=self.builder.get_object("results_list")
    info=self.builder.get_object("results_label")
    Bname=line.index("_b_")
    Lname=(line.index(".txt"))+4
    txtname=line[Bname:Lname]
    line=txtname.replace("_b_", "/")
    #line=line.replace("_s_", " ")
    N=line.index(".pdf")
    line=line[:N]
    [dir_x, file_x]=ntpath.split(line)
    nTemp = txtname.split('.pdf-')
    page_x = nTemp[1]
    page_x = page_x[:-4]
    results_list.append((file_x, dir_x, page_x, txtname))

def open_pdf(self, widget):
    results_tree=self.builder.get_object("results_tree")

```

```

        (model,iter)=results_tree.get_selection().get_selected()

        if iter is None:

error_dialog=Gtk.MessageDialog (None,0,Gtk.MessageType.ERROR,Gtk.Buttons
Type.CANCEL,"ERROR")

        error_dialog.format_secondary_text("NOT SELECTED. Select a
file to open.")

        error_dialog.run()

        error_dialog.destroy()

    else:

        print "0: "+model[iter][0]
        print "1: "+model[iter][1]
        print "2: "+model[iter][2]
        print "3: "+model[iter][3]

        auxIt=model[iter][3]

        nTemp = auxIt.split('.pdf-')
        nMod = nTemp[1]
        nMod = nMod[:-4]

        #nIt=auxIt.index(".pdf-")+4
        #nFi=auxIt.index(".txt")-1

        #page_x=auxIt[nIt:nFi]

        pdfcmd="evince          "+(model[iter][1]).replace(" ", "\\
")+ "/" +(model[iter][0]).replace(" ", "\\ ") + ".pdf --page-label="+nMod

        print pdfcmd

        os.system(pdfcmd)

    def open_txt(self,widget):

        results_tree=self.builder.get_object("results_tree")

        (model,iter)=results_tree.get_selection().get_selected()

        if iter is None:

error_dialog=Gtk.MessageDialog (None,0,Gtk.MessageType.ERROR,Gtk.Buttons
Type.CANCEL,"ERROR")

        error_dialog.format_secondary_text("NOT SELECTED. Select a
file to open.")

        error_dialog.run()

        error_dialog.destroy()

    else:

        print model[iter][0]

```

```

        print model[iter][1]
        print model[iter][2]
        print model[iter][3]
        direct=self.directory+"/"+model[iter][3]
        os.system("xdg-open "+direct)

    def save_csv(self,widget):
        print "Saving"
        results_tree=self.builder.get_object("results_list")
        aux=os.path.realpath(__file__)
        aux=aux[:-18]
        print aux
        with open(aux+"results_"+self.word_f+".csv","wb") as out:
            csv_out=csv.writer(out)
            csv_out.writerow(["File","Directory","Num. Page"])
            for row in results_tree:
                # Print values of all columns
                csv_out.writerow(row[:-1])

logging.basicConfig(filename='time_grep.log',level=logging.DEBUG,format
='% (asctime)s %(message)s')
search_script=search_words()
search_script.run()

```

Appendix 19. eDiscovery gawk python code

```

#!/usr/bin/env python

import subprocess, logging, time, os, tempfile, ntpath, gi, glob,
string, sys, csv

gi.require_version("Gtk","3.0")

from gi.repository import Gtk
from gi.repository import Gdk

class search_words():
    directory=""
    word_f=""
    num_results=0

```

```

results=Gtk.TextBuffer()
columns=["File","Directory"]
def __init__(self):
    self.builder=Gtk.Builder()
    self.builder.add_from_file("eDiscovery.glade")
    self.win=self.builder.get_object("main_window")
    self.win.connect("delete-event",Gtk.main_quit)
    self.win.set_title("Addvante eDiscovery")
    self.win.set_resizable(True)
    self.win.set_default_size(800,500)
    self.win.move(400,200)

