Revealer Toolkit – User Guide

Jose Navarro

September 16, 2009

Contents

Chapter 1

License

1.1 License of this document

1.2 License of the Revealer Toolkit

Chapter 2

Acknowledgements

- INCIDE (Investigacion Digital S.L., www.incide.es) where developers and testers work
- Manu Ginés aka xkulio creator of the original Chanchullos Revealer
- Generalitat de Catalunya for partial funding of the project
- People that have collaborated to (and suffered) the project: Jose Navarro, Luis Gómez, Sara Rincón, Abraham Pasamar, Julián Sotos, Helena Fuentes, ...

Chapter 3

Introduction

the Revealer Toolkit is a framework and simple scripts for computer forensics. It uses Brian Carrier's The Sleuth Kit as the backbone, as well as other free tools.

The aim of the Revealer Toolkit is to automate rutinary tasks and to manage sources and results from another perspective than the usual forensic frameworks. It will be specially useful in cases with several computers and digitals forensic sources.

RVT is developed and actively tested by computer forensic investigators working at INCIDE, spanish company sited at the beautiful city of Barcelona (see www.incide.es for more details)

You can find additional information, packages and all the source code at http://code.google.com/p/revealertoolkit

Introduction to version 0.2

The current state of the project can be described as a '*proof of concept*', that is, RVT version 0.2 proofs that the objectives we are looking for are reachable, but further work is necessary in order to have a stable version.

Therefore, RVT v0.2 can be used to automate computer forensic tasks on a group of several digital forensic images, but the code is still buggy, some tasks have to be run manually, and the interface has to be improved. A complete working RVT system has been installed and is free (like in beer *and* in freedom) for download at http://revealertoolkit.googlecode.com/files/RVT_v0.2.zip .

The objective of next version (0.3) will be to clean the code, solve bugs and ease the interaction. At the same time, more scripts and modules will be developed. For version 0.3, a *RVT Developer Manual* is planned, as well as a automated reporting engine.

For any questions, help or comments, please, do not hesitate to drop an message in our newsletter (http://groups.google.com/group/revealertoolkit).

Chapter 4

Installation

This is and example installation of the Revealer Toolkit Shell on a Debian *lenny* 5.00. Also covers the creation of a valid folder structure and a sample image.

4.1 Operating system

RVT is designed to work on Debian linux systems, and some errors have been reported when used on other linux flavours, so be patient.

Download lastest Debian stable version. In this example, Debian 5.00 *lenny* is used.

Some additional packages are required to be installed with *apt-get*:

```
apt-get install openssh-server unzip sudo vim dosfstools gcc g++ make
apt-get install libxml-simple-perl libdate-manip-perl

adduser analyst
addgroup forensics
groupadd -g 1010 forensics
adduser analyst forensics
adduser analyst disk

mkdir /media/morgue
chgrp forensics /media/morgue
chmod g+sw /media/morgue

echo -e "\n\numask 002" >> /etc/profile
```

Download and install lastest The SleuthKit version. Code can be downloaded from *http:// www.sleuthkit.org/ sleuthkit/ download.php*. For version 3.0.1, as root:

```
cd
http://puzzle.dl.sourceforge.net/sourceforge/sleuthkit/sleuthkit-3.0.1.tar.gz
tar -zxf sleuthkit-3.0.1.tar.gz
cd sleuthkit-3.0.1
./configure
make
make install
```

Also, adding additional loop devices is recommended. Modify your kernel boot to include the option *max_loop=32* to increase the number of loop devices to 32. Additionaly, these commands should be executed after every boot (so can be included in debian's /etc /init.d /bootmisc.sh).

```
for i in $(seq 8 31) ; do mknod /dev/loop$i b 7 $i ; done
chmod 660 /dev/loop* ; chown 0.disk /dev/loop*
```

as root, add these lines to the sudoers file (with *visudo* command):

```
Defaults:%forensics !authenticate

%forensics ALL=(root) /bin/mount, (root) /bin/umount, (root) /sbin/losetup
```

4.2 Software

Download and install last stable version of RVT. As root:

```
mkdir /usr/local/src/revealer
cd /usr/local/src/revealer

wget revealertoolkit.googlecode.com/files/RVT_v0.2.zip
unzip RVT_v0.2.zip
chmod a+x RVT/RVT.pl
```

while a proper perl package is prepared, this trick should be done to take RVT inside the path (if you know something better, please, let me know). Create */usr/local/bin/rvt* with the following content:

```
#!/bin/bash
cd /usr/local/src/revealer/RVT
perl RVT.pl $*
cd - > /dev/null
```

and give it proper permissions:

```
chgrp forensics /usr/local/bin/rvt
chmod g+x /usr/local/bin/rvt
```

4.3 Folder structure

as user analyst:

```
mkdir /media/morgue/images
```

RVT is not able yet to create new image and case folders, so you must do it manually (sorry). For example, for case 100101, codename *ghost*:

```
mkdir /media/morgue/images/100101-ghost
mkdir -p /media/morgue/100101-ghost/100101-01-1/{mnt/output}
```

4.4 Sample image

If you are installing RVT for evaluation purposes, maybe you don't have a hard disk image at hand. You can use a example image, like those on dftt.sf.net:

```
cd /media/morgue/images/100101-ghost
wget http://freefr.dl.sourceforge.net/sourceforge/dftt/1-extend-part.zip
unzip 1-extend-part.zip
mv 1-extend-part/ext-part-test-2.dd 100101-01-1.dd
```

You can also create a image from scratch. See appendix A for an example.

