Master's thesis

2019

Horst Dumstorff

Master's thesis

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Information Security and Communication
Technology

NTNU
Norwegian University of
Science and Technology

Horst Dumstorff

Virtualize A Piece Of Evidence Or Mount Its Partition With Linux

May 2019

NTNU
Norwegian University of
Science and Technology

Norwegian University of
Science and Technology

Virtualize A Piece Of Evidence Or Mount Its Partition With Linux

MISEB

Submission date:	May 2019
Supervisor:	Prof. Dr. Katrin Franke, NTNU
Co-supervisor:	Police Superintendent Kurt H. Hansen (MSc), PHS

Norwegian University of Science and Technology
Department of Information Security and Communication Technology

Acknowledgement

I would like to thank my supervisor, Professor Dr. Katrin Fanke and Police Superintendent Kurt H. Hansen (MSc), for their patience and dedication.

I thank the Datenverarbeitungsgruppe (DVG) of the Polizeiinspektion Cloppenburg for the support, in particular, Joachim Janßen.

Furthermore, I want to thank Mary Schwarzbach and my family, especially Maik Dumstorff, for proofreading this document.

I would also like to thank my family for relieving me during this time, so that I could concentrate fully on the master thesis.

H.D.

Abstract

In a computer forensic investigation, there is always a divergence between the time required to visualize data from evidence and the times when investigators need the data to evaluate it.
On the one hand, the processing time that is required for the analysis of existing and deleted data, and on the other hand, the time that elapses to make potentially existing data visible. Data, which the investigator needs immediately because it can be crucial for solving the case.
The motivation for this master thesis is to shorten the time it takes to invest until the first sighting of data.
More and more often, offenders try to disguise their illegal activities on data carriers with new methods and acquired expertise. Therefore, it is important for the investigator to be able to access hidden or deleted data. This in turn, depending on the amount of data and complexity, often entails immense processing time. The first step of the forensic-process is to create a forensic clone of a disk image and store it in a file. A clone meaning a copy of the evidence data. With this copy further forensic investigations are then made.
In special cases, such as terrorism or hostage-taking means, it applies in a very short reaction time, to being able to access data from the evidence without changing the evidence itself (e.g., changing timestamps). With the results of the forensic investigation, follow-up measures can then be recognized and initiated. These follow-up measures are required to take place immediately. Under certain circumstances, indications of further assassination, accomplices or escape routes can be obtained from this data. This would prevent the possibility of imminent actions, thus save lives and/or arrest the perpetrators which in turn prevent further acts. That would not be possible without an immediate inspection of the data. Now, following the well-tried method, a lot of time would be lost until the first data could be used from the evidence.
To produce this result, Information Technology (IT) specialists (forensic scientists) are needed to process this data and make it available to the investigator in an appropriate manner so that the investigator is able to use the data in the ongoing investigation. This takes a lot of time. Time and knowledge are the limits of this method, currently.

The aim of this master thesis is to study whether an investigator without the special IT know-how can work out the required data and everything in the shortest possible time.
Two ways would be conceivable, on the one hand with standard Linux programs to mount partitions from the evidence (hard disk or image), on the other hand with standard Linux programs to virtualize the evidence as ”Virtual Machine”, in a new way. Now the detailed results would be available in a much shorter time even without an IT specialist. If the investigator is able to view the data of the evidence, he can review the possible data contents of the applications (e-mail, chats, internet protocol, own documents and recently used documents, etc.). He does not need any particular knowledge of the structure of the used file system, the can simply start the applications and watch the results as if he were operating the physical PC. It may then be very important to use this data in the ongoing investigation and deduce further follow-up.

List of Appended Papers

In the specialization course, I wrote about the forensic data backup of mobile Apple phones, the iPhone. This master thesis refers to some passages. These parts have been adapted to this work or, if possible, updated.

Dumstorff, H., Forensic Data Backup Of Mobile Apple Phones, IMT4215 Specialization Project.

Table of Contents

List of Tables

List of Figures

Abbreviations

AFF	Advanced Forensic Format
AMD	Advanced Micro Devices
API	Application Programming Interface
APT	Advanced Package Tool
ARM	previously Advanced RISC Machine, originally Acorn RISC Machine
BIOS	Basic Input Output System
bit	Binary Digit
BSI	Bundesamt für Sicherheit in der Informationstechnik
CD	Compact Disk
CD-ROM	Compact Disc Read-Only Memory
CPU	Central Processing Unit
CRIS	Code Reduced Instruction Set
dd	Data Definition
DMG	Disk iMaGe, Apple Disk Image
DOS	Disk Operating System
DVD	Digital Video Disc
E01	Encase image file format
EDK	EFI Development Kit
EFI	Extensible Firmware Interface

ETFS Embedded Transaction Filesystem

EWF Expert Witness Compression Format

F2FS Flash-Friendly File System

FAT File Allocation Table

FTK Forensic Toolkit

FUSE File system in USErspace

GB Giga Byte

GNU GNU's Not Unix!

GPT GUID Partition Table

GRAND GRand Unified Bootloader

GUI Graphical User Interfacem

HDD Hard disk drive

IBM International Business Machines Corporation

IDE Integrated Development Environment

ISA Industry Standard Architecture

ISO International Organization for Standardization

IT Information Technology

JFFS Journaling Flash File System

KB Kilo Byte

KDE K Desktop Environment

KSM Kernel Samepage Merging

KVM Kernel-based Virtual Machine

LILO LInux LOader

LTS long-term support

LUKS Linux Unified Key Setup

MB Mega Byte

MD	Message-digest
MIPS	Microprocessor without Interlocked Pipeline Stages
NAND	Not And (electronic logic gate)
NOR	Not OR Gate
NTFS	New Technology File System
NTNU	Norges teknisk-naturvitenskapelige universitet
OS	Operating System
OVMF	Open Virtual Machine Firmware
PC	Personal Computer
PCI	Peripheral Component Interconnect
PHS	Politihgskolen
QEMU	Quick EMUlator
RAM	Random-access memory
RISC	Reduced Instruction Set Computer
ROM	Read-Only Memory
RPM	Red Hat Package Manager
SATA	Serial Advanced Technology Attachment
SD	Secure Digital Memory Card
SHA	Secure Hash Algorithm
SPIFFS	File System for SPI NOR flash devices
SSD	Solid-State-Drive
TB	Tera Byte
TDK	Tky Denki Kagaku Kgy
TOR	The Onion Router
UEFI	Unified Extensible Firmware Interface
URPMI	User RedHat Package Manager Installation

USB	Universal Serial Bus
VGA	Video Graphics Array
VHD	Virtual Hard Disk
VM	Virtual Machine
VMDK	Virtual Machine Disk
Wi-Fi	Wireless Fidelity
WWW	World Wide Web
Xfce	XForms Common Environment
XML	Extensible Markup Language
Yaffs	Yet Another Flash File System

Chapter 1

Introduction

1.1 Target Group

This research is intended to assist law enforcement authorities in the investigation of criminal offenses. It should give investigators the opportunity to gain a first quick insight into electronic evidence.

This work contains detailed information about the virtualization process of a PC image. The secured data from the seized PC, which were performed from their data storages. They come from computers used for or involved in criminal offenses.

I would like to briefly explain the problem in the following example.

> In special cases, such as terrorism or hostage-taking, it shall a very short reaction time to gain insight into the data from the evidence without changing the evidence itself.
> The results of the forensic investigation can then be used to identify, detect and initiate follow-up actions. These follow-up actions must be carried out immediately. Under certain circumstances, indications of further attacks, accomplices or escape routes can be obtained from these data.
> This would prevent the possibility of forthcoming actions and perhaps lives could be saved. This could then perhaps prevent the perpetrators from further actions, which would not have been possible without verification of the data.

The aim of this research is to virtualize a piece of evidence with Open Source tools. An open source operating system, as well as an open source virtualization software, shall be used. The virtualization software should be parameterizable so that changes in the start command can be made quickly. These start commands must then still be storable so that they can be executed again at a later time without great effort.
Open source knowledge is a concept with freely available software tools including the corresponding source code. We are given tools with all the information that belongs to these programs. Open Source Knowledge means that this information, found or programmed by others, may be used, modified and distributed.

1.2 Motivation and Background

Computer Forensics, digital forensics or IT-Forensics began approximately at the end of the 19th century.
Digital forensics is concerned with the screening and analysis of digital traces in IT systems. The aim is detecting facts and possible perpetrators in case and computer crime and prepare digital evidence as well as preparing court-proof (Stefan Meier, 2016).
This is the core of forensic computer science. Forensic computer science refers to the application of scientific methods in computer science to questions of legal administration. This includes a wide range of questions in the broad context of digital examinations, for example, after appropriate examinations in certain cases.
All these questions are summarized under the umbrella term Digital Forensics. Digital forensics, therefore, deals with the search, identification, backup and analysis of digital tracks. (Horst Dumstorff, November, 2017).

I have been working in computer forensics for over 10 years now. Throughout this time, I have increasingly found that electronic evidence is the most important piece of data in criminal action procedures. Electronic evidence can clarify criminal cases most effectively. In 85% of criminal cases, electronic evidence is relevant to solving cases (Diana Nadeborn, 2018).
In most cases of fraud, counterfeiting or misuse, electronic evidence like pictures, videos or documents are crucial for solving the crimes. All these different types of data are stored on the internal memory of electronic device. Most electronic devices have a permanent memory that forensics can take advantage of. Since data has been stored permanently, in most case it can also be read out again.
While this research is being written, almost every household in the world has means of communication, such as a PC, a laptop, tablet, or smartphone, to connect to the Internet or connect to other users via the Internet. And these numbers are rising daily.

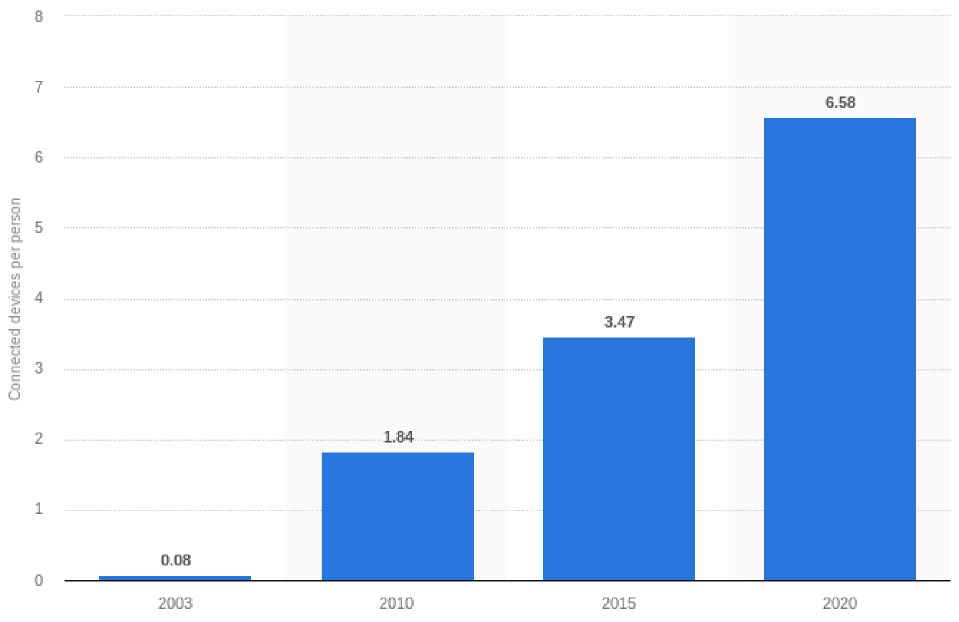


Figure 1.1: Number of network connected devices per person around the world from 2003 to 2020 [1]

In Figure 1.1 you can see that in 2015 there have been 3.47 of network-connected devices per person around the world. Until 2020 it is expected that there will be 6.58 of network connected devices per person around the world.
This, in turn, means that more seized devices that have been used for criminal matters have to be forensically evaluated every day. For each confiscated computer, an image must be created for each built-in hard disk. In order to illustrate the rapid increase in storage capacity, Chapter 2.2.1 gives a brief overview of the different types of data storage and their storage capacity. (Michael Hale Ligh, Andrew Case, Jamie Levy and AAron Walters, 2014; Margaret Rouse, 2015).
Every image then has to be prepared forensically by IT-specialists. Depending on the requirements, among other things, deleted data will be restored. The fact that after each forensic image of a hard disk still the preparation of an image is added, which leads to a huge amount of work. This takes a long time and the investigators are waiting for the evaluation results.And now there should, by using the results of there, the feasibility to use an ”unskilled person” to do the work.