    #make the filter button work by default when you press enter
    filtButton = self.builder.get_object("filter_but")
    filtButton.connect("clicked", self.do_filter)

    word_text = self.builder.get_object("filter_text")
    word_text.connect("activate", self.do_filter)
    word_text.grab_focus_without_selecting()

    button_dir = self.builder.get_object("choosedir_but")
    button_dir.connect("clicked",self.get_dir)

    button_filt = self.builder.get_object("pdf_but")
    button_filt.connect("clicked",self.open_pdf)

    button_filt = self.builder.get_object("txt_but")
    button_filt.connect("clicked",self.open_txt)

    button_csv = self.builder.get_object("csv_but")
    button_csv.connect("clicked",self.save_csv)

def run(self):
    self.win.show_all()
    Gtk.main()

```

```

def get_dir(self, widget):
    self.open_dialog=Gtk.FileChooserDialog("Select a
folder",None,Gtk.FileChooserAction.SELECT_FOLDER,(Gtk.STOCK_CANCEL,
Gtk.ResponseType.CANCEL,"Select",Gtk.ResponseType.OK))
    self.open_dialog.set_default_size(800,400)
    self.open_dialog.set_current_folder("/home/addvante/Desktop/")
    self.open_dialog.connect("response",self.open_dir_dialog)
    self.open_dialog.show()
    self.win.hide()

def do_filter(self, widget):
    results_list=self.builder.get_object("results_list")
    info=self.builder.get_object("results_label")
    info.set_text("Searching... Please wait.")
    print "SET: Searching..."
    results_list.clear()
    word_text=self.builder.get_object("filter_text")
    dire=self.builder.get_object("choosedir_text")
    self.word_f=word_text.get_text()
    self.directory=dire.get_text()
    if self.word_f=="":
        error_dialog=Gtk.MessageDialog(None,0,Gtk.MessageType.ERROR,Gtk.Buttons
Type.CANCEL,"ERROR")
        error_dialog.format_secondary_text("MISSING WORD. Insert a
word please.")
        error_dialog.run()
        error_dialog.destroy()
    elif self.directory=="":
        error_dialog=Gtk.MessageDialog(None,0,Gtk.MessageType.ERROR,Gtk.Buttons
Type.CANCEL,"ERROR")
        error_dialog.format_secondary_text("MISSING DIRECTORY.
Insert a directory please.")
        error_dialog.run()
        error_dialog.destroy()
    else:
        startTime = time.time()
        os.chdir(self.directory)

```

```

        entrada = self.word_f
        x=0
        for file in glob.glob("*.txt"):
            match = 0
            proc = ""
            cmd="gawk '" + entrada + "' IGNORECASE=1 "+file+"\"

proc=subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE)

            for line in proc.stdout.readlines():
                if line != "":
                    x=x+1
                    if (match == 0):
                        match = 1
                        self.prepare(file)

        print "OK ALL"
        print x
        if x==0:

error_dialog=Gtk.MessageDialog (None,0,Gtk.MessageType.ERROR,Gtk.Buttons
Type.CANCEL,"ERROR")

            error_dialog.format_secondary_text("Not found any file
matching this word in this folder.")
            error_dialog.run()
            error_dialog.destroy()
            info.set_text("0 results:")
        else:
            self.num_results=x
            info.set_text(str(self.num_results)+" results:")
        endTime = time.time()
        duration = endTime - startTime
        logging.info('Gawk Duration: ' + str(duration) + '
seconds')

    def open_dir_dialog(self,dialog,response_id):
        open_dialog=dialog
        if response_id==Gtk.ResponseType.OK:
            self.directory=open_dialog.get_uri().replace("file://","")
            self.open_dialog.hide()

```

```

        self.win.show()
        self.win.move(400,200)
        dir_text = self.builder.get_object("choosedir_text")
        dir_text.set_text(self.directory)
    elif response_id==Gtk.ResponseType.CANCEL:
        self.open_dialog.hide()
        self.win.show()
        self.win.move(400,200)

def prepare(self, line):
    line=line.rstrip()
    results_list=self.builder.get_object("results_list")
    info=self.builder.get_object("results_label")
    Bname=line.index("_b_")
    Lname=(line.index(".txt")+4)
    txtname=line[Bname:Lname]
    line=txtname.replace("_b_","/")
    #line=line.replace("_s_", " ")
    N=line.index(".pdf")
    line=line[:N]
    [dir_x,file_x]=ntpath.split(line)
    nTemp = txtname.split('.pdf-')
    page_x = nTemp[1]
    page_x = page_x[:-4]
    results_list.append((file_x,dir_x, page_x, txtname))

def open_pdf(self,widget):
    results_tree=self.builder.get_object("results_tree")
    (model,iter)=results_tree.get_selection().get_selected()
    if iter is None:
        error_dialog=Gtk.MessageDialog(None,0,Gtk.MessageType.ERROR,Gtk.Buttons
        Type.CANCEL,"ERROR")
        error_dialog.format_secondary_text("NOT SELECTED. Select a
        file to open.")
        error_dialog.run()

```