4.5 First test

Just for testing, some first commands can be executed. As analyst, execute the RVT Shell:

```
$ rvt
$
```

After the preliminar scanning, RVT Shell will offer you a prompt:

```
RVT >
```

Now, execute these commands:

```
images scanall
set level 100101-01-1
script strings generate
script timelines generate
script search quickcount emails
script search quickcount accounts
quit
```

You can check that the directory */media/morgue/100101-ghost/100101-01-1/output* has been populated with results.

The same can be achieved creating a file with the commands and piping it to the RVT Shell using the *-b* argument:

```
cat preforensics.rvt | RVT -b
```

4.6 Additional software

RVT is a forensic framework that uses software from the Revealer Toolkit project, but also from other sources. Additional software can be installed to activate some modules.

- *RVT tools*: some little but nice and GNU/GPL tools that are shipped with RVT source code, under the folder, guess, *tools*. In order to some RVT functions to be operative, make sure that all the tools are on the system PATH.
- *libpst*: libpst (www.five-ten-sg.com/libpst) parses Microsoft Outlook mailboxes, and is handled by RVT_mail module. See section B.2 for more information on libpst installation
- *RegRipper*: RegRipper (www.regripper.net) offers modular parsing and data mining on Microsoft Windows registry hives, and it is handled by RVT_regripper module. See section B.3 for more information on RegRipper installation

Chapter 5

Folder structure

The morgue stores disk images and results of forensic analysis. Each morgue must have a strict folder structure.

For now, this folder structure (image and case folders) must to be maintained manually, so don't forget to create image and case folders (including disks, mnt and output folders) every time you import a new forensic image in RVT.

On page 5.1, the RVT folder structure is graphicaly represented.

5.1 Morgues

The Revealer Toolkit can handle more that one morgue. By default, one morgue is defined in RVT, located at */media/morgue*.

5.2 Cases

Each forensic case is determined by a *case number* and a *case codename*, separated by a dash. For example, the example case created in the Chapter 4 is noted as *100101-ghost*.

Each case has a folder assigned in the morgue, under the folder *images*, where the disk images are stored.

Also, each case has a folder in the morgue, where all the forensic results are stored.

The folder structure, at the case level, for the example installation shown in chapter 4, will be:

```
/media/morgue/100101-ghost

/media/morgue/imagenes/100101-ghost
/media/morgue/imagenes/100101-ghost/100101-01-1.dd
```

where *100101-01-1.dd* is the dd image of a disk.

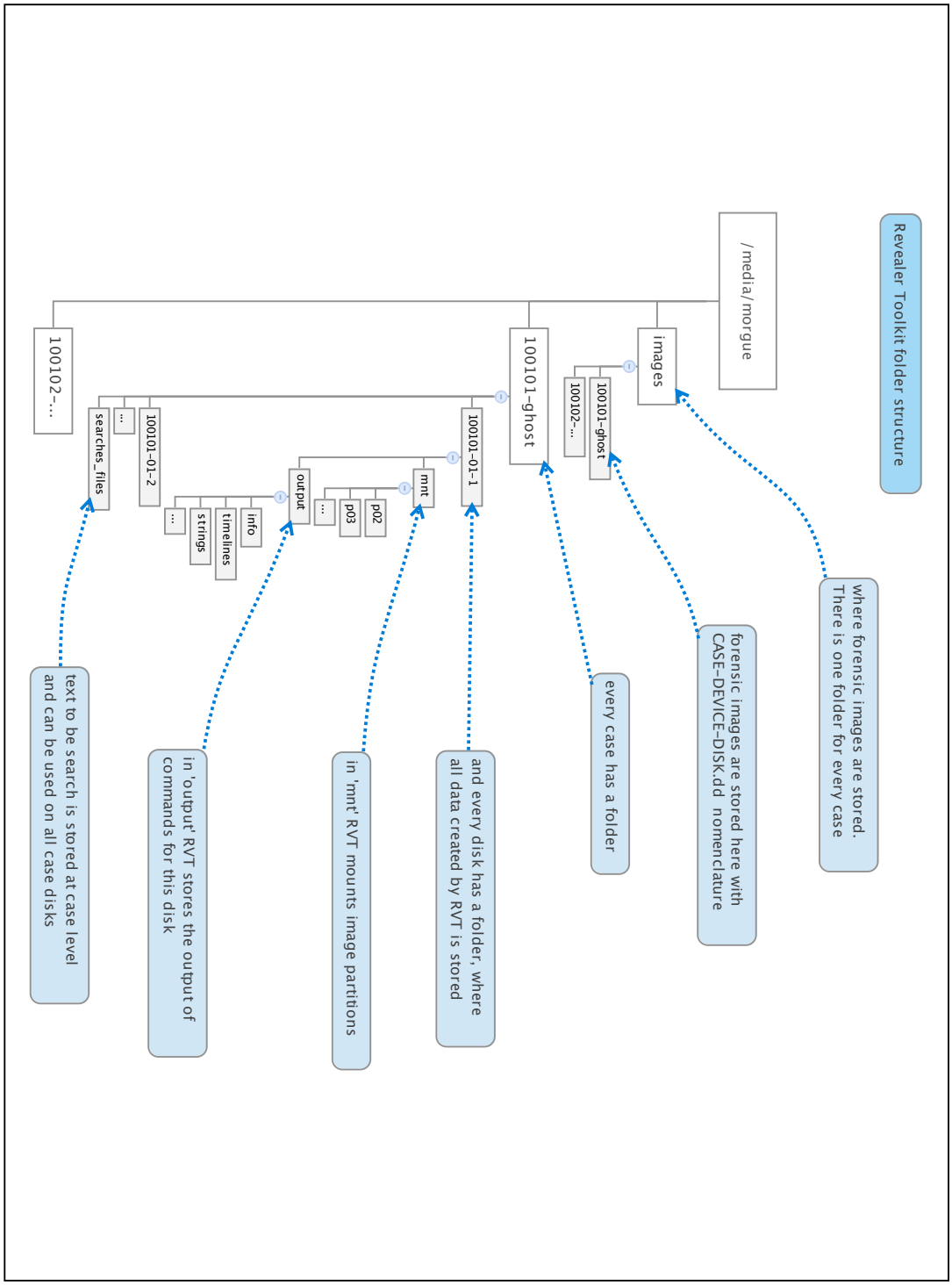

Figure 5.1: RVT folder structure

5.3 Devices, disks and partitions

Under the Revealer Toolkit, information sources are organized with:

- Devices: each case has a number of devices: computers, cell phones, digital cameras, ... They are numbered sequencially from 01 to 99.