There are several terms in forensic for an ”unskilled person”. In the article ”Tiered forensic methodology model for Digital Field Triage by non-digital evidence specialists” in the ELSEVIER Journal (Ben Hitchcock, Nhien-An Le-Khac, Mark Scanlon, 2016), there is the term ”Non Digital Forensic Investigators”. This article describes exactly the problem underlying this research. Due to a limited budget but high level of education, digital forensic analysts are scarce among police forces around the world. This article describes the training of front-line personnel in the field triage process without the need for an IT-forensic specialist.
From the concept of frontline members, a new process model is developed. The results of the implementation of this new model should show how specialists and non-specialists in digital evidence can better respond to the growing increase in digital evidence (Ben Hitchcock, Nhien-An Le-Khac, Mark Scanlon, 2016).
From the perspective of this research, any investigator or police officer who is not an IT specialist but has basic IT skills is an ”unskilled person”.

There are several persons in the forensic investigation and many of them are not IT specialists. Those, we would like to say, we define for the rest of the research as ”unskilled persons” in IT forensics.

1.3 Research Question

In a criminal offense, attempts should first be made to secure electronic evidence. If electronic evidence of a criminal offense is available, it should be searched for case-related data. This data may still exist in the file system of the evidence or may have been previously deleted.

Definition 1 (Image)**.** *To obtain a 100% copy of the evidence, a forensic clone of the original storage medium is stored in a file. This forensic copy, denote as an image, can be used to perform further examinations without changing the original files (Alexander Geschonneck, 2014)*

[1]`https://www.statista.com/statistics/678739/forecast-on-connected-devices-per-person/`

In some criminal cases, such as kidnapping or terrorism, it is very important to get the evidence as quickly as possible. There are several ways to make the hard drive data visible. The first and fastest way is to start the seized computer. But this would change data and make it reduce the probative value as evidence. Another way of getting access to the data of the digital evidence is to mount the partitions that were created on the evidence to be read-only on the evaluation computer. That means to integrate these partitions like drives in the system of the evaluation computer. For this method the executor needs knowledge about the file system and the directory structure. To find specific data, such as: E-mails or documents, you need to know where they are stored in the directory structure. There are special programs needed to make the data visible. Another method is the virtualization of the evidence. To do this first a virtualization software is needed. Then a suitable configuration must be put together to start the ”image” of the evidence within the virtualization software (Michael Kofler and Ralf Spenneberg, 2012).

The basic question that arises from the problem, is that there is a method that solves these problems. First, I will investigate if such a method exists. If there is no indication that such a method exists, I have decided to create such a software program myself. This self-created program should use only, if feasible, open source tools provided or supported by the operating system for further processing. Due to licensing reasons, no commercial products should be used. Nevertheless, the whole functionality should be covered.

And now the following questions arises:

Main Research Question 1:
Can a person, unskilled in IT forensics (such as an investigator) virtualize an image from seized PCs with the required hardware and software and without professional help?

Research Question 2:
Can I find a commercial or noncommercial way or a program that will be able to do the virtualization with a previously defined test series from PC images?

Research Question 3:
When developing my own virtualization method, are there any special features of the hardware to consider when virtualizing images?

1.4 Construction of the Master Thesis

- **Chapter 1:** The introduction discusses the target group of this master thesis. Digital forensics, motivation and background to this work will be explained. In addition, the questions that arise in this work are shown.

- **Chapter 2:** To understand the master thesis, some background information is provided here. An introduction to the historical evolution of data storage, its rapid evolution, and the massive expansion in memory size is presented.

  It is said that an image is an exact copy of electronic evidence.

  The ability to virtualize with an operating system and open source software image. File system and partitions are discussed. The Basic Input Output System (BIOS) system and the system requirements of the analysis PC are explained.

- **Chapter 3:** The methodology of the study design is presented. The structure of the research is explained and this prepares what implementation of my new method. This provides the basis from my new software were also source code all there is provided in the appendix.

- **Chapter 4:** The results of my systematic group study and mine as well as the results of this master thesis are presented in text form. The ”virtualization of electronic evidence” and the storage of case-related data found during the search on the electronic evidence will be explained.

- **Chapter 5:** In this chapter, various results are discussed. Are the expectations resulting from the research questions fulfilled?
  The results are compared with the research questions and the resulting conclusions drawn. Some possible further developments are presented, which could be integrated into this master thesis.

- **Appendix:** The used commands of Operating System (OS) Linux, additional configurations and Linux Users and Groups are presented. A script for creating a virtual Universal Serial Bus (USB) stick and the API documentation for my self-programmed Python program is also listed.

Chapter 2

Theoretical Foundations

The following chapter presents relevant background theories to shed light on the research topic. To prepare for this research, I did literature research to gain inside information on the research topic. The following chapter presents relevant background topics such as Forensic, Data storage and structure, and BIOS system.
For this, I used the institutional repositories of the universities, such as the search engine Oria of the NTNU University Library. I also used the resources of the World Wide Web (WWW). In the following, I have listed some discoveries on this topic. Most relevant sources are referenced throughout this thesis and are summarized in the bibliography 5.4.

2.1 Forensic

The term forensics or digital forensics was already slanted in Chapter 1.2 Forensic.
In order to create an image of a seized PC, it is very important to work according to forensic requirements like ”chain of custody”, ”volatility”, ”evidence integrity” just to name a few. Otherwise, the evidence is useless in court.
Therefore, it is important to adhere to the general forensic methodology.

2.1.1 ”Live forensics” - ”Post-mortem forensic”

In general, IT forensics can be classified in two groups, ”live forensics” and ”post-mortem forensics” with regard to the time of the investigation.
In the case of ”live forensics” or ”online forensics”, the investigation already begins during the term of the incident. An attempt is made mainly to secure ”volatile data”. Among other things, this data may include the main memory content. The main memory content contains amongst other things information about existing network connections and started processes.
The ”post-mortem analysis” or ”offline forensics” helps clarify the incident later. This is basically done by examining disk images for nonvolatile tracks. Data on mass storage devices can be lost because as a result of the aging of the medium. Sectors can be destroyed.
The main focus is on the extraction and investigation of deleted, renamed and otherwise hidden and encrypted files from mass storage.

Due to the immense probative value of electronic evidence in a forensic investigation, this section is intended to provide an overview of how the evidence-safe preparation of a data carrier image is described. In literature, this process is often referred to as forensic duplication. Forensic duplication should be rigorously made in a forensic investigation, with a few well-founded exceptions.
In a criminal case, an electronic proof is provided, such as a personal computer. As part of a forensic duplication an image of the hard disk of the affected system is created (Bundesamt für Sicherheit in der Informationstechnik, BSI, 2011).

If the ”live forensics” is used, we definitely need an IT specialist (IT forensic expert). In ”Post-mortem analysis”, the backup can be created by an investigator, but the analysis of the image can only be done by an IT specialist later on.
During this research, described here, the post-mortem analysis is applied. After an image has been created, it is now virtualized.

2.1.2 Image as backup of the electronic reference

A proof is a 100% copy of the data store of the proof. It is a copy of the file system and contains files, file structures, file system information, etc. The boot sector is part of the information. It is usually the first sector of bootable media. This sector is the first sector of a bootable medium. It contains information about the file system and, if available, the operating system that should be started. It may also contain a program that is required to start an operating system or should be executed before the start. If the boot sector exists, it is possible to use the image to restore the system (Robert Love, Stephen Figgins, Ellen Siever, Arnold Robbins, 2009).
These data are very important in this research to enable this virtualization.
The downside of an image compared to a physical system is that when creating the image, the system is just an image of the actual resources on the disk. It is not the whole physical system. It lacks information about the contents of the main memory. Thus, all applications or similar must be restarted when the image is virtualized as a backup on the evaluation computer (the host) (Falk Gaentzsch and Prof. Norbert Pohlman, 2015).
Although the main memory data are important for post-mortem analysis in forensics, this data is beyond the scope of this research project.

2.1.3 Physical or Logical acquisition

In law enforcement, the content from data storages has to be copied to be used for analyses or as evidence instead of the original ones. This has to be done in a proper way with valid methods, so that these images can be used as evidence in the court of law. To do this, the acquired data will usually be stored into images. Images are a container where all acquired data is stored. Such a container can store additional information for the case without altering any stored data. The most important in the forensic use of images is that the contents of the images must not be changed. With a single MD5-hash (see Hash 2.1.4) of the image, a fingerprint was set which can be used to prove that not one bit of the image has been changed after the acquisition was finished.
A physical acquisition from a data storage is a 100% copy that contains all data that the original

data storage had. This includes all files, even deleted files, partition tables and boot records. These files are also needed to visualize an OS that is stored in an image. From physical images deleted files can be restored and recovered.
A logical image can be used to acquire a full data storage when deleted files are not needed or if only selected files should be acquired. Logical images can be mounted to analyze their contents. Virtualizing from stored OS would not be possible without the missing partition and boot files.

Physical acquisition

A physical acquisition is a complete backup of the existing hard drive. The copy is a bit by bit (bitstream) copy of all data and stored in a file or divided in files. The image contains the identical data set as the original, the copy can also be used instead of the original data storage (Marjie T. Britz, Ph.D., 2013).
In order to perform this type of acquisition, physical access to the data storage is required.
A physical acquisition is the basis to perform the virtualization of a seized PC.

Logical acquisition

A logical copy is only possible with a tool that can interpret the file system of the drive to be backed up. The tool recognizes the used file system or an encrypted file system and can make the data structure visible. Directory structures and their contents are thus made available and can be copied logically. Not only user-specific files can be copied. Depending on the rights assigned to the user, all files, including system files, can be copied.

2.1.4 Image creation

For image creation, the market offers a lot of software solutions, from a simple command to a software package, containing various analysis options.
There are commercial software or even non-commercial software, such as open source products. Some commercial software are Forensic Toolkit (FTK), EnCase or X-Ways Forensics, just to name a few. Free software would be FTK Imager, The Sleuth Kit or the DD Command, originally written for UNIX, but now also available for other operating systems.
In explanation, we will go into the dd command. The dd command is a simple, powerful commando that was originally used as a tool to copy in a Unix environment. There are two ways to use the command in a forensic way. First, the dd command can be used for logical copies. It can copy a datastructur or single files. The second way is the physical copy that means an exact duplication of data storage or partitions for the preservation of evidence. Dd manages to duplicate almost every block device bit by bit and is available for almost every operating system.

The following parameters call the command:

”dd if = <source>of = <destination>”

The parameter *"if="* specifies the device to be duplicated. This can be a file, an entire disk, just a partition or respectively another block device. The parameter *"of="* specifies the destination where the source data should be duplicated. Here you can specify either a target disk, a target partition, a file or again another block device. There are also other kinds of drives possible.

For example, if the investigator wants to copy the entire first hard disk bitwise, the command might look like this:

"dd if=/dev/hda of=/media/xxxxxx_forensic_images/xxxxxx_image-hda.dd"

An uncomplicated none commercial software solution for creating images is FTK Imagers. In the forensic environment, complex commercial software solutions with a variety of functions, including image creation, can be EnCase, FTK or X-Ways Forensics, to name a few.

Forensic images are always created with a resulting hash value.
When image and evidence are hashed and the value of both is the same then the content of both are identical.
Most software solutions generate a hash value from the evidence before creating an image, and after creating a hash value from the image. If these two values are the same, the contents are identical as well.

Hash or hash values are used in many areas, such as data encryption, but especially in forensics, to ensure that files were not changed. In this way it can be proven in court that a particular file or images are unchanged. Hash values are numbers that derive from the contents of files and meet certain criteria. Any changes to the file contents will also change the hash value. In the forensic, a hash is used for identification, verification, and authentication of data files, it is a form of a checksum.
Because the hash value of a file is unique at a given time, it is considered to be the same as a fingerprint.

One of the key activities that will be performed during an investigation is the generation of a cryptographic hash or hash of electronic evidence. A cryptographic hash takes an arbitrary amount of data as input and returns a fixed-size string as output.
The resulting value is a hash of the data. Common hash algorithms that are applied during a forensic investigation are MD5, SHA1 and SHA256 hashes.

If the dd command is used, it is possible to hash the evidence first e.g. with the md5sum command and then the image with the same command.

```
md5sum /dev/hda
$ 4b779c46d1c2c5cb68bc14ad4c462146 /dev/sda
```

and
(after *"dd if=/dev/hda of=/media/xxxxxx_forensic_images/xxxxxx_image-hda.dd"*)

```
md5sum xxxxxx_image-hda.dd
$ 4b779c46d1c2c5cb68bc14ad4c462146 xxxxxx_image-hda.dd
```

Since the values are the same, the contents are the same. (Cory Altheide and Harlan Carvey, 2011)

2.2 Data storage and structure

2.2.1 Data storage

The rapid increase in storage capacity is described here. It should make clear what storage capacities an IT forensic expert is dealing with. The long processing time in IT forensics results from these large capacities of data storages and this is one of the reasons for this research work.