```

        error_dialog.destroy()
    else:
        print model[iter][0]
        print model[iter][1]
        print model[iter][2]
        print model[iter][3]
        auxIt=model[iter][3]
        nTemp = auxIt.split('.pdf-')
        nMod = nTemp[1]
        nMod = nMod[:-4]
        #nIt=auxIt.index(".pdf-")+4
        #nFi=auxIt.index(".txt")-1
        #page_x=auxIt[nIt:nFi]
        pdfcmd="evince "+(model[iter][1]).replace(" ", "\\ ")+" /"+(model[iter][0]).replace(" ", "\\ ")+" .pdf --page-label="+nMod
        print pdfcmd
        os.system(pdfcmd)

    def open_txt(self,widget):
        results_tree=self.builder.get_object("results_tree")
        (model,iter)=results_tree.get_selection().get_selected()
        if iter is None:
            error_dialog=Gtk.MessageDialog(None,0,Gtk.MessageType.ERROR,Gtk.Buttons
            Type.CANCEL,"ERROR")
            error_dialog.format_secondary_text("NOT SELECTED. Select a
            file to open.")
            error_dialog.run()
            error_dialog.destroy()
        else:
            print model[iter][0]
            print model[iter][1]
            print model[iter][2]
            print model[iter][3]
            direct=self.directory+"/"+model[iter][3]
            os.system("xdg-open "+direct)

    def save_csv(self,widget):

```



```

    print "Saving"
    results_tree=self.builder.get_object("results_list")
    aux=os.path.realpath(__file__)
    aux=aux[:-18]
    print aux
    with open(aux+"results_"+self.word_f+".csv", "wb") as out:
        csv_out=csv.writer(out)
        csv_out.writerow(["File", "Directory", "Num. Page"])
        for row in results_tree:
            # Print values of all columns
            csv_out.writerow(row[:-1])

logging.basicConfig(filename='time_gawk.log', level=logging.DEBUG, format
='% (asctime)s %(message)s')
search_script=search_words()
search_script.run()

```

Appendix 20. eDiscovery starting window glade code

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.12"/>
  <object class="GtkWindow" id="search_window">
    <property name="can_focus">False</property>
    <child>
      <object class="GtkBox" id="box1">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <child>
          <object class="GtkLabel" id="T">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="label" translatable="yes">SEARCH</property>
            <attributes>
              <attribute name="font-desc" value="Abyssinica SIL Bold

```

```

10"/>

    <attribute name="weight" value="ultraheavy"/>
    <attribute name="scale" value="3"/>
  </attributes>
</object>
<packing>
  <property name="expand">True</property>
  <property name="fill">True</property>
  <property name="position">0</property>
</packing>
</child>
<child>
  <object class="GtkGrid" id="grid1">
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <property name="row_spacing">1</property>
    <property name="column_spacing">4</property>
    <child>
      <object class="GtkButton" id="but_grep">
        <property name="label"
translatable="yes">Choose</property>
        <property name="visible">True</property>
        <property name="can_focus">True</property>
        <property name="receives_default">True</property>
        <property name="hexpand">True</property>
      </object>
      <packing>
        <property name="left_attach">1</property>
        <property name="top_attach">0</property>
      </packing>
    </child>
    <child>
      <object class="GtkButton" id="but_gawk">
        <property name="label"
translatable="yes">Choose</property>
        <property name="visible">True</property>
        <property name="can_focus">True</property>

```

```

        <property name="receives_default">True</property>
        <property name="hexexpand">True</property>
    </object>
    <packing>
        <property name="left_attach">1</property>
        <property name="top_attach">1</property>
    </packing>
</child>
<child>
    <object class="GtkLabel" id="label1">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label" translatable="yes">GREP
search</property>
    </object>
    <packing>
        <property name="left_attach">0</property>
        <property name="top_attach">0</property>
    </packing>
</child>
<child>
    <object class="GtkLabel" id="label2">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label" translatable="yes">GAWK
search</property>
    </object>
    <packing>
        <property name="left_attach">0</property>
        <property name="top_attach">1</property>
    </packing>
</child>
</object>
<packing>
    <property name="expand">False</property>
    <property name="fill">True</property>
    <property name="position">1</property>