  A device is named as *casenumber-devicenumber*, for example, *100101-01* for the case 100101 and device 01.

- Disks: each device has a number of disks: hard disks, CD's, memory cards, ... They are numbered sequencially from 1 to 9

  A disk is named as *case-device-disknumber*, for example, *100101-01-1*, for the disk 1

- Partitions: each disk can have several partitions, numbered from 01 to 99. The numeration used by the Sleuthkit command *mmls* is used.

  A partition is noted as *case-device-disk-ppartition*, for example, 100101-01-1-p02 for partition 02.

Under each case folder, a folder must exist for every disk to be analyzed.

This folder structure, at a disk level, for the example installation shown in chapter 4, will be:

```
/media/morgue/100101-ghost
/media/morgue/100101-ghost/100101-01-1

/media/morgue/imagenes/100101-ghost
/media/morgue/imagenes/100101-ghost/100101-01-1.dd
```

5.4 Forensic results

The Revealer Toolkit Shell manages and executes *script modules*, which performs forensic operations on the disk images and disk information. The results of these script modules are stored in the corresponding disk folder, under a folder named *output*.

The folder structure and file content stored here depends on each script module. See chapter 7 for further information.

Furthermore, under the disk folder other folder exists, named *mnt*, that contains the mounting points for the image partitions.

Then, the complete folder structure for the example shown in chapter 4 will be:

```
/media/morgue/100101-ghost
/media/morgue/100101-ghost/100101-01-1
```

```
/media/morgue/100101-ghost/100101-01-1/mnt
/media/morgue/100101-ghost/100101-01-1/output

/media/morgue/imagenes/100101-ghost
/media/morgue/imagenes/100101-ghost/100101-01-1.dd
```

Chapter 6

Command guide

The Revealer Toolkit Shell provide several commands used to (a) manage your forensic images and (b) execute forensic operations over them.

RVT Shell is a Perl script that, when executed, performs a scan of the morgue and of all the images stored into it. After that, a prompt is shown and commands can be introduced.

6.1 General interaction

- to execute one command, type it at the prompt, add the corresponding arguments, and type RETURN. Some commands return information on the screen, some, write information at the *output* folder of the corresponding disk at the morgue. Other, do both.
- type one command followed by *?* to obtain help about it
- type the TAB key to obtain a list of available commands

```
Welcome to Revealer Tools Shell (v0.2):

 RVT  >
        case
        cluster
        images
        info
        inode
        losetup
        mount
        script
        set
        test
 RVT  >
```

6.2 Cheatsheets

For those who know the name of the game, the following tables could be used as quick reference.

- Command quick reference, Table 6.1, page 15
- Script modules quick reference, Table 6.2, page 16

6.3 Command: *case*

Case management.

6.3.1 *case list*

Gives a list of the cases stored in the morgue.

```
 RVT  > case list
Cases in the morgue:
        100101 'ghost':
                100101-01-1
```

6.4 Command: *images*

Image management.

6.4.1 *images list*

Gives a list of the disk images stored in the morgue.

```
 RVT  > images list
Images in the morgue:
        100101 'ghost':
                100101-01-1.dd
```

6.4.2 *images partition info*

Gives information about a partition.

case list lists cases in the morgue
images list lists images in the morgue
images partition info <partition> prints partition's information **images partition table <disk>** prints partitions of a disk **images loadconfig** Loads the morgue config from XML file
info list shows some morgue's information **info debug <dumpmorguexml|dumprvtcases|dumprvtcfg>** prints part of the RVT internal configuration
losetup assign <disk|partition> assigns a loop device to the partition **losetup delete <disk|partition>** deletes a loop device assign to the partition **losetup list** lists the losetups devices assigned to RVT partitions **losetup recheck** updates RVT internal configuration on loop devices
mount assign <disk|partition> mounts a partition **mount delete <disk|partition>** umounts a partition **mount list** list mounted RVT partitions **mount recheck** updates RVT internal configuration on mounted partitions
cluster allocationstatus <cluster> <partition> prints cluster allocation status **cluster extract <raw|ascii> <cluster[,num]> <partition>** extracts a cluster, or *num* clusters, from partition **cluster generateindex <disk>** creates files for a cluster-inode quick resolution **cluster toinode <cluster> <partition>** prints all inodes associated with a cluster
inode allocationstatus <inode> <partition> prints inode allocation status
set level [<case|device|disk|partition>] sets working case level

Table 6.1: Command quick reference

script filelist generate <disk> generates lists of files for each partition
script files allocfiles <disk> generates lists of allocated files for each partition
script search clusterlist <file> <disk> builds a list of clusters and file paths from a previous search **script search clusters <file> <disk>** extract clusters that match a previous search **script search file <delete|edit|list|show>** manages search files **script search launch <file> <disk>** launches a search **script search quickcount <name:regexpr> <disk>** counts the appearance of a regular expression. Predefined regexpr: *emails, accounts, ips.*
script strings generate <disk> generates strings for all partitions of a disk
script timelines generate <disk> generates timelines for all partitions of a disk
script regripper listmodules list available RegRipper modules **script repripper execmodule <plugin> <hivetype> <partition>** executes one RegRipper module on the last modified hive of the type specified partition **script regripper execallmodules <hivetype> <partition>** executes all RegRipper modules on all hive files on the specified partition
script mail parsepsts <partition> parse all PST files on partition

Table 6.2: Modules quick reference

```
 RVT  > images partition info 100101-01-1-p03

Info for partition  - 100101-01-1-p03:

Filesystem:     FAT12
Cluster size:   2048
Sector size:    512
Offset:         0000010240 sectors ( 5242880 bytes )
```

6.4.3 *images partition table*

Gives the partition table of an image.

```
 RVT  > images partition table 100101-01-1

        03:     6 MB    Linux (0x83)
        02:     5 MB    Linux (0x83)
```

6.4.4 *images scanall*

RVT scans the morgue. No output given.