In the late 19th century, punch cards were used for the first time as digital storage. The most famous mission in which the cards were used was the American census in 1890 (Sebastian Helmer, 2012).
The first hard drive, (5 Mega Byte (MB)), was invented by International Business Machines Corporation (IBM) (at the request of the US Air Force) and introduced in early 1956. The journey continued over the 3.5-inch floppy disk (360Kilo Byte (KB) to 1440 KB) over the Compact Disc Read-Only Memory (CD-ROM) in the early 1980s, the ZIP drive in 1994, to the first USB stick (64 MB) in 1996. The journey continued in the same way over the Secure Digital Memory Card (SD) (SanDisk 8 MB) in 2001, the DVD (4.7 Giga Byte (GB)-17.08 GB) appeared as an alternative on the digital storage market in 2001. Followed by the Blu-ray Disc in 2006 with a storage capacity up to 200GB (from TDK).
The journey is now in the present, a hard drive, the first Helium-Filled MG07ACA HDD from Toshiba (Anton Shilov, 2017) with 14 TB was developed in 2017.
The history of the Solid-State-Drive (SSD) dates back to the 1970s, when flash memory was still in its infancy. In 1985, the first SSD was installed in an IBM PC. In 2001, a 3.5" flash memory SSD with 14 GB was sold. In the mid-2000s, SSDs were installed in notebooks by default (Sven Schefer, 2011). And recently on March 20th, 2018, the largest SSD in the world was presented with 100 Tera Byte (TB) (futurezone, 2018).

This description of digital storage mediums from the beginning to the present day, should show how rapidly the development precedes. From a tape with storage capacity of 80 bytes to a hard disk with storage capacity of 100 TB (which equals 109.951.162.777.600 bytes) in just 128 years. This list does not describe all storage media developed in the past, but it gives the most important elements and illustrates the storage capacities digital forensics deal with today. While the data storage has changed rapidly, the way of getting hands on legal or illegal data has changed as well. In the past, data could only be transported via data carriers from one PC to another. However, the Internet also introduced the intranet, the in-house network. These networks make it easy to send data from one place to another.

2.2.2 Filesystem

What is a file system? What does a file system manage? What abstractions does a file system provide for the rest of the operating system?

The main purpose of computers is to create, edit, store and retrieve data. A file system provides the methods to support these tasks. And all this happens on a permanent storage medium, such as a hard disk. And therefore the file system is also a very important part of this thesis because it is needed to perform virtualization of a PC system. File systems manage persistent storage and form an integral part of all operating systems. There are many different approaches and methods for managing persistent storage. That's why there are a lot of file systems that have different focuses.
For very specific tasks, specially developed file systems are used. For example, for a CD-ROM, the ”ISO 9660” has been developed, ”FAT12” for floppy disk. For some devices, such as the flash memory, even several file systems have been developed, depending on the structure of the data carrier and intended use. There is e.g. ”ETFS”, ”F2FS”, ”Yaffs” for NAND flash; ”JFFS”, ”SPIFFS” for NOR flash memory and so on (Brian Carrier, 2005).

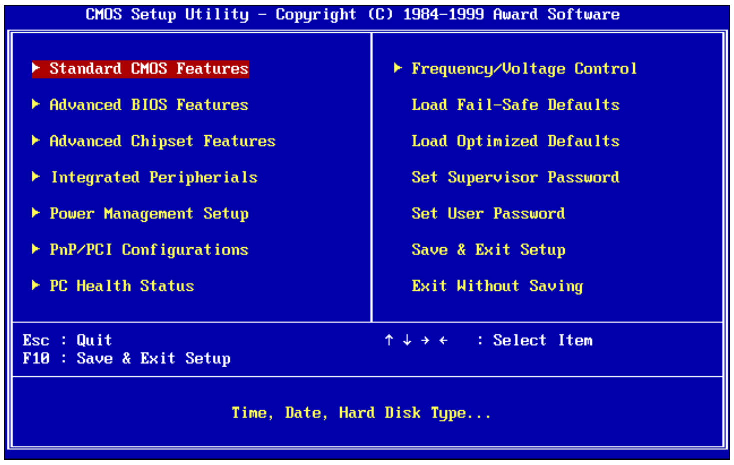
2.2.3 Partition

Computers need to provide access to the information stored on a hard disk internally or externally. Each hard drive can store millions of bits. How does a computer now ”know” where to find the required information? To simplify this search, hard disks are divided into individual, identifiable areas. A disk is a sequence of consecutive sectors for the operating system. The division of a data carrier in sectors is made during the low-level formatting. These sectors are thus the smallest logical data storage unit to be addressed by the operating system. In a partition table partitions are (at least one) registered. A partition is a collection of consecutive sectors in which a file system resides.

2.3 BIOS System

When you start the computer, the Basic Input Output System (BIOS) performs startup tasks to discover and start the existing hardware. During this process, partition boot code is loaded and executed by the designated boot device (eg, a hard disk). This is the first stage of the bootstrap loader.
The boot-up task is a small program that may load and start a more complex bootloader code into random access memory. This more complex code, such as GRand Unified Bootloader (GRAND) or LInux LOader (LILO), are bootloaders that provide a user with multiple boot capabilities to launch different operating systems or different versions of the same operating system. But there are other options, such as different operating system load options or standalone programs (eg, games) that can be run without an operating system.
Whether or not a more complex code is loaded, after the first or second stage of the bootloader, a kernel is loaded that handles the further communication, commands, and execution.

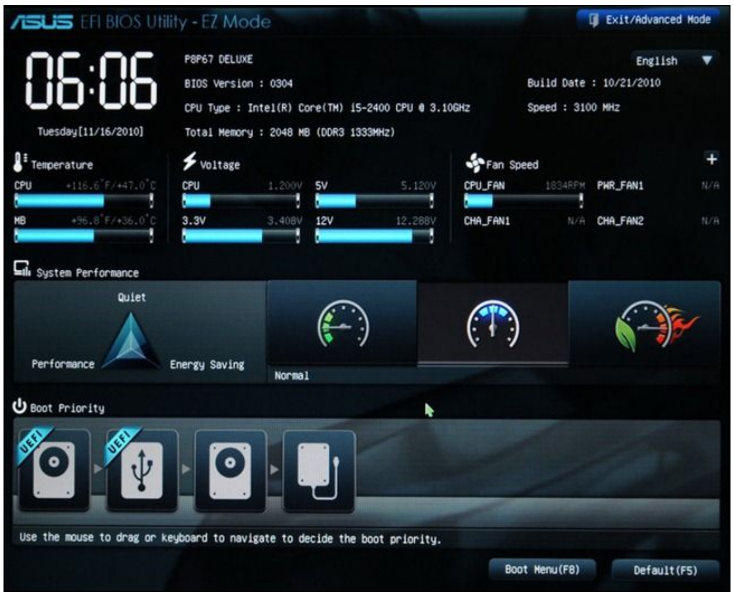
The original BIOS, see Figure 2.1, was launched in 1981. Since the market introduction, the BIOS has always been developed further. In 2006, the Unified Extensible Firmware Interface (UEFI) was introduced, which was delivered in 2011 for the first time with a PC. In legacy computers and in the latest computers, these BIOS systems still exist to this day (Janez Puhan, 2015).



```
CMOS Setup Utility - Copyright (C) 1984-1999 Award Software

▸ Standard CMOS Features           ▸ Frequency/Voltage Control
▸ Advanced BIOS Features             Load Fail-Safe Defaults
▸ Advanced Chipset Features          Load Optimized Defaults
▸ Integrated Peripherials            Set Supervisor Password
▸ Power Management Setup             Set User Password
▸ PnP/PCI Configurations             Save & Exit Setup
▸ PC Health Status                   Exit Without Saving

Esc : Quit                           ↑ ↓ → ←   : Select Item
F10 : Save & Exit Setup

                  Time, Date, Hard Disk Type...
```

Figure 2.1: The original BIOS [1]

The first visible difference is that UEFI has a graphical user interface with mouse support and Internet access. See Figure 2.2. One of the main reasons for the introduction of UEFI is the 64 bit architecture support. Because most new PCs and operating systems are 64-bit capable and now fully support 64-bit.



```
06:06                P8P67 DELUXE
Tuesday[11/16/2010]  BIOS Version : 0304                                Build Date : 10/21/2010
                     CPU Type : Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz  Speed : 3100 MHz
                     Total Memory : 2048 MB (DDR3 1333MHz)

Temperature                  Voltage                          Fan Speed
CPU  +116.6°F/+47.0°C        CPU   1.200V   5V    5.120V      CPU_FAN  1834RPM   PWR_FAN1  N/A
MB   +96.8°F/+36.0°C         3.3V  3.408V   12V  12.288V      CHA_FAN1 N/A       CHA_FAN2  N/A

Use the mouse to drag or keyboard to navigate to decide the boot priority.
```

Figure 2.2: The UEFI BIOS [2]

Other reasons for the introduction include the administration of larger hard disks and a larger number of partitions. While the original BIOS only managed 2.2 TB hard disks, the GUID

[1]https://www.howtogeek.com/56958/htg-explains-how-uefi-will-replace-the-bios/

[2]https://www.howtogeek.com/56958/htg-explains-how-uefi-will-replace-the-bios/

Partition Table (GPT) partitioning scheme allows UEFI to address up to 18 exabytes. The original BIOS can have one extended partition and unlimited logical partitions per hard disk. UEFI works according to the GPT standard, and should therefore support at least 128 partitions.

There are even more reasons to think of UEFI, but not all of them can be listed here.
In order to support the operating system of the analysis PC UEFI for QEMU, a UEFI package has to be installed. For this we take the package Open Virtual Machine Firmware (OVMF), which is an EDK-II-based project. It enables UEFI support for virtual machines. OVMF includes sample UEFI firmware for QEMU and KVM (Janez Puhan, 2015).

2.4 Linux operating system

Images of PCs can be virtualized with any common OS. There are OS as well as virtualization software, that can be downloaded free of charge from the Internet. But it was decided for Linux as the OS for the evaluation computer. In this chapter, some background information is discussed why Linux was selected as the operating system. The history of Linux is presented and the distribution we used discussed.

2.4.1 Linux history and distributions

A young Finnish computer science student named Linus Benedict Torvalds (born on December 28, 1969 in Helsinki, Finland) developed the LINUX operating system in 1991 as a student of the University of Helsinki.
Until today there are several Linux distributions on the market.
A Linux distribution is an operating system consisting of a software collection based on the Linux kernel (by Linus Torvald) and often a package management system (Marcus Fischer, 2010).

During the preparation of the research, 436 active Linux distributions were counted on the website ”`https://archiveos.org/linux/`”.
This number varies daily, if you follow this website.

Currently, no official information about the number of different Linux distributions can be found on the market.
The confusing selection and ever increasing number of Linux distributions (”distros”) can be irritating for Linux novices.
There are a lot of other distributions that are better tailored than others to a specific purpose. However, the following distributors are the most popular and offer active forums and mailing lists that users can get help from in case of problems and questions.
Linux Mint and Ubuntu have the reputation of being the easiest for new users who want to be productive with Linux as quickly as possible. These users do not claim to master the distributions in all their complexity. At the other end of the spectrum, Arch Linux has more advanced distributions that require a lot of learning to do their effective work. OpenSUSE, Fedora and Debian GNU / Linux mark a good middle ground. CentOS is an Enterprise Distributon, suitable

for all those who want stability, reliability and long-term support instead of the latest features and brand new software (elysium, 3. Januar 2017).