```

```

        </packing>
    </child>
</object>
</child>
</object>
</interface>

```

Appendix 21. eDiscovery operating window glade code

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
    <requires lib="gtk+" version="3.20"/>
    <object class="GtkListStore" id="results_list">
        <columns>
            <!-- column-name File -->
            <column type="gchararray"/>
            <!-- column-name Directory -->
            <column type="gchararray"/>
            <!-- column-name num_page -->
            <column type="gchararray"/>
            <!-- column-name txt_name -->
            <column type="gchararray"/>
        </columns>
    </object>
    <object class="GtkWindow" id="main_window">
        <property name="can_focus">False</property>
        <property name="icon">icone_AddVANTE.jpg</property>
        <child>
            <object class="GtkGrid">
                <property name="visible">True</property>
                <property name="can_focus">False</property>
                <child>
                    <object class="GtkLabel">
                        <property name="visible">True</property>
                        <property name="can_focus">False</property>

```

```

        <property name="hexexpand">True</property>
        <property name="label" translatable="yes">Select directory
with OCR results:</property>
        <property name="xalign">0.019999999552965164</property>
    </object>
    <packing>
        <property name="left_attach">0</property>
        <property name="top_attach">1</property>
    </packing>
</child>
<child>
    <object class="GtkGrid">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="column_spacing">5</property>
        <child>
            <object class="GtkButton" id="choosedir_but">
                <property name="label" translatable="yes">Choose
Directory</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <property name="margin_left">5</property>
                <property name="margin_right">6</property>
            </object>
            <packing>
                <property name="left_attach">1</property>
                <property name="top_attach">0</property>
            </packing>
        </child>
        <child>
            <object class="GtkEntry" id="choosedir_text">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="margin_left">5</property>
                <property name="margin_right">5</property>
                <property name="hexexpand">True</property>

```

```

        <property name="text"
translatable="yes">/home/addvante/Desktop/OCRMassiu</property>
    </object>
    <packing>
        <property name="left_attach">0</property>
        <property name="top_attach">0</property>
    </packing>
</child>
</object>
<packing>
    <property name="left_attach">0</property>
    <property name="top_attach">2</property>
</packing>
</child>
<child>
    <object class="GtkLabel">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label" translatable="yes">Insert word to
search (regular exp):</property>
        <property name="xalign">0.019999999552965164</property>
    </object>
    <packing>
        <property name="left_attach">0</property>
        <property name="top_attach">3</property>
    </packing>
</child>
<child>
    <object class="GtkLabel" id="results_label">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label"
translatable="yes">Results:</property>
        <property name="xalign">0.019999999552965164</property>
    </object>
    <packing>
        <property name="left_attach">0</property>

```

Filter

```
        <property name="top_attach">5</property>
    </packing>
</child>
<child>
    <object class="GtkGrid">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="column_spacing">5</property>
        <child>
            <object class="GtkButton" id="filter_but">
                <property name="label" translatable="yes">
                    </property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <property name="margin_left">5</property>
                <property name="margin_right">5</property>
            </object>
            <packing>
                <property name="left_attach">1</property>
                <property name="top_attach">0</property>
            </packing>
        </child>
        <child>
            <object class="GtkEntry" id="filter_text">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="margin_left">5</property>
                <property name="margin_right">5</property>
                <property name="expand">True</property>
            </object>
            <packing>
                <property name="left_attach">0</property>
                <property name="top_attach">0</property>
            </packing>
        </child>
    </object>
</child>
```