Necessary when adding a new image to the morgue. You can check what RVT thinks that is in your morgue executing *info debug dumpmorguexml* (see 6.5.2)

6.4.5 *images loadconfig*

Loads the morgue config from XML file.

6.5 Command: *info*

Manages RVT Shell configuration information.

6.5.1 *info list*

Gives RVT Shell configuration information.

```
 RVT  > info list

        List of morgues:
                /media/morgue
                /media/datos


        List of morgues of images:
```

```
                /media/morgue/imagenes
                /media/datos/imagenes
```

6.5.2 *info debug*

dump the content of some system variables, like RVT configuration or morgue status:

- *info debug dumpmorguexml*: dumps content of *RVTmorgueInfo.xml* as imported by RVT
- *info debug dumprvtcases*: dumps content of internal variable *RVT_cases* (for developers only)
- *info debug dumprvtcfg*: dumps content of current RVT configuration

```
 RVT  > info debug dumprvtcfg

\$VAR1 = {
          'morgueInfoXML' => '/media/morgue/RVTmorgueInfo.xml',
          'log_level' => '0',
          'mount_gid' => '1010',
          'mount_umask' => '007',
          'paths' => [
                     {
                       'images' => [
                                     '/media/morgue/images'
                                   ],
                       'morgues' => [
                                      '/media/morgue'
                                    ],
                       'tmp' => '/tmp'
                     }
                   ],
          'tsk_path' => '/usr/local/bin'
        };
```

6.6 Command: *losetup*

Loop device management.

6.6.1 *losetup assign*

Assigns a loop device to each partition of a case, disk or partition

```
 RVT  > losetup assign 100101

sudo losetup -f /media/morgue/imagenes/100101-ghost/100101-01-1.dd -o 5242880

sudo losetup -f /media/morgue/imagenes/100101-ghost/100101-01-1.dd -o 5120
```

6.6.2 *losetup delete*

Delete loop devices assigned to a case, disk or partition

```
 RVT  > losetup delete 100101
```

6.6.3 *losetup list*

List all loop devices assigned.

```
 RVT  > losetup list
Loop devices:
        loop1   100101-01-1.dd  5120
        loop0   100101-01-1.dd  5242880
```

6.6.4 *losetup recheck*

Updates loop device information from the operating system.

6.7 Command: *mount*

Mount points management.

6.7.1 *mount assign*

Mounts the partitions of a case, disk or partition at the path *morgue/100xxx-case/100xxx-device-disk/mnt/p0N*, where N is the partition number.

```
 RVT  > mount assign 100101

sudo mount /media/morgue/imagenes/100101-ghost/100101-01-1.dd /media/morgue/100101-g

sudo mount /media/morgue/imagenes/100101-ghost/100101-01-1.dd /media/morgue/100101-g
```

6.7.2 *mount delete*

Umounts partitions of a case, disk or partition

```
 RVT  > mount delete 100101
```

6.7.3 *mount list*

List all mounted points.

```
 RVT  > mount list
Mounted partitions:
        100101-01-1.dd  loop=/dev/loop0 offset=5242880
        100101-01-1.dd  loop=/dev/loop1 offset=5120
```

6.7.4 *mount recheck*

Updates mount points information from the operating system.

6.8 Command: *cluster*

Operations on clusters.

6.8.1 *cluster allocationstatus*

Prints cluster allocation status

```
 RVT  > cluster allocationstatus 2 100101-01-1-p02
Cluster 2: Allocated (Meta)
```

6.8.2 *cluster extract*

Prints the contents of the cluster. First argument specifies a post-treatment done to the cluster content before printing:

- raw: prints original cluster without changes
- ascii: only prints ascii characters. Others are substituted with a dot

Second argument is the cluster number to be extracted. When the cluster number is appended with a comma and a integer number, extracts this number of consecutive clusters. For example, *cluster extract raw 100,2 100101-01-1-p02* extracts clusters 100 and 101.

```
 RVT  > cluster extract ascii 3 100101-01-1-p02
-------------------------------------------
/usr/bin/blkcat -o 0000000001 /media/morgue/imagenes/100101-espejismo/100101-01-1.dd
```

6.8.3 *cluster generateindex*

Creates sort of an index for quick cluster-to-inode resolution. Required for performing searches. This is one of the few commands that writes files on the morgue.

See *script search* (página 35 module for more information.

6.8.4 *cluster toinode*

Prints all the inodes associated with a cluster.

```
 RVT  > cluster toinode 2 100101-01-1-p02

inodes:

159700
```

6.9 Command: *inode*

Operations on inodes

6.9.1 *inode allocationstatus*

Prints inode allocation status

```
 RVT  > inode allocationstatus 2 100101-01-1-p02

inode 2: Allocated
```

6.10 Command: *script*

Modular scripts. Each module is explained in a separated chapter (see Chapter 7 for further information)

6.11 Command: *set*

Sets RVT Shell configuration information.