The next two statistics show the 10 most popular Linux distributions on the market in 2016. Staistik Figure 2.3 shows the breakdown of distributions in the market. In the statistic Figure 2.4 the popularity of the over the last years until 2016 can be seen:

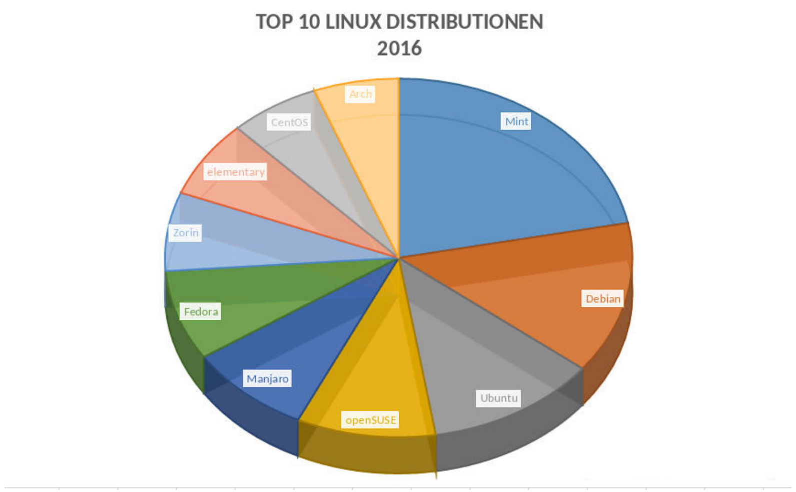


Figure 2.3: The 10 most popular distributions in 2016, proportionately [3]

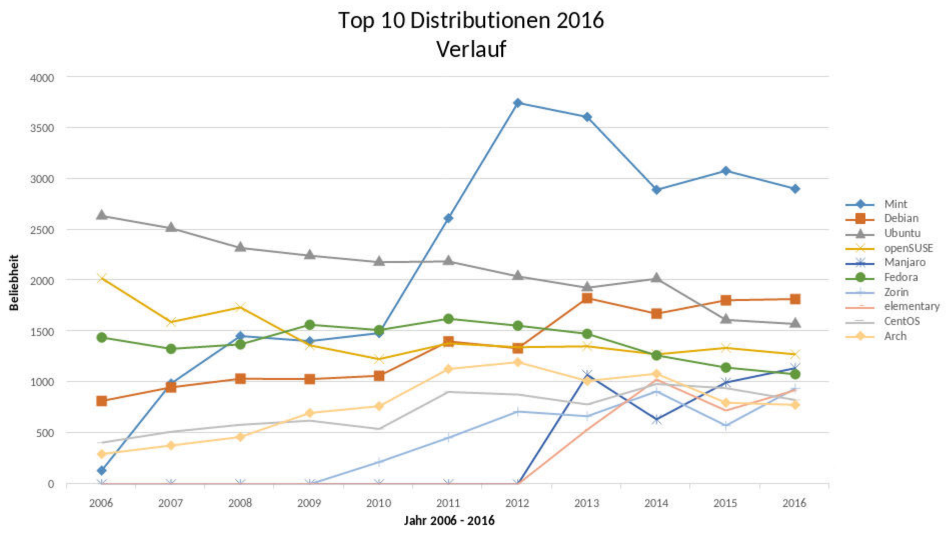


Figure 2.4: The 10 most popular distributions 2016, development [4]

For this research I decided to use the Linux Mint distribution, as I will explain later.

[3]https://games4linux.de/der-grosse-linux-distribution-report-die-10-beliebtesten-distributionen-2016/

[4]https://games4linux.de/der-grosse-linux-distribution-report-die-10-beliebtesten-distributionen-2016/

2.4.2 Dealing with Filesystems

A Linux operating system can integrate and support the most popular file systems from home. This means that supported file systems always have read access to all data. Apart from exceptions, in most cases it is also possible to have write access to the embedded file system. As a rule, entire partitions are automatically included during the boot process. For this purpose, the system to be integrated must have a corresponding construction manual. This construction manual defines which partitions should be hung in which directories. Such a construction manual can be found in the file *"/etc/fstab"*. This file may contain additional information about the partitions or drives that should not be automatically mounted, such as removable disks or CD-ROM drives. (F. Kalhammer, 2001).
Under Linux there is a way to query all types of file systems, which are basically supported by the kernel of the Linux system. If the necessary software packages are installed, you can call the menu item *"File systems"* with the command "make menuconfig". Then all the supported file systems and their properties will be displayed.
Various file systems are automatically supported by Mint, just to name a few: The Extends 4, File system in USErspace (FUSE), Reiserfs, JFS, XFS, GFS and so on.
There are also submenus such as *"CD-ROM/DVD Filesystems"*, *"DOS/FAT/NT Filesystems"* or *"Pseudo Filesystems"* in which the support of their file system is described in detail.

Not always can a file system be fully integrated into a Linux system. There may be limitations, e.g. at the NTFS of Mirosoft. By default, an NTFS file system can be included in reading. The NTFS writing is not really possible! The limitations of writing are extremely high.
Therefore, the file system "NTFS-3g" [5] was developed by Tuxera Inc., Finland [6]. This file system is based on the FUSE. If the software package "NTFS-3g" is installed, there should be no more restrictions (Tuxera, 2017).

There are various filesystems that can be installed like a program. There are also pseudo file systems, such as sysfs, usbfs or devpts. All these file systems contain only purely virtual files with information or devices mapped to a file.
Some file system that are based on the principle of FUSE can be integrated into a Linux operating system, for example:

- fuseiso ⇒ for integrating ISO files (Ubuntu Documentation, 2009),
- sshfs ⇒ to connect a remote file system via ssh (secure shell, a cryptographic network protocol for operating network services over an unsecured network). Here it does not matter which remote file system the foreign computer uses, because the administration of the remote file system takes place at the remote computer (Jack Wallen, 2017).
- ...

The variety of different file systems and the simplicity of installing new and different file systems is one of the reasons why I have chosen Linux as the operating system for the evaluation PC. Another reason is the FUSE file system, which is the basis of many virtualization programs.

[5] `https://wiki.ubuntuusers.de/Windows-Partitionen_einbinden/NTFS-3G/`
[6] `https://www.tuxera.com/`

2.4.3 Linux Mint

All Linux distribution based on the Linux kernel of Linus Torvalds. Linux Mint is based on the Ubuntu distribution. Linux Mint was developed by the Frenchman Clement Lefebvre. In 2006 he published his first version.
But Linux Mint is not just an Ubuntu system with a number of new programs and updated desktop themes. The developers have emphasized that through a variety of graphical ”mint tools” the usability is improved (DistroWatch, 2018; Linux Mint, 2017).

This includes

- The Mint Desktop, an utility to configure the desktop environment.
- The MintMenu, a new and elegant menu structure for easier navigation.
- The MintInstall, an easy to use software administration.
- The MintUpdate, a software updater.

To list only the most important among the various tools and hundreds of additional improvements.
It has the ability to install it directly from the live session on the hard drive. In computer science, the term live system or direct start system refers to an operating system that can be started in a live session without installation. By default, the contents of the mass storage devices in the system (such as hard disks or SSDs) are not affected or altered. Mint developed his own design to stay true to the line of easy usability (Sphinx, 2017).

Advantages:

The tools are developed in-house and the amount of user-friendly adjustments can be described as excellent. The inclusion of multimedia codecs and the ability to accept suggestions from their user community is rewarded by the users (DistroWatch, 2018; Linux Mint, 2017).

Disadvantages:

The alternative ”Community” editions do not always contain the latest features. The project does not publish safety instructions (DistroWatch, 2018; Linux Mint, 2017).

Software Package Management:

APT with mintInstall using DEB packages (compatible with the Ubuntu repositories) (DistroWatch, 2018; Linux Mint, 2017).

Available editions:

One main edition (with MATE and Cinnamon), ”minor” editions (with KDE and Xfce), Linux Mint ”Debian” edition (rolling-release with MATE or Xfce) (DistroWatch, 2018; Linux Mint, 2017).

Conclusion:

Because Mint is easy to install, very easy to use and equipped with various programs that are needed for this research, Linux Mint has been chosen as OS for the analysis PC of this thesis.

2.5 Virtualization

Virtualization is used today in many fields. The use increases daily, because among other things operating costs are saved and administrators can do their work from the home office (Marcus Fischer, 2010). As described in Chapter 1.1, a piece of evidence should be virtualized as quickly as possible. During my research I couldn’t find a product that automatically starts the virtualization of an image after a simple creation of a configuration file. There are a lot of programs around the topic of virtualization. But finding the right one for this thesis was the challenge (David Wolski, 2015, 2016).
Virtualization has a number of common applications, all designed to make their technology an abstraction of physical resources. In computer science, virtualization refers to the simulation of a hardware or software object. Virtualization can create virtual devices or services such as emulated hardware, operating systems or data storage. This allows computer resources, such as running one operating system inside another, to be provided (Matthew Portnoy, 2012).
If the expression virtualization is used, the terms emulation and simulation are often used in conjunction. These terms may be similar but not the same. Falsely, the terms are often used interchangeably. The concepts of virtualization, emulation and simulation, however, are very different technical approaches, each with different goals and areas of application (David Wolski, 2014).

For the remainder of this thesis I defined virtualization, emulation and simulation as follows:

Definition 2 (Virtualization)**.** *Virtual machines that are provided by a virtualization environment should simulate or emulate as little as possible. They should pass hardware access directly to the actual system components such as processor, graphics card or hard disk. These hardware accesses are then only managed by the virtualization environment. In order to ensure the stable operation of a virtual machine, system components such as graphics cards or certain chipsets are also emulated (David Wolski, 2014).*

Definition 3 (Emulation)**.** *An emulation imitates the appearance of a system so that programs that want to access the system can find compatible software and hardware interfaces. The aim of emulation is to simulate only the defined, visible behavior. Internally, however, the behavior in the system is quite different. Today, emulators are often used as an example in the programming* *of hardware. The emulator simulates the behavior of the hardware on the newly programmed software (David Wolski, 2014).*

Definition 4 (Simulation)**.** *In a simulation, an external complete system with its hardware and software is simulated. This means that a simulation of such a system usually recreates a closed environment. No direct hardware access is possible from this simulation. All system components were simulated using the software. In contrast to emulation, a simulation not only simulates externalities, but also the detailed internal logic (David Wolski, 2014).*
Simulations are used, among other things, to simulate a processor platform on another system and to develop operating systems or individual applications. As an example today older computer systems are emulated to develop new software for the old system or to change and adapt older software.

In order to have virtualization, we often rely on commercial tools. At the beginning of my research I found the program Live-View (Chapter 2.5.1), which can automatically build configuration files for VM goods from an image. Live-View is free of charge, but VM-Ware is a commercial software and did not correspond to my idea of building a solution with non-commercial software.

For the virtualization of seized PCs, I was looking for a parameterizable program. This was necessary in order to be able to go into the different hardware technical contents of the created image. Because the image comes from different computers, the hardware configuration is always different. And only through a parametrisable virtualization program can you adjust this. During my research I came across the program Quick EMUlator (QEMU) again and again. QEMU is a generic and open source machine emulator and virtualizer, which I will discuss in more detail in the next chapter.

Even VirtualBox, a free virtualization program from Oracle, uses some virtual QEMU hardware devices and has an integrated dynamic recompiler based on QEMU. VirtualBox also offers the possibility to be administrated and configured from the command line. For this, the command-line interface ”VBoxManage” is provided. VBoxManage supports all the features that the graphical user interface gives you access to. But this program offers a lot more. It provides properties that the graphical interface does not offer, such as Control advanced and experimental configuration settings for a Virtual Machine (VM). But first a configuration file in Extensible Markup Language (XML) format must be created, with which then VirtualBox can be started. From my point of view, QEMU is a bit easier to control, because I just call a parameterized command. No configuration file is required at this time. If an error occurs when starting a VM with a parameterized command, a parameter in the command can simply be changed. The command can be restarted without first changing a configuration file. In my opinion, this seems to be the easiest way.

QEMU supports virtualization when e.g. the Kernel-based Virtual Machine (KVM) kernel module is used. KVM is a virtualization solution for the Linux kernel. KVM is a part of the Linux kernel since version 2.6.20 (Feb. 2007). Thus, the circle closes why I have selected Linux as the operating system. Linux with KVM and QEMU create the solid and free (open source) basis for a stable virtualization of computer systems. Because this variant can also be parameterized, all that is needed now is a configuration tool to compile the image-specific configurations and start QEMU.

2.5.1 Live-View

Under Windows e.g. the variant ”Live View” [7] can be used. Live View is a graphical forensics tool that creates the configuration files for a VMware virtual machine from a raw dd image. This VMware virtual machine is booted in a secure environment and can then be analyzed. Changes are written to a separate file, leaving the image unchanged. To start the virtual machine, the program VMware Workstation [8] is required. In contrast to other virtualization systems, where a configuration file has to be created by hand before the virtualization system can be started with the configuration file, Live-View is free to create configuration files. This is done by searching the image for specific factors (Harlan Carvey, 2009). It can boot full disk raw images, bootable partition raw images, physical disks (attached via a USB or Firewire bridge), and specialized and closed image formats (using 3rd party image mounting software).
The version ”Live View 0.7b” contains the following operating systems:

- Windows 2008, Vista, 2003, XP, 2000, NT, Me, 98
- Linux (limited support)

With the ”LiveView 0.8 RC1” version, Windows 10 systems can also be used with live forensics.