```

    <packing>
      <property name="left_attach">0</property>
      <property name="top_attach">4</property>
    </packing>
  </child>
  <child>
    <object class="GtkGrid">
      <property name="visible">True</property>
      <property name="can_focus">False</property>
      <property name="column_spacing">2</property>
      <property name="column_homogeneous">True</property>
      <child>
        <object class="GtkButton" id="pdf_but">
          <property name="label" translatable="yes">OPEN
PDF</property>
          <property name="visible">True</property>
          <property name="can_focus">True</property>
          <property name="receives_default">True</property>
          <property name="margin_left">5</property>
          <property name="margin_right">5</property>
        </object>
        <packing>
          <property name="left_attach">0</property>
          <property name="top_attach">0</property>
        </packing>
      </child>
      <child>
        <object class="GtkButton" id="txt_but">
          <property name="label" translatable="yes">OPEN
TXT</property>
          <property name="visible">True</property>
          <property name="can_focus">True</property>
          <property name="receives_default">True</property>
          <property name="margin_left">5</property>
          <property name="margin_right">5</property>
        </object>
        <packing>

```



```

        <property name="left_attach">1</property>
        <property name="top_attach">0</property>
    </packing>
</child>
<child>
    <object class="GtkButton" id="csv_but">
        <property name="label" translatable="yes">SAVE
CSV</property>
        <property name="visible">True</property>
        <property name="can_focus">True</property>
        <property name="receives_default">True</property>
        <property name="margin_left">5</property>
        <property name="margin_right">5</property>
    </object>
    <packing>
        <property name="left_attach">2</property>
        <property name="top_attach">0</property>
    </packing>
</child>
</object>
<packing>
    <property name="left_attach">0</property>
    <property name="top_attach">6</property>
</packing>
</child>
<child>
    <object class="GtkScrolledWindow" id="results_view">
        <property name="visible">True</property>
        <property name="can_focus">True</property>
        <property name="margin_left">5</property>
        <property name="margin_right">5</property>
        <property name="margin_top">3</property>
        <property name="vexpand">True</property>
        <property name="shadow_type">in</property>
        <child>
            <object class="GtkTreeView" id="results_tree">
                <property name="visible">True</property>

```

```

    <property name="can_focus">True</property>
    <property name="model">results_list</property>
    <child internal-child="selection">
      <object class="GtkTreeSelection"/>
    </child>
    <child>
      <object class="GtkTreeViewColumn" id="file_column">
        <property name="title">File</property>
        <child>
          <object class="GtkCellRendererText"
id="cellrenderertext0"/>
          <attributes>
            <attribute name="text">0</attribute>
          </attributes>
        </child>
      </object>
    </child>
    <child>
      <object class="GtkTreeViewColumn"
id="directory_column">
        <property name="title">Directory</property>
        <child>
          <object class="GtkCellRendererText"
id="cellrenderertext1"/>
          <attributes>
            <attribute name="text">1</attribute>
          </attributes>
        </child>
      </object>
    </child>
    <child>
      <object class="GtkTreeViewColumn" id="page_num">
        <property name="title">Num. Page</property>
        <child>
          <object class="GtkCellRendererText"
id="cellrenderertext2"/>
          <attributes>
            <attribute name="text">2</attribute>

```

```

        </attributes>
    </child>
</object>
</child>
</object>
</child>
</object>
<packing>
    <property name="left_attach">0</property>
    <property name="top_attach">7</property>
</packing>
</child>
<child>
    <object class="GtkImage">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="margin_top">6</property>
        <property name="margin_bottom">6</property>
        <property name="pixbuf">mini.png</property>
    </object>
    <packing>
        <property name="left_attach">0</property>
        <property name="top_attach">0</property>
    </packing>
</child>
<child>
    <object class="GtkLabel">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="halign">end</property>
        <property name="margin_left">2</property>
        <property name="margin_right">2</property>
        <property name="margin_top">2</property>
        <property name="margin_bottom">2</property>
        <property name="label" translatable="yes">Developed by ISG-
UPC</property>
        <attributes>

```

```

        <attribute    name="font-desc"    value="&lt;Enter  Value&gt;
8"/>

        </attributes>
    </object>

    <packing>
        <property name="left_attach">0</property>
        <property name="top_attach">8</property>
    </packing>
</child>
</object>
</child>
</object>
</interface>

```

Appendix 22. eDiscovery desktop code

```

[Desktop Entry]
Name=eDiscovery
Exec=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/EDIS
COVERY/Search.py
Path=/home/jordi/Escriptori/TFM/Forensics_Scripts/Scripts_with_GUI/EDIS
COVERY/
Terminal=false
Type=Application

```

Glossary

A list of all acronyms and what they stand for.

GUI	Graphical User Interface
OCR	Optical Character Recognition
OS	Operative System