6.11.1 *set level*

Sets the work level to a specific case, device, disk or partition. When the work level is stablished, is notified at the prompt and there is no need of indicate it as argument at the commands.

```
 RVT  > images partition table 100101-01-1

        03:     6 MB    Linux (0x83)
        02:     5 MB    Linux (0x83)

 RVT  >
```

```
 RVT  > images partition table


I don't know what is this

 RVT  > set level 100101-01-1

 new format: disk
 RVT 100101-01-1 > images partition table

        03:     6 MB    Linux (0x83)
        02:     5 MB    Linux (0x83)

 RVT 100101-01-1 >
```

Chapter 7

Script modules

Scripts modules are forensic software components that perform specific forensic tasks on the stored images and information.

Each module has particular objectives, methods, arguments and results, the documentation of which is detailed in the following sections.

7.1 *script filelist*

Module for automatic generation of file lists in every partition. Mainly oriented to Windows filesystems, does not show permissions, owner, or group.

7.1.1 *script filelist generate*

Generates file lists for all partitions of a disk

```
script filelist generate <disk>
```

It creates several files in the morgue with this path and name:

```
output/FileList/<disk>-p<partition>_FileList.csv
```

7.2 *script files*

Module for performing operations on mounted images.

7.2.1 *script files allocfiles*

Creates a file with a list of all the allocated files of the image.

```
files allocfiles <disk>
```

It creates a file in the morgue with this path and name:

```
<morguepath>/<case>/<disk>/output/info/alloc_files.txt
```

with a list of allocated files and folders.

7.3 *script search*

Module for performing searches. See the Appendix 8.5 for a step-by-step tutorial.

7.3.1 *script search clusterlist*

Builds a list of clusters and file paths that matches a previous search.

```
script search clusterlist <search file> <image>
```

It creates several files in the morgue with this path and name:

```
<morguepath>/<case>/<disk>/output/searches/cbusq_<search>-<partition>
<morguepath>/<case>/<disk>/output/searches/pbusq_<search>-<partition>
```

with this format:
cbusq:

```
<cluster>:<inode>:<allocation status>:<file path>
```

pbusq:

```
<file path> (<allocation status)
```

7.3.2 *script search clusters*

Extract the clusters matched in a previous search

```
script search clusters <search file> <image>
```

It creates several files in the morgue with this path and name:

```
<morguepath>/<case>/<disk>/output/searches/ibusq_<search>-<partition>
```

with this format:

```
---------------------------
<byte offset>:<cluster>:<allocation status>:  <line that matches search>

<cluster content>
```

7.3.3 *script search file*

Commands to manage search files:

- script search file delete <name>: deletes file
- script search file edit <name>: creates and/or edit file with *vim*
- script search file list: lists search files created
- script search file show <name>: prints file's contents

The files are created per case, and are stored at this path:

```
<morguepath>/<case>/searches_files
```

7.3.4 *script search launch*

Launches a search in an image.

```
script search launch <search file> <image>
```

It creates several files in the morgue with this path and name:

```
<morguepath>/<case>/<disk>/output/searches/busq_<search>
```

with this format:

```
<strings file that matches>: <byte offset> <line that matches>
```

7.3.5 *script search quickcount*

Launch a quick search in a case or in an image with a count of the results.

```
script search quickcount <name:regular expression>  <image>
```

It creates several files in the morgue with this path and name:

```
<morguepath>/<case>/<disk>/output/info/count_<name>
```

with this format:

```
<times that the match appear>  <match>
```

There are some quickcounts preconfigured:

- script search quickcount emails <image>: search emails addresses
- script search quickcount accounts <image>: search bank accounts
- script search quickcount ips <image>: search ip addresses
- script search quickcount phones <image>: search phone numbers

7.4 *script strings*

Module for creating and managing string files of images.

7.4.1 *script strings generate*

Generates strings for all partitions of a disk

```
script strings generate <disk>
```

It creates several files in the morgue with this path and name:

```
<morguepath>/<case>/<disk>/output/strings/strings-<case>-<partition>.asc
<morguepath>/<case>/<disk>/output/strings/strings-<case>-<partition>.uni
```

Strings files of ASCII characters have *.asc* extension.
Strings files of Unicode (UTF8) characters have *.uni* extension.

7.5 *script timelines*

Module for creating and managing timelines.

7.5.1 *script timelines generate*

Generates timelines for all partitions of a disk

```
script timelines generate <disk>
```

It creates several files in the morgue with this path and name:

```
output/timelines/<disk>-p<partition>_iTL.csv
output/timelines/<disk>-p<partition>_iTL-day.sum
output/timelines/<disk>_TL.csv
output/timelines/<disk>_TL.txt
output/timelines/<disk>_TL-day.sum
output/timelines/<disk>_TL-hour.sum
```

Where *i*timelines (from now on **iTL**) are calculated based on inodes, and timelines, with file names. *day* and *hour* sums are line counts grouped by days and hours.

7.6 *script webmail*

Module for detecting and managing webmail.
Deprecated. Has to be rewritten.

7.7 *script software*

Module for detecting software.

Deprecated. Has to be rewritten.

7.8 *script regripper*

Module for parsing Microsoft Windows registry hives using RegRipper (www.regripper.net). See section B.3 for notes about its installation.

7.8.1 *script regripper listmodules*

List all RegRipper available modules. If this command does not return anything, please refer to section B.3 for notes about RegRipper installation.