The following programs are example software of a commercial or non-commercial nature, which are not included in the operating system:

- Bochs [9](Runs on Windows, Linux, and Xbox, is free and open source)
- Parallels [10](Runs on Mac platform, Windows and Linux)
- Microsoft Virtual PC [11](Runs on Windows, DOS and OS/2)
- Virtual Iron [12](Run on Windows and Linux)
- Win4Lin [13](Runs on Linux)

2.5.2 QEMU

QEMU is short for Quick EMUlator. QEMU is a free, and an open-source software (GNU General Public License, version 2.) for virtualizing (using KVM) and emulating hardware (Johannes Plötner and Steffen Wendzel, 2012). QEMU is made in Japan. It is a hosted hypervisor

[7]`http://liveview.sourceforge.net/`
[8]`https://www.vmware.com/products/workstation-pro.html?build=1744117`
[9]`http://bochs.sourceforge.net/`
[10]`www.parallels.com`
[11]`www.microsoft.com/windows/products/winfamily/virtualpc/default.mspx`
[12]`www.virtualiron.com/`
[13]`http://win4lin.net/content/`

that creates a virtual computer hardware platform, which describes what it does.
It emulates CPUs, translating one instruction set into another using binary machine code. Virtualization with KVM only works if the hardware supports virtualization (for example: Intel-VT or AMD-V), otherwise the system will not be emulated. This method is well known as dynamic binary translation.
QEMU emulates systems with the following processor architectures: x86, x64, PowerPC (32- and 64-bit), ARM (32- and 64-bit), Alpha, CRIS, LatticeMico32, m68k and Coldfire, MicroBlaze, MIPS, Moxie, SH -4, S / 390, Sparc32 / 64, TriCore, OpenRISC, Unicore and Xtensa. The QEMU virtual machines emulate all devices needed to run a VM Guest. It supports, for example, several types of network cards, block devices (hard and removable drives), USB devices, character devices (serial and parallel ports), or multimedia devices (graphic and sound cards).

Among other things, QEMU supports overlay images (delta images). This overlay image is based on an existing image file. This overlay image saves all changes to the original image file. The original image will not be changed (Thomas Ritzau and Robert Warnke, 23. Mär. 2010).

The following image formats are supported from QEMU: raw (dd), qcow2, bochs, cloop, cow, dmg, nbd, parallels, qcow, qed, vdi, vhdx, vmdk and vvfat (QEMU, 2018).
QEMU also offers a number of device models in addition to the emulation of CPUs.
These make it possible to run operating systems on the emulated hardware without modifying the operating systems. QEMU is capable of emulating hardware and running different hardware-specific operating systems on this emulated hardware (Thomas Ritzau and Robert Warnke, 23. Mär. 2010).
In addition, open source programs have the advantage that they are constantly being developed by many developers. Thererfore errors are quickly recognized and remedied. In addition, QEMU is KVM's predestined virtualization solution from RED HAT [14](developed by Qumranet, Qumranet was purchased by Red Hat in September 2008). These develop e.g. the Virtio drivers which continue to ensure even more compatibility.

2.5.3 Requirements

During my research, I found various managing tools that can manage virtual machines. But I could not find a tool to compile configurations. That is the reason why I decided to program this configuration tool myself.

Following, I show what dependencies and requirements the research for such a virtualization and the self-programmed configuration tool "Virtual PC" have resulted in. Various programs must be installed. Following, I show which programs. I show which configuration files need to be adapted. The user groups that must be created are displayed. And you have to add the logged in user to these groups.
Now the installed system is searched for the required programs.

[14]https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/virtualization_getting_started_guide/chap-virtualization_getting_started-products

The *"dpkg -l"* command displays almost all installed programs. If this command is changed to, *"dpkg -l | grep pmount"* then within the listed programs the program "pmount" is searched for.

If the program is installed the output looks like this:

```
$ dpkg -l | grep pmount
    ii pmount 0.9.23-3build1 amd64 mount removable devices as normal user
```

If the program is not installed, it returns only the Promt:

```
    $ dpkg -l | grep pumount
    $
```

It is also possible to string the commands together as follows:

```
    "dpkg -l | grep 'pmount | xmount | parted'"
```

With this command we have now queried which programs have been installed with this distribution and which programs may be missing. All missing programs needed for this work can now be installed.
The following programs / packages should be installed:

- pmount
- xmount
- fusermount
- parted
- qemu
- ovmf

Standard commands like:

- cp
- rm
- mkdir
- grep
- mktemp
- lsblk
- blkid
- losetup

should have been installed with the Mint distribution.

Now some configuration changes have to be made with administrator rights.
If files are not available, they can be created as user root.

The *"/etc/pmount.allow"* is adjusted by adding the following entries:

```
/dev/loop[0-9]
/dev/sd[b-z]
/dev/sd[b-z] [0-9]
```

The *"/etc/pmount.conf"* is adjusted by adding the following entries:

```
loop_allow = yes
loop_devices = /dev/loop0, /dev/loop1, /dev/loop2
```

The *"/etc/fuse.conf"* will be adjusted by adding the following entries:

```
user_allow_other
```

Finally, group affiliations of the user have to be adjusted. The executing user should belong to the following groups:

- disk
- fuse
- kvm

If one or more groups are not available, they can be created with the command

"groupadd <group>"

Chapter 3

Methodology

In the previous chapters, there was an introduction to the topic and the problem that is to be solved with this thesis and the research questions. The different research questions involve the answer to individual methods. In order to answer the research questions, this chapter explains the research methodology. This chapter explains the choice of research approach, the research design. This is followed by a discussion on the possibility of achieving usable results. It concludes with a brief discussion of the problems and limitations arising from the research methodology. Furthermore, the problems that have arisen during the research are pointed out.

3.1 Research Design

This thesis will use quantitative and qualitative research strategies. To answer the research questions I have done a literature research with keywords in the previous chapter. I searched first for a way how images that include an OS can be started in a virtual environment. The used virtualization system should be able to virtualize different types of images with different contents. These are e.g. Images created from seized PCs with different operating systems. Such as Windows, Apple, Linux or other Unix operating systems. It should also work with different computer architectures, such as Intel or AMD. If possible the images should be started automatically or with a minimum as possible1 of needed action from the investigator. For this a program or configuration tool is needed to include the hardware driver from the analyses machine e.g. for the graphic card. The required data for the configuration of the virtualization must be noticed already during the seizure.

To find a virtualization system that works in this way I started the first part of my empirical study work. Experimental research tries to find out how a certain procedure influences the result. I tested different virtualization tools to find a system that could work in the wanted way. I programmed a configuration tool myself because I did not find one that could do this work. The work was done with try and error until it worked in the wanted way. I think that the first part was the qualitative research strategy because I had no measurable results like values or numbers. Otherwise, it can also attend to a quantitative research strategy under the line that working or not working can be seen as two values like 0 and 1. The experimental part of this work will be described in Chapter 4.2.1.

But can an unskilled investigator really work with the created environment? This can not be answered in a theoretical way but in an experimental part. To prove this, I tested the same environment that I used for the experimental work that I have done myself. I have tested this with 10 unskilled investigators described in Chapter 4.2.2. This part uses a quantitative research where the results can be shown in percent or simply by counting. How many investigators have been able to work with the experimental environment.

3.2 Research Methods

3.2.1 Literature Research

To be able to answer the research questions, I started the literature research. I have used the following search engine for literature search: the NTNU discovery tool/search engine Oria, the Specialist Search Engines of the University College Dublin and Google Scholar. To search for literature using these search engines, I searched with the following keywords: virtualization, forensic virtualization, images, mount, mount partitions, ”unskilled person” and ”unskilled person” in IT forensics. In order to extend my literature research, I searched for additional terms such as command-line virtualization, window virtualization, Linux virtualization, Virtualbox, QEMU. And I found interesting information that would help me find a solution to the problem. One result of the search was QEMU, an open source software for virtualization that can be started from the command line. Further, the search revealed that the missing gap could be closed with a self-developed program written in the Python programming language. At this point, I focused my literature research to fix problems that occurred during programming and tested my script with a Laptop that had a Linux OS.

During the literature research, I searched for something that can virtualize an operating system. Something that visualizes an image of a seized PC so easily that perhaps ”unskilled persons” can work with it. It should be a simply usable combination of soft- and hardware. That means that the used images only had to be connected to an analysis computer and as many as possible were detected automatically.
At this point I have not found any commercial or non-commercial hard or software that was able to do this. The idea was born to use open source tools and a laptop with Linux OS and a self-written program for the practical implementation. The literature research was expended to find solutions for parts or the whole task. The information found in the literature research was used and tested step by step. During these steps, I also tested methods that were unsuccessful and had to be discarded at the end.
Focusing on the successful methods found in literature research I was able to virtualize some images with a laptop that uses Linux as its operating system. As intended, only open source tools were used. To realize an environment that could be used by an unskilled person, I started to program my own tool. It took a long time until the program worked in the way I expected. To get the answer if unskilled persons were able to work with my tested environment, that’s should be tested with probands.

3.2.2 Empirical Study

Experiments are used in science as an aid to try them out under real conditions. (Moy, T., and Tyrkus, M. J. (Eds.), 2016).

I started my empirical study with a Windows 10 operating system. To mount images windows programs have been used. Further experiments were done with Linux derivatives. Linux operating systems offer several methods to mount partitions from images. For this, it is needed to define the partitions from first to the last sector. For the first part of my empirical studys, VirtualBox and QEMU were found. Both can be started from the command line. For the second part, saving the commands, I had to create my own application.

Among other things, I choose Python as the programming language because it was part of the study. The most important reason is that Python is a standard in the Python forensic community (Preston Miller and Chapin Bryce, 2016). The first problem was to create a useful GUI that could also be operated from unskilled persons. I had not found any libraries that could portray the frontend that I wanted. Looking for tools that can fix this problem for Python I did not find a solution, I had to develop some GUI elements myself. During programming, it turned out that the required system commands did not exist or had not worked properly.

To use the needed commands I implemented Linux commands in a system call of the script that Python only had to execute. This had the consequence that the program that I named ”Virtual PC” is only executable on Linux OS. After testing this script successfully with both tools, QEMU was the only tool that was able to store results to an overlay file. I decided to use QEMU and the self written-program that should work in the desired way. During programming, images were used that contained Windows, Linux and Mac operating systems. The used Mac images could not be virtualized in a simple way that is needed for this solution. Further tests with Mac images were discarded because it didn't seem feasible in the time that was available for this master thesis. After I had tested my program successfully with the described environment, I was able to go to part 2 of my empirical study.

To know if unskilled persons were able to work with the same environment using my self-written program a second experimental part was done. 10 unskilled investigators were used as probands. After being instructed in the use of the self-written configuration tool, they should be able to connect and virtualize 2 Windows and one Linux image.

3.2.3 New Method Design and Validation

After all a new method to virtualize images in a fast and simple way has arisen. To validate this method different images containing Windows and Linux OS have been used and tested in the same environment. First errors and problems e.g. with different computer architectures, such as Intel or AMD, have been fixed. To validate the results every image has been tested more than twice. At least 10 unskilled investigators have tested this new method successfully.

Chapter 4

Experiment and Results

4.1 Experiment Setup

This section describes the experiment setup with the focus on intelligibility.

4.1.1 Hardware

To run the virtualization, I used a Laptop as an analysis PC.
The Analysis PC is a standalone machine with the following specifications Table 4.1.

Component:	Description
Description:	Laptop
Vendor:	LENOVO
Version:	Lenovo G50-70
Width:	64 bits
Size:	2682MHz
Capacity:	3100MHz
Processor:	Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz
Disk:	2TB ST2000LM003 HN-M
BIOS:	BIOS boot specification is supported
UEFI:	UEFI is supported

Table 4.1: Hardware Configuration Analysis PC

4.1.2 Software

One of the goals of this research project is to use free and open source tools to virtualize image that contain a operating system. The used software and version you can see in (Table 4.2)

Type:	Name and version
OS Distributor ID:	LinuxMint
OS Release:	19
OS Codename:	Tara
QEMU emulator:	version 2.11.1
Python:	3.6.5
pmount:	0.9.23-3build1
xmount:	0.7.3-1build2
fusermount:	version: 2.9.7
parted:	3.2-20

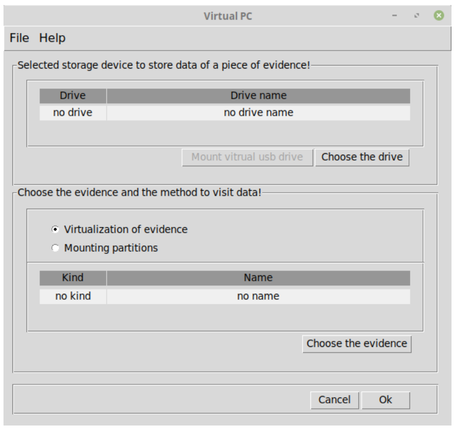
Table 4.2: Software Configuration Analysis PC

4.1.3 Implemented Proof of Concept

One problem in the preparation of the research was to determine how the results could be verified. Due to the lack of comparable work, I decided to program a configuration tool for the found virtualization software. A tool in which a configuration can be compiled for each specific case, which can also be saved. A program that combines all the necessary commands in the correct order and with a Graphical User Interfacem (GUI). This would give the investigator the chance to access data from an electronic evidence in very little time. The tool should then also be able to save the assembled configurations for the evidence and reopen them later for another proof of evidence. Likewise, it should be possible for the investigator to save found traces on a storage medium.

The next step was to determine in which programming language the tool should be realized. During my studies in Norway at PHS and NTNU, I got in touch with some programming languages/script languages and operating systems. I decided to use the operating system Linux and the platform-independent scripting language Python. The primary reason for choosing the programming language is that Python is a standard in the Python forensic community.
After the Python program has been implemented and all previous questions have been answered, 1369 lines of source code have been created.

Now I briefly introduce the GUI of the Python program ”Virtual PC”.
The Figure 4.1 is the starting mask:



Virtual PC

File Help

Selected storage device to store data of a piece of evidence!

Drive	Drive name
no drive	no drive name

Mount vitrual usb drive | Choose the drive

Choose the evidence and the method to visit data!

● Virtualization of evidence
○ Mounting partitions

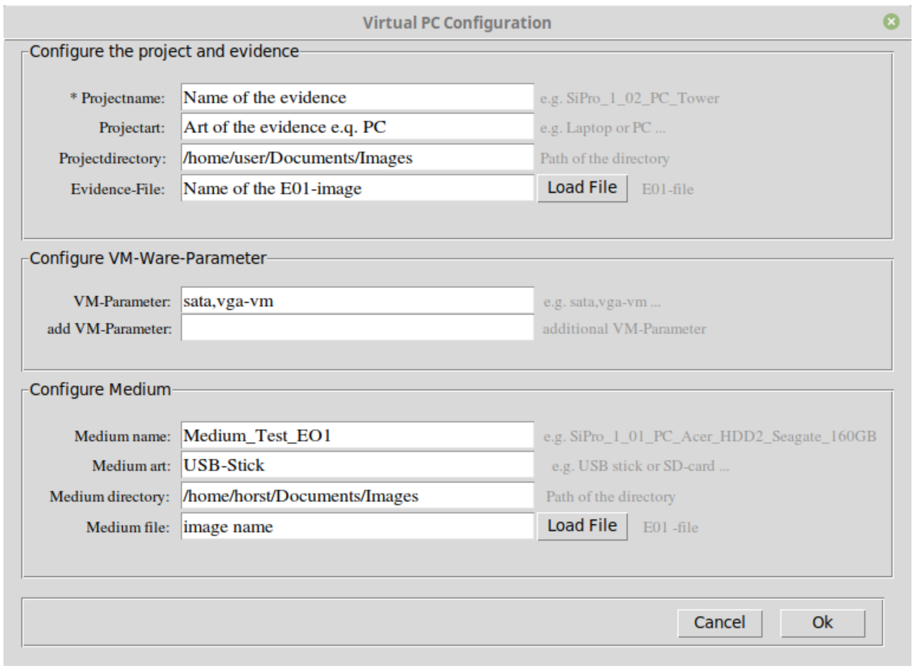
Kind	Name
no kind	no name

Choose the evidence

Cancel | Ok

Figure 4.1: Virtual PC start dialog

The Figure 4.2 is the Configuration Dialog:



Virtual PC Configuration

Configure the project and evidence

* Projectname:	Name of the evidence	e.g. SiPro_1_02_PC_Tower
Projectart:	Art of the evidence e.q. PC	e.g. Laptop or PC ...
Projectdirectory:	/home/user/Documents/Images	Path of the directory
Evidence-File:	Name of the E01-image	Load File E01-file

Configure VM-Ware-Parameter

VM-Parameter:	sata,vga-vm	e.g. sata,vga-vm ...
add VM-Parameter:		additional VM-Parameter

Configure Medium

Medium name:	Medium_Test_EO1	e.g. SiPro_1_01_PC_Acer_HDD2_Seagate_160GB
Medium art:	USB-Stick	e.g. USB stick or SD-card ...
Medium directory:	/home/horst/Documents/Images	Path of the directory
Medium file:	image name	Load File E01 -file

Cancel | Ok

Figure 4.2: Virtual PC configuration dialog

The complete API Documentation of the program ”Virtual PC”can be found in the Appendix C.

4.2 Experiment Description and Results

This chapter describes the results of the solution approach, which are the research questions formulated in Chapter 1.3. First, the methods for creating the experiment are described both my empirical study and those of my systematic group study. Within these descriptions, the results of the various experiments are described and associated with the research questions. The

detailed description for anyone interested in, reviewing or even reproducing my results can be found in the appendix.
For the sake of completeness, the term experiment, which has already been mentioned several times before, should be defined in this chapter. An experiment is an investigation and a method for data collection in which purposefully formulated research questions should be checked for their validity. It is a very good result of an experiment if the answers to the formulated research questions can be clearly doubted or confirmed.

4.2.1 My First Empirical Study

The first attempts at virtualization were made with a Windows 10 system. With Windows' own program ”Mount-DiskImage” it is possible, to mount an image (virtual hard disk or ISO) as a partition (Microsoft, N/A), by using the Powershell. With a Windows 10 system, a read-only mount of a partition is possible with the tool ”diskpart”. As an administrator you can set a partition to read-only mode with the command ”attributes volume set read-only” after diskpart has been executed (Admin, Top Password Software, Inc, 2018).
Further attempts have been made with Linux derivatives. With Linux there are several methods to mount partitions. But to mount partitions from an image, the partitions firstly must be defined, to find the first and the last sector. For this purpose ”losetup” (Chapter A.1.12) was used. After a partition has been detected, this partition can be mounted with ”pmount” (Chapter A.1.8). Each user can mount with ”pmount” partitions, even without administrator rights.
The focus is to give an investigator access to data from a data storage from an evidence as quickly as possible. During my research, the solution seemed to lie in finding a virtualization software that could be executed with a parameterizable command. This would increase the speed of execution, as these commands are easy to change. These commands are also easy to save and manage.
As described in Chapter 2.5, I came across two programs, VirtualBox and QEMU that fit into my goal, free or open source. And both can be started from a command line. Although VirtualBox can be completely parameterized and configured from the command line, it can not be started via a parameterized application start. Therefore VirtualBox is not an option for this research.
QEMU is the right choice as described in Chapter 2.5.2, because QEMU can work with overlay files. The changes during the work with the original image are stored separately in an overlay file. That ensures that the original image was not changed in any way. This ensures that we work forensically. QEMU is also the right choice in a second point, because QEMU can be started parameterizably from the command line.
I started my empirical study with my evaluation laptop (Chapter 4.1.1) with installing the operating system Linux Mint 19 (Chapter 4.1.2). And next, all other necessary measures (see Chapter A), which resulted from the literature research, were carried out. All required applications (Chapter A.1) that were not automatically installed with the operating system were then installed. This is followed by the customization of the configuration files (Chapter A.2). Finally, missing Linux users and groups were created (Chapter A.3).
After the hard- and software for the analysis PC were assembled, I created some images from test computers with different operating systems. Starting with Windows XP, Windows 7, Windows 10, Linux Mint 19 and Mac El Capitan (10.11) operating systems.
I was not able to virtualize this Mac OS. Fixing the occurred bugs involved, a lot of time was

these because it required the manipulation of system files of the operating system in the virtual machine. In my research of these problems, I found out that a suitable way to virtualize an image from a Mac computer only works with a Mac computer.
All other methods to virtualize the Mac image would not be so feasible that a non-skilled person would be able to work with it. Another point is that the Mac computer is very rare as working equipment in the German police. Only some specialists have a Mac computer. All other investigators have a computer with a windows operation system like I have assembled and used for the virtualization. This is the reason why I have discarded to virtualize a Mac operating system. This may be a point for further work.

Encrypted images or images with encrypted partitions have not yet been considered. Furthermore, password-protected data or operating systems have not been considered. During my work on this research, on some Windows 7 installations the graphic card driver had to be changed in the configuration of the program ”Virtual PC” to run with the specified Windos 7 installation. An error with the number 0x0000007b is a special error that has occurred on my Windows7 system. This error may have occurred on Windows 7 and later systems, including Windows 10. If the hard disk controller in Windows is disabled or faulty, this problem appears. Here the registry must be changed. There are various hints on the internet. For me, the easiest way was to start the image in the virtual environment with a LiveCD [1] in the CD-ROM drive. The program ”Dr.Web Registry Editor ”can then be started from the CD-ROM. With Dr.Web Registry Editor values [2] can be changed in the registry, which are responsible for the misconduct (Arne Arnold, 2015).
At the path
”HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Msahci”
set the entry ”Start” to the value 0. For the second path in the registry
”HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\IastorV”,
the entry ”Start” must also be set to 0.
This solution applies to a Windows 7 or Vista operating system. For higher Windows versions, there are other solutions on the Internet. There were also problems with some XP installations during my tests. Here the SATA driver had to be reset.

While I was working on virtualization, I could see some weaknesses in how it works. A weak point is the missing saving of parametrised start commands. If an error occurs when starting a virtualization, the virtualization does not start and must be aborted. Now the start command has to be revised and rewritten. If a virtualization needs to be restarted from evidence at a later time, the start command must also be reparameterized and rewritten. This is a point that emerges from the development and test phase that this start command should be storable and manageable.
During my research, I could not find a way to manage or save these startup commands. Therefore, there is no way to easily customize these startup commands until virtualization is bug-free and executable. That was the deciding factor to design and program such a configuration tool myself and to create the software configuration tool ”Virtual PC”. The program ”Virtual PC” can be seen in the Chapter 4.1.3, the working method in the Chapter 4.2.3. This program was so developed that an unskilled person is able to work with the given environment. The source code for this created program can be found in the appendix of my research.

[1] `https://free.drweb-av.de/`
[2] `http://www.pc-experience.de/wbb2/thread.php?threadid=31929`

4.2.2 My Second Systematic Group Study

The experimental structure of my systematic group study is exactly the same as in my empirical study (Chapter 4.2.1). In my systematic group study, I selected 10 probands, scientific subjects, that corresponded to my definition of an ”unskilled person in IT forensics”. The term ”unskilled person in IT forensics” is explained in Chapter 1.2. In simple words this means that a person who is not a trained IT-Specialist is an unskilled person.
For the experiment, 10 policemen between 25 and 40 years were used as test subjects. There were 9 male probands and 1 female proband. Overall, they were trained investigators and had experience in their discipline. The computer is part of their permanent work tool.

At the beginning of the experiment, each subject had basic IT skills. This means that each subject can operate his work programs as a user. For this research project, the test persons were instructed in the use of the configuration tool ”Virtual PC” developed in the thesis research. In addition, they were introduced to the basic knowledge of IT forensics (Bundesamt für Sicherheit in der Informationstechnik, BSI, 2011). For this purpose, the guideline ”IT-Forensik” of the German Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik (BSI)) was used.
I have prepared three images for the test. I wanted to use the most common operating systems. This was done with regard to the order of use of Windows, Mac and Linux, as shown in Figure 4.3. As explained before, I have discarded, using an image from a Mac operation system because I was not able to virtualize a Mac operation system in a suitable way. Windows was clearly the most used operating system. This was the reason why I used two Windows and one Linux image for the test.