7.8.2 *script regripper execmodule*

Executes one RegRipper module of a particular hive type on a partition. RVT picks the most suitable hive file (simply, it gets the most recent modified hive), and prints results on the screen.

```
script regripper execmodule <plugin> <hivetype> <partition>
```

where <hivetype> can be one of these: sam, security, software, system or ntuser

7.8.3 *script regripper execallmodules*

Execute all RegRipper modules on all the hive files on a partition.

```
script regripper execallmodules  <hivetype> <partition>
```

where <hivetype> can be one of these: sam, security, software, system, ntuser **or** all

It stores the results on this folder *output/regripper*, one file result per hive file, with this format:

```
output/regripper/<hivetype>-<last modified data>
```

The first line of each results file have the path to the hive file on which RegRipper has been executed.

7.9 *script mail*

Module for extract emails in text format from Microsoft Outlook mailboxes (PST) using *libpst* (www.five-ten-sg.com/libpst).

7.9.1 *script mail parsepsts*

Search all Microsoft Outlook mailboxes (PST extension) and parse them with *libpst*

```
script mail parsepsts <partition>
```

Results are stored under *output/mail*, where a folder is created for each PST file with the format:

```
output/mail/pst-n
```

where n is a increasing integer number.

7.10 *script lnk*

Parsing and management of Microsoft Windows LNK files.

7.10.1 *script lnk generate*

Parse all LNK files of the disk (only those with *lnk* extension).

```
script lnk generate <disk>
```

Results are stored in a file, one line per lnk file, in the following path:

```
output/lnk/<disk>_lnk.csv
```

7.11 *script report*

Generates reports (usualy pdf files)

7.11.1 *script report search2pdf*

Generates pdf files from the 'ibusq' files previously generated by the *script search clusters* command and located at:

```
<morguepath>/<case>/<disk>/output/searches/ibusq_<search>
```

Results are stored under *output/reports/searches* and contains a pdf file for each 'ibusq' file, with the following format:

```
output/reports/searches/ibusq_<search>.pdf
```

Each pdf file contains all search terms highlighted and non printeble charachters are been replaced by dots (.).

Chapter 8

Operational guides and tricks

8.1 RVT Shell tricks

The Revealer Toolkit can accept commands by several ways. One of them is through a shell. This section will provide you with some tricks to ease your interaction with RVT.

Yes, you will feel that RVT Shell is buggy and that lacks support for some keys, terminals and functionality, but, well, I made it myself and it is not perfect. It will be improved or substituted in future versions. Suggestions are welcome.

8.1.1 command level

Commands in RVT are *nested* in levels. Each command level can contain commands, but also other command levels. You can enter a level entering its name in RVT Shell, and the RVT prompt will be modified.

For example, in order to execute the following commands, they can be written directly to RVT ...

```
RVT > info list
...
RVT > info debug dumpmorguexml
...
RVT > info debug dumprvtcases
...
RVT > info debug dumprvtcfg
...
```

... but the following session, taking profit of command levels, also is possible:

```
 RVT  > info
```

```
RVT  info> list
...
RVT  info> debug
RVT  info debug> dumpmorguexml
...
RVT  info debug> dumprvtcases
...
RVT  info debug> dumprvtcfg
...
```

To go up one level, enter '*r*':

```
RVT  info debug> r
RVT  info> r
RVT  >
```

8.1.2 case level

When working with a case or disk, usually you will have to type a lot of types this case or disk number because a lot of commands need it as argument. The *set level* command (see section 6.11.1) will ease you user experience because it fixes a 'case level' for the session. Commands that expect a case, device, disk or partition as an argument will use the 'case level' established by *set level* when the argument is not provided.

The case level can be changed with other *set level* command, and erased when executing *set level* without arguments.

For example:

```
RVT  > images partition table 100101-01-1

       03:     6 MB    Linux (0x83)
       02:     5 MB    Linux (0x83)

RVT  >
RVT  > images partition table


I don't know what is this

RVT  > set level 100101-01-1

new format: disk
RVT 100101-01-1 > images partition table
```

```
        03:     6 MB    Linux (0x83)
        02:     5 MB    Linux (0x83)

 RVT 100101-01-1 >
```

Moreover, case level can be determined as an argument to RVT itself:

```
$ rvt -l 100101-01-1
Scanning morgues. Please wait ...



Welcome to Revealer Tools Shell (v0.2):


 new format: disk
 RVT 100101-01-1 >
```

8.1.3 TAB key

When using the RVT Shell, the TAB key can be used in two ways:

1. at a command level, before writing, will show the available commands and levels
2. when a command or level is half written, will show the available command and levels that begin with the fragment that is written

For example:

```
 RVT  >  <TAB>
case
cluster
images
info
inode
losetup
mount
script
set
test
 RVT  > i<TAB>
images
info
inode
```

```
 RVT  > in<TAB>
info
inode
```

8.1.4 command history

RVT Shell provide a very simple history feature, which consist of remembering the last command typed, that can be accessed type the up arrow.

Future versions will improve the command history with bigger *memory*.

8.2 entities expansion

Several commands recive a entity as an argument (a case, a device, a disk or a partition). In some cases, is useful that a command could be applied to all the subentities below a entity of higher level. This could be achieved with *entities expansion*.

For example, the command

```
mount assign 100101-01-1-p02
```

will mount the 02 partition of the disk *100101-01-1*. But, what if 100101-01-1 have six partitions? and what if 100101 case have ten disks, every one of them with six partitions?

RVT will apply, automaticaly, a command to a group of subentities below a entity of higher level when the operands '@devices', '@disks' or '@partitions' are used just after the entity argument.

For example, the following command will mount all partitions of all disks of all devices of a case:

```
mount assign 100101@partitions
```

Other example: the following command will launch a search on all de disks of a case:

```
script search launch mysearches 100101@disks
```

Note: you will see that *mount assign 100101*, without entity expansion, also mounts all partitions of the case 100101. The reason is that some commands, coded before entity expansion, have internal support for managing several types of entites

8.3 chaining commands

Some of the RVT commands can take a long time to execute. If want to execute a list of commands sequencially you can choose between two options.