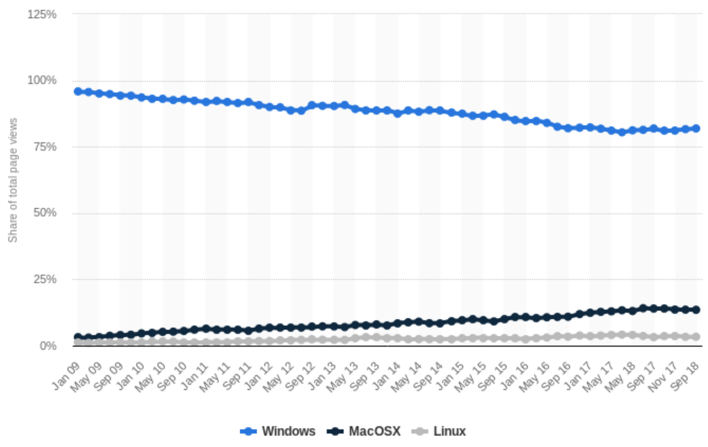


Figure 4.3: Market share of the leading operating system editions in Germany from January 2009 to May 2018

The following specifications document the three test machines from which the test images have been created. The two Windows PC are displayed in Table 4.3 and Table 4.4, and the one Linux Laptop in Table 4.5 respectively. The computers are standalone machines.
In this experiment, I first introduced the probands to the principle of this study. I explained, that we want to virtualize the delivered image. The probands received an image previously acquired

Component:	Description
Description:	Computer
Vendor:	self-construction
Width:	64 bits
RAM:	64 GByte
Processor:	Intel (R) Xeon (R) CPU E5 v3 @ 2.60 GHz
Disk:	Samsung SSD 850 PRO 256GB
BIOS:	BIOS boot specification is supported
UEFI:	UEFI is supported
OS Distributor ID:	Microsoft Windows 10 Professional

Table 4.3: Configuration Windows 10 PC Image

Component:	Description
Description:	Computer
Vendor:	self-construction
Width:	64 bits
RAM:	16 GByte
Processor:	Intel (R) Core (TM) i7-3770 CPU @ 3.40 GHz
Disk:	STOSHIBA DT01ACA300 SCSI Disk Drive
BIOS:	BIOS boot specification is supported
UEFI:	UEFI is supported
OS Distributor ID:	Microsoft Windows 7 Professional (Service Pack 1)

Table 4.4: Configuration Windows 7 PC Image

Component:	Description
Description:	Laptop
Vendor:	LENOVO
Version:	Lenovo G50-70
Width:	64 bits
RAM:	4 GByte
Processor:	Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz
Disk:	2TB ST2000LM003 HN-M
BIOS:	BIOS boot specification is supported
UEFI:	UEFI is supported
OS Distributor ID:	LinuxMint
OS Release:	19
OS Codename:	Tara

Table 4.5: Configuration Linux PC Image

with FTK Imager. This image met the forensic requirements and was in the E01 format [3]. Now

[3]`http://www.forensicsware.com/blog/e01-file-format.html`

the subjects should virtualize this image on the analysis PC with the tools provided and without further help.
First, the data carrier with the image of one of the test PCs should be connected to the analysis PC. In order to ensure smooth virtualization, the image of the test PCs should be copied to the analysis PC. After the program ”Virtual PC” has been started, metadata and paths can be set. Under the item ”Configure VM Parameter” one can specify parameters that are decisive for the virtualization such as ”sata” and ”vga-vm”. With OK, the previously compiled configuration is saved. In the main GUI a storage medium is selected and with a radio button the search method is selected. One can choose between virtualization and mount partitions. If the radio button is set to”Virtualization of evidence”, the virtualization is started with entering OK. If an error occurs during startup, parameters can be changed and restarted until virtualization starts without error.

After the test persons have created and saved a configuration file with the configuration tool ”Virtual PC”, the virtualization program can be started with the settings that were saved in the configuration file. Now the operating system that was stored in the used images will be started in the virtual environment.
After completion of the virtualization, the evidence data found could not be stored on a storage medium, since this possibility was not considered at the time. Therefore, these data could not be used for further investigations at that time without further effort. This should be incorporated into the program ” Virtual PC ” based on this finding.
Eight out of ten probands were able to virtualize all three images. There were small problems with two probands. They had little difficulties in parameter input because parameters were wrongly written or wrongly indicated with the Linux image. After a short briefing, the experiment went very well, and the test persons all reached their goal. In this experiment, there were no problems with the Windows image, only minor problems as described with the Linux image. Thus, of 30 possible tests, 28 were successfully completed. That is a rate of 93.33%. Two tests were unsuccessful at first, which is a quota of 6.67%. They did not make it in the first attempt. In the second attempt after a short support, they were then successful.
From the experiment, the probands suggested saving the possible input parameters in a list. This is basically possible, but then the list would have to be maintained permanently. Therefore this idea was again discarded. Another suggestion was not to virtualize any image of evidence, it would be sufficient to mount partitions. After all, experienced investigators know where certain data can be found in the file system. This idea proved to be beneficial as partitions are much faster to mount than to virtualize an image. There are several ways to mount partitions from an image, without virtualization of an operating system.
Preliminary research revealed that, for the Windows operating system, various tools are available on the market. For most tools, various settings can be made through a graphical user interface. I would like to mention only three of them here.

- Mount Image Pro [4]
- FTK Imager [5]
- OSFMount [6]

[4]http://www.mountimage.com/
[5]http://marketing.accessdata.com/ftkimager4.2.0
[6]https://www.osforensics.com/tools/mount-disk-images.html

For the operating system Linux, the program ”pmount” can be used (Chapter A.1.8). How the mount was implemented can be seen in the following Chapter 4.2.3.4.
This variant was implemented in the program ”Virtual PC”. With this implemitation the investigator can choose between virtualization and mounting partitions.
A further improvement wished by the investigators was the ability to store important investigative data or evidence. This idea turned out to be very good also was implemented by me so that an additional storage media can be integrated into virtualization and mounting environment. The data found can then be stored on these additional storage media. They will still exist when the program ”Virtual PC” has been closed. The storage media only needs to be mounted in the analysis PC after finishing ”Virtual PC” to process the data further. How this works can be seen in the following Chapter 4.2.3.5.

The last suggestion was that, if no storage medium is available, a way should also be created to store found evidence in this case. For this, an ”virtual” memory stick was created, on which data can be stored, which is still available after the program is finished. After shutting down the virtual machine this ”virtual” memory stick can be mounted and the data copied to the analysis PC. This option was also implemented in the program and is now available to the users as an additional feature. Further information on this storage of the data can be found under the Chapter 4.2.3.6.

4.2.3 Configuration tool ”Virtual PC”

4.2.3.1 Computer processor architecture

Before I explain the program ”Virtual PC” that I have programmed for this thesis, we take a look at the current system architecture. As can be seen in the statistics Figure 4.4, the most common and used in the PC field CPU architectures are ”Intel” and ”AMD”. Except for a few percent, they divide the whole market among themselves.

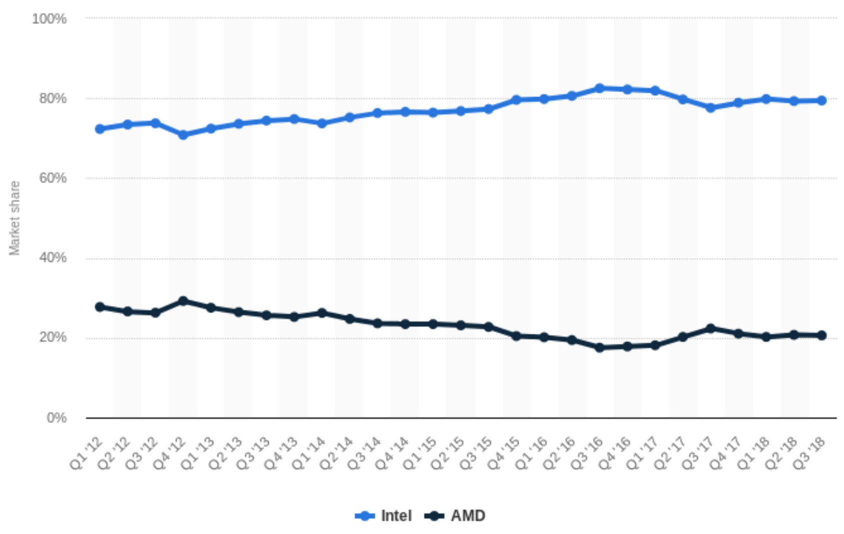
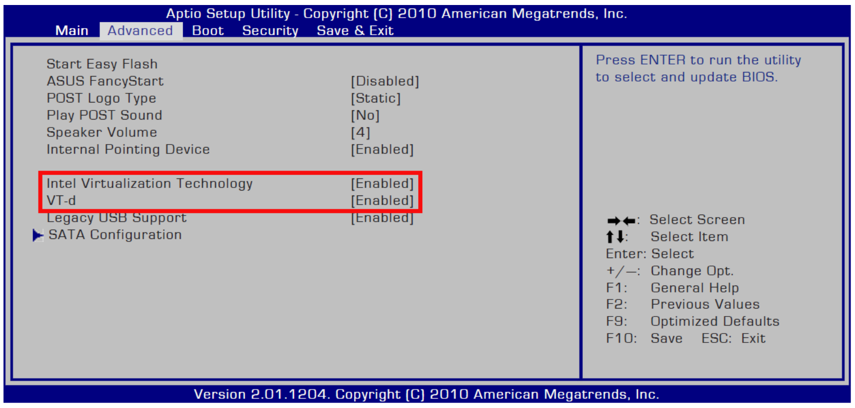


Figure 4.4: Distribution of AMD and Intel x86 computer processors worldwide, from 2012 to 2018

Basically, the program has been developed for an Intel architecture. But during my research I also considered the AMD architecture. After some changes, the program is also executable on the AMD architecture.
To check if Intel or AMD architecture is present, I used the follwing command on the analysis PC:

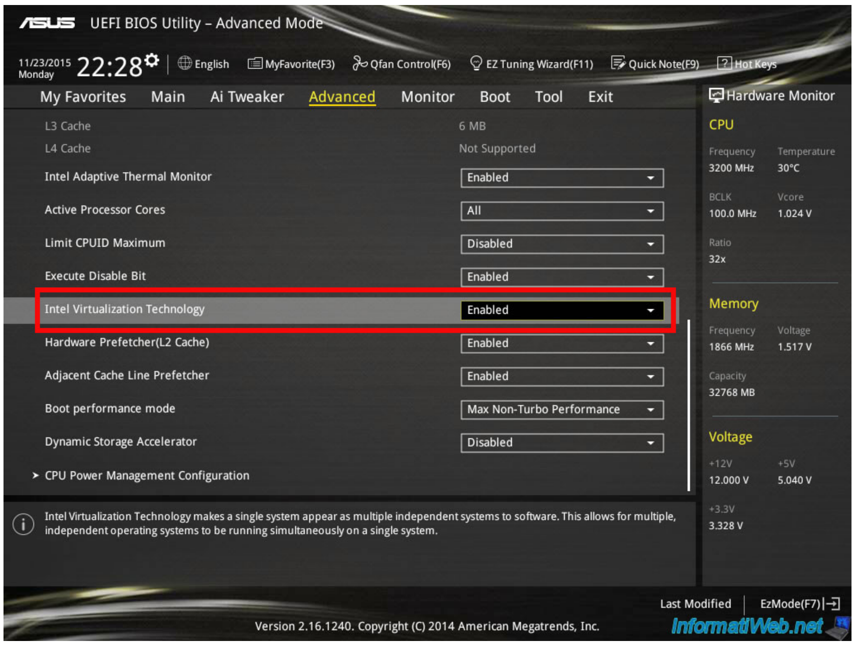
Host ∼ $ grep ”vmx” /proc/cpuinfo

If lines are output with ”vmx”, these are ”Intel VT” processors. As shown in BIOS in Figure 4.5 and in UEFI-BIOS Figure 4.6-4.7.



```
Aptio Setup Utility - Copyright (C) 2010 American Megatrends, Inc.
Main  Advanced  Boot  Security  Save & Exit

Start Easy Flash                                  Press ENTER to run the utility
ASUS FancyStart                 [Disabled]        to select and update BIOS.
POST Logo Type                  [Static]
Play POST Sound                 [No]
Speaker Volume                  [4]
Internal Pointing Device        [Enabled]

Intel Virtualization Technology [Enabled]
VT-d                            [Enabled]
Legacy USB Support              [Enabled]
▶ SATA Configuration
                                                  →←:  Select Screen
                                                  ↑↓:  Select Item
                                                  Enter: Select
                                                  +/—: Change Opt.
                                                  F1:  General Help
                                                  F2:  Previous Values
                                                  F9:  Optimized Defaults
                                                  F10: Save   ESC: Exit

Version 2.01.1204. Copyright (C) 2010 American Megatrends, Inc.
```