On one hand, RVT Shell will execute sequencially commands separated by ';'. One example:

```
 RVT  > case list; images list

Cases in the morgue:
100101 'espejismo':
100101-03-1
100101-02-1


Images in the morgue:
100101 'espejismo':
100101-01-2.dd
100101-01-1.dd
100101-03-1.dd
100101-02-1.dd
```

On the other hand, you can create a file containing a command on each line, and pipe them to RVT in batch mode (-b argument). For example:

```
$ cat commands.rvt
case list
images list
$
$ cat commands.rvt | rvt -b

Welcome to Revealer Tools Shell (v0.2):

 RVT  > case list

Cases in the morgue:
100101 'espejismo':
100101-03-1
100101-02-1

images list

Images in the morgue:
100101 'espejismo':
100101-01-2.dd
```

```
100101-01-1.dd
100101-03-1.dd
100101-02-1.dd

$
```

and, of course:

```
$ echo "case list; images list" |  rvt -b

Welcome to Revealer Tools Shell (v0.2):

 RVT  > case list

Cases in the morgue:
100101 'espejismo':
100101-03-1
100101-02-1

images list

Images in the morgue:
100101 'espejismo':
100101-01-2.dd
100101-01-1.dd
100101-03-1.dd
100101-02-1.dd

$
```

8.4 Logs

RVT will log its activity to the local syslog, including commands launched, time and date, error messages, IP from the user (in the case that access to the RVT server is done with ssh), and a bit more.

For example, this unix command will show you the last RVT log messages:

```
# grep 'RVT: ' /var/log/syslog
Jun 12 19:31:42 revealer RVT: INFO root@192.168.1.10 executing : RVT_images_list
Jun 12 19:31:42 revealer RVT: INFO root@192.168.1.10 ending session
Jun 12 19:31:53 revealer RVT: INFO root@192.168.1.10 starting up RVT v0.2
Jun 12 19:31:53 revealer RVT: INFO root@192.168.1.10 loading configuration
Jun 12 19:31:53 revealer RVT: INFO root@192.168.1.10 Morgue XML configuration loaded. Last updated: 20
```

```
Jun 12 19:31:53 revealer RVT: INFO root@192.168.1.10 Run 'images scanall' command to update
Jun 12 19:31:53 revealer RVT: INFO root@192.168.1.10 RVT shell started
Jun 12 19:31:53 revealer RVT: INFO root@192.168.1.10 executing : RVT_case_list
Jun 12 19:31:53 revealer RVT: INFO root@192.168.1.10 executing : RVT_images_list
Jun 12 19:31:53 revealer RVT: INFO root@192.168.1.10 ending session

...
```

and this, the commands on a particular disk:

```
# grep '100101-03-1' syslog | grep 'executing'

Jun 12 16:17:56 revealer RVT: INFO root@192.168.1.10 executing : RVT_images_partition_table 100101-03-
Jun 12 16:18:37 revealer RVT: INFO root@192.168.1.10 executing : RVT_set_level 100101-03-1
Jun 12 16:18:43 revealer RVT: INFO root@192.168.1.10 executing 100101-03-1: RVT_script_timelines_gener
Jun 12 16:19:06 revealer RVT: INFO root@192.168.1.10 executing : RVT_set_level 100101-03-1
Jun 12 16:19:11 revealer RVT: INFO root@192.168.1.10 executing 100101-03-1: RVT_script_timelines_gener
Jun 12 16:19:27 revealer RVT: INFO root@192.168.1.10 executing 100101-03-1: RVT_mount_assign
Jun 12 16:19:36 revealer RVT: INFO root@192.168.1.10 executing 100101-03-1: RVT_script_files_allocfile
Jun 12 18:34:18 revealer RVT: INFO root@192.168.1.10 executing : RVT_inode_allocationstatus 2 100101-0
Jun 12 18:34:44 revealer RVT: INFO root@192.168.1.10 executing : RVT_inode_allocationstatus 2 100101-0
Jun 12 18:53:38 revealer RVT: INFO root@192.168.1.10 executing : RVT_mount_delete 100101-03-1-p02

...
```

8.5 Searches

The Revealer Toolkit can perform searches on forensic images. Several commands are required to be executed to perform a search, that are discussed in this section.

The following two commands commands are required to be executed once before launching searches on a disk:

- *script strings generate <disk>*: extracts ascii and unicode strings from forensic image
- *cluster generateindex <disk>*: create a index for quick cluster–inode resolution

Next step is to create a file that contains the terms to be searched. These files can be created with RVT commands, and are visible to all disks on a case:

- *script file edit <file name>*: this command opens a *vim* file editor, that can be used to write a search term on each line. Writing the file and exiting *vim* returns to RVT and a file is created in the directory <case>/searches_files

Warning!: these rules apply when creating search terms (sorry, these limitations will be eliminated in future versions):

1. search is always case insensitive. Also, always write search terms in lower case

2. do not include characters outside ASCII charset (no accents, no ñ's, ...) So, for example, you cannot search "barça campió", so you have to search "a campi" instead.

3. some regular expression characters can be used, although escape sequences have not been well tested. So, you can use '.', '*', '?', but not backslash or complex regular expressions

Now, searches can be launched. Three commands create different output related with a search file:

- *script search launch <file name>*: greps every search term on <file name> over the strings files. Results are stored on *output/searches/busq_<search term>*

- *script search clusters <file name>*: for each *busq_* file generated before, extract the cluster that contains the term searched and writes the results on *output/searches/ibusq_<search term>*

- *script search clusterlist <file name>*: it takes each *busq_* file generated before and creates two new files: *cbusq_<search term>*, which contains a list of clusters that contain the searched terms, its allocation status and path if allocated, and *pbusq_<search term>*, which contains a list of paths associated with allocated clusters that contains search terms[1]

Now, you are ready to review megabytes and megabytes of results, that are stored under disk folder *output/searches*, in a black shell window :)

A typical search session could be like this, where the objective is to search the words 'revealer' and 'toolkit' on all disks of the case '100101-ghost':

```
RVT > set level 100101
RVT > script strings generate @disks
RVT > cluster generateindex @disks
RVT 100101 > script search
RVT 100101 script search > file edit myFile

('i' key pressed on vim)
revealer
```

[1]beware! if search terms are allocated on slack space, files could be included in the pbusq file that do actually not include this search term (because is on the slack space!)

```
toolkit
(ESC key pressed and then ':wq' to write and exit back to RVT)

RVT 100101 script search > launch myFile @disks; clusters myFile @disks;
 clusterlist myFile @disks
```

8.6 F-Response compatibility

The Revealer Toolkit is fully compatible with F-Reponse (www.f-response.com), that is, forensic tasks on remote F-Response disks can be automated through RVT.

But, RVT is *optimized* to be used with F-Response, so expect a considerable load of time and network traffic on some operations.

In order to use a F-Response created device, or, in general, any disk device as a RVT image, create a symbolic link to it from the *images* directory (f.ex., *ln -s /dev/sdx /media/morgue/images/ 100101-ghost/100101-02-1.dd*)

Future versions will include RVT commands for managing F-Response targets and devices.

8.7 Known problems

Remember that RVT is not stable yet, so there are some problems pending to be solved in future versions:

- error verbosity: RVT do not offer a lot of verbosity when an error occurs
- commands dependences: to be executed, each command has a list of dependences: executables installed on the system, other commands have to be executed before, arguments, ... Sometimes, a command fails because one dependence is not found and is not checked automatically before executing
- keyboard does not work as expected: RVT Shell code is very simple, and it does not handle properly a lot of keyboard actions, as movement arrows, delete keys, etc. This problem is worse when working with remote terminals, screens, and other shell tricks
- search engine has to be improved
- ...

Appendix A

Sample image from scratch

Sometimes could be useful to create an image from scratch. A simple procedure is provided in order to create such image from linux.

as user:

```
cd /media/morgue/imagenes/100101-ghost/
dd if=/dev/zero bs=1024 count=10240 > 100101-01-1.dd
```

as root:

```
fdisk /media/morgue/imagenes/100101-ghost/100101-01-1.dd

in fdisk:

x (additional functions)

c (cylinders)
1024
s (sectors)
10
h (heads)
2

r (main menu)
n (new partition)
p
1
1   (first cylinder)
512 (half of the disk)
n (new partition)
p
2
```

```
513 (half of the disk)
1024 (last cylinder)
w (save and exit)
```

and now, let's put some info inside, as root:

```
echo "images scanall; losetup assign 100101-01-1; losetup list" | RVT.pl -b
   (the last command reveals loop devices created)
mkfs.vfat /dev/loop0
mkfs.vfat -F 32 /dev/loop1

mkdir /media/aux1
mount /dev/loop0 /media/aux1
cp /home/analyst/RVT/RVT.pl /media/aux1
umount /media/aux1
mount /dev/loop1 /media/aux1
cp /home/analyst/RVT/RVT.pl /media/aux1
sync
rm /media/aux1/RVT.pl
echo "my email address is myemail@revealertoolkit.com" > /media/aux1/textfile.txt
umount /media/aux1

echo "losetup delete 100101-01-1" | RVT.pl -b
```

Appendix B

Additional software installation

B.1 RVT tools

The Revealer Toolkit is not just a framework, but also include some little but nice tools that are shipped with the source code of the application under the folder, guess, *tools*.

In order to some RVT functions to be operative, make sure that all the tools are on the system PATH.

At this time, these are the tools included:

- *dumplnk.pl:* code adapted from Jacob Cunningham's lnk-parse, which parses the contents of Microsoft Windows LNK files and output it in CSV format
- *csv2latex.pl:* useful script to fill LaTeXtables with the content of a csv file

- $guess_csv.pl : guessestheseparatorcharacterofavalue-separatedfile$ $plot_bars.pl : usinggnuplot, gener$

B.2 libpst Utilities

Download lastest stable package from http://www.five-ten-sg.com/libpst/packages, unzip and compile. After that, copy or link *readpst* executable to */usr/local/bin*.

- ```
  cd /usr/local/src
  mkdir libpst
  cd libpst
  wget http://www.five-ten-sg.com/libpst/packages/libpst-0.6.37.tar.gz
  tar -zxf libpst-0.6.37.tar.gz
  cd libpst-0.6.37
  ```
  ```

```
./configure
make
cp src/readpst /usr/local/bin
```

B.3 RegRipper

Download lastest stable package from www.regripper.net and unzip:

```
cpan Parse::Win32Registry

mkdir /usr/local/src/regripper
cd /usr/local/src/regripper
wget http://www.regripper.net/RegRipper/RegRipper/rr_20080909.zip
unzip rr_20080909.zip

apt-get install tofrodos
dos2unix rip.pl
```

some changes have to be made on RegRipper code to be executable on Linux:

```
1c1
< #! c:\perl\bin\perl.exe
---
> #!/usr/bin/perl
29c29
< my $plugindir = "plugins\\";
---
> my $plugindir = "/usr/local/src/regripper/plugins/";
92c92
<  require "plugins\\".$plugins{$i}."\.pl";
---
>  require $plugindir.$plugins{$i}."\.pl";
```

and final touches:

```
perl rip.pl -l > plugins/plugins

ln -s /usr/local/src/regripper/rip.pl /usr/local/bin/rip
chmod a+x /usr/local/src/regripper/rip.pl
```