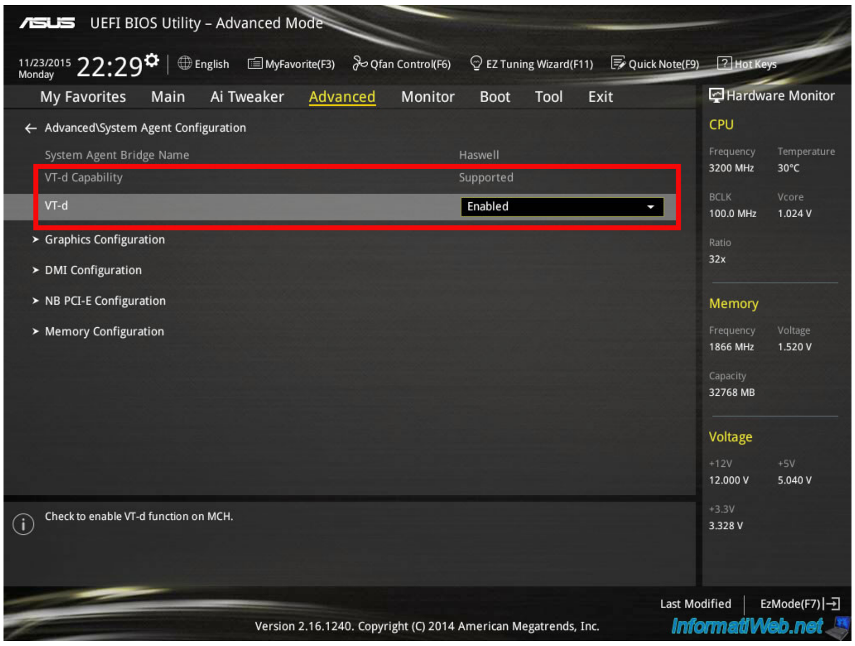
Figure 4.5: Intel VT Parameter in the original BIOS [7]



```
UEFI BIOS Utility – Advanced Mode
11/23/2015 Monday 22:28   English  MyFavorite(F3)  Qfan Control(F6)  EZ Tuning Wizard(F11)  Quick Note(F9)  Hot Keys

My Favorites  Main  Ai Tweaker  Advanced  Monitor  Boot  Tool  Exit        Hardware Monitor

L3 Cache                                  6 MB                         CPU
L4 Cache                                  Not Supported                Frequency   Temperature
Intel Adaptive Thermal Monitor            Enabled                      3200 MHz    30°C
Active Processor Cores                    All                          BCLK        Vcore
Limit CPUID Maximum                       Disabled                     100.0 MHz   1.024 V
Execute Disable Bit                       Enabled                      Ratio
Intel Virtualization Technology           Enabled                      32x
Hardware Prefetcher(L2 Cache)             Enabled                      Memory
Adjacent Cache Line Prefetcher            Enabled                      Frequency   Voltage
Boot performance mode                     Max Non-Turbo Performance    1866 MHz    1.517 V
Dynamic Storage Accelerator               Disabled                     Capacity
> CPU Power Management Configuration                                   32768 MB
                                                                       Voltage
Intel Virtualization Technology makes a single system appear as        +12V        +5V
multiple independent systems to software. This allows for multiple,    12.000 V    5.040 V
independent operating systems to be running simultaneously on a        +3.3V
single system.                                                         3.328 V

                                              Last Modified | EzMode(F7)
Version 2.16.1240. Copyright (C) 2014 American Megatrends, Inc.   InformatiWeb.net
```

Figure 4.6: Intel Virtualization Technology in the UEFI-BIOS [8]

[7]https://superuser.com/questions/746440/how-to-enable-vt-x-on-asus-k53sv
[8]https://us.informatiweb.net/tutorials/it/9-bios/215--enable-iommu-or-vt-d-in-your-motherboard-bios.html


UEFI BIOS Utility – Advanced Mode

11/23/2015 Monday 22:29 | English | MyFavorite(F3) | Qfan Control(F6) | EZ Tuning Wizard(F11) | Quick Note(F9) | Hot Keys

My Favorites | Main | Ai Tweaker | Advanced | Monitor | Boot | Tool | Exit

Advanced\System Agent Configuration

```
System Agent Bridge Name                 Haswell
VT-d Capability                          Supported
VT-d                                     Enabled
> Graphics Configuration
> DMI Configuration
> NB PCI-E Configuration
> Memory Configuration
```

Check to enable VT-d function on MCH.

Hardware Monitor

```
CPU
Frequency    Temperature
3200 MHz     30°C
BCLK         Vcore
100.0 MHz    1.024 V
Ratio
32x

Memory
Frequency    Voltage
1866 MHz     1.520 V
Capacity
32768 MB

Voltage
+12V         +5V
12.000 V     5.040 V
+3.3V
3.328 V
```

Last Modified | EzMode(F7)

Version 2.16.1240. Copyright (C) 2014 American Megatrends, Inc.

InformatiWeb.net

Figure 4.7: Intel VT Parameter in the UEFI-BIOS [9]

And another way to query the architecture is:

Host ∼ $ grep "svm" /proc/cpuinfo

If lines with "svm" are output, these are "AMD-V" processors. As shown in BIOS in Figure 4.8 and in UEFI-BIOS Figure 4.9-4.10.

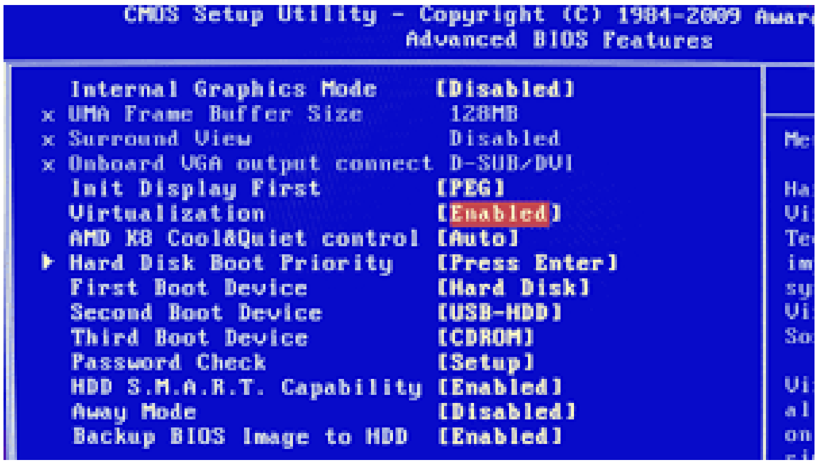
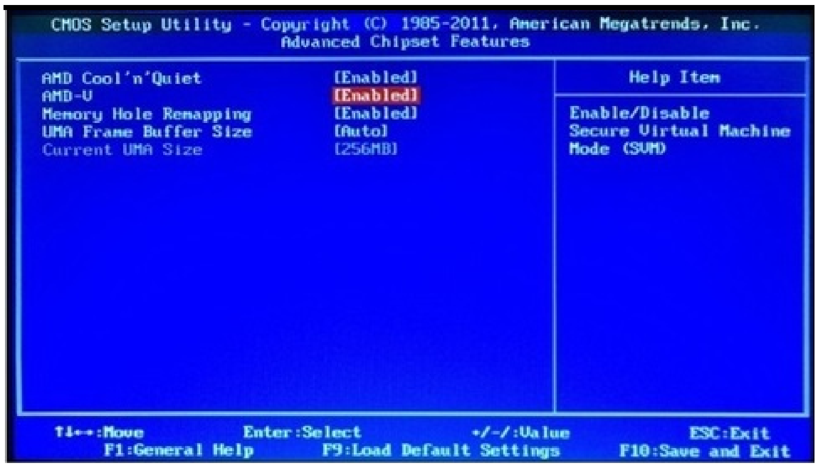

```
          CMOS Setup Utility - Copyright (C) 1984-2009 Awar
                       Advanced BIOS Features

    Internal Graphics Mode          [Disabled]
  x UMA Frame Buffer Size            128MB
  x Surround View                    Disabled
  x Onboard VGA output connect      D-SUB/DVI
    Init Display First              [PEG]
    Virtualization                  [Enabled]
    AMD K8 Cool&Quiet control       [Auto]
  ▶ Hard Disk Boot Priority         [Press Enter]
    First Boot Device               [Hard Disk]
    Second Boot Device              [USB-HDD]
    Third Boot Device               [CDROM]
    Password Check                  [Setup]
    HDD S.M.A.R.T. Capability       [Enabled]
    Away Mode                       [Disabled]
    Backup BIOS Image to HDD        [Enabled]
```

Figure 4.8: AMD Virtualisierung in the original BIOS [10]

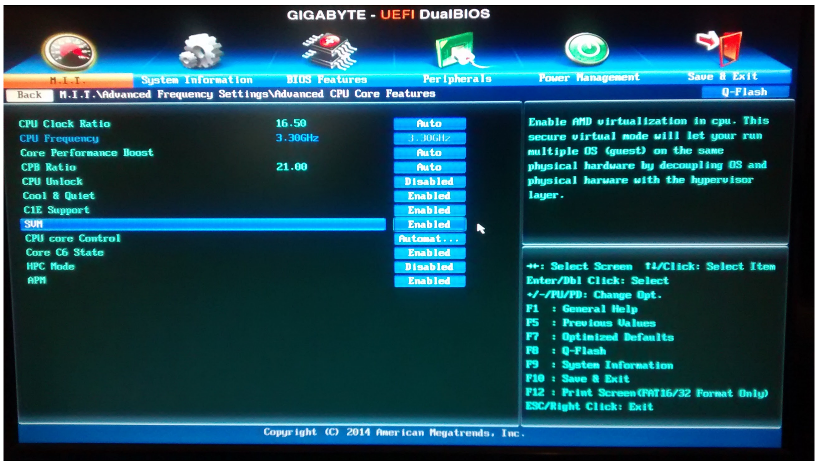
[9]https://us.informatiweb.net/tutorials/it/9-bios/215--enable-iommu-or-vt-d-in-your-motherboard-bios.html

[10]https://askubuntu.com/questions/256792/how-do-i-enable-hardware-virtualization-technology-vt-x-for-use-in-virtualbox



```
CMOS Setup Utility - Copyright (C) 1985-2011, American Megatrends, Inc.
                    Advanced Chipset Features

AMD Cool'n'Quiet            [Enabled]          Help Item
AMD-V                       [Enabled]
Memory Hole Remapping       [Enabled]     Enable/Disable
UMA Frame Buffer Size       [Auto]        Secure Virtual Machine
Current UMA Size            [256MB]       Mode (SVM)

↑↓←→:Move        Enter:Select        +/-/:Value          ESC:Exit
     F1:General Help       F9:Load Default Settings     F10:Save and Exit
```

Figure 4.9: AMD SVM-Parameter in the original BIOS [11]



```
M.I.T.\Advanced Frequency Settings\Advanced CPU Core Features

CPU Clock Ratio              16.50      Auto
CPU Frequency                3.30GHz    3.30GHz
Core Performance Boost                  Auto
CPB Ratio                    21.00      Auto
CPU Unlock                              Disabled
Cool & Quiet                            Enabled
C1E Support                             Enabled
SVM                                     Enabled
CPU core Control                        Automat...
Core C6 State                           Enabled
HPC Mode                                Disabled
APM                                     Enabled

Enable AMD virtualization in cpu. This
secure virtual mode will let your run
multiple OS (guest) on the same
physical hardware by decoupling OS and
physical harware with the hypervisor
layer.

→←: Select Screen  ↑↓/Click: Select Item
Enter/Dbl Click: Select
+/-/PU/PD: Change Opt.
F1  : General Help
F5  : Previous Values
F7  : Optimized Defaults
F8  : Q-Flash
F9  : System Information
F10 : Save & Exit
F12 : Print Screen(FAT16/32 Format Only)
ESC/Right Click: Exit

Copyright (C) 2014 American Megatrends, Inc.
```

Figure 4.10: AMD SVM Parameter in the UEFI-BIOS [12]

If there is no output, hardware virtualization is not supported, QEMU can only be installed without KVM acceleration.
Since virtualization with QEMU works best on a 64-bit system, a 64-bit Linux should also be installed. Before installation, the system can be queried via a Mint Live CD (Linux Mint operating system without installation).

Host ∼ $ grep "lm" /proc/cpuinfo

If lines with "lm" are output on the command, a 64-bit Linux can be installed, because the CPU is able to work with 64-bit architecture.

Now there is also the possibility to optimize the system with more RAM. Thus, the virtual machines can be provided with more memory. Optionally, then activate the parameter "Kernel

[11] https://www.itworld.com/article/2981515/virtualization/virtualbox-diagnose-and-fix-vt-xamd-v-hardware-acceleration-errors.html

[12] https://unix.stackexchange.com/questions/283212/how-to-enable-iommu-on-gigabyte-ga-970-gaming-sli-cf

Samepage Merging (KSM)", to greatly reduce the overall memory consumption under certain circumstances. To find out if the kernel supports KSM, enter the following command.

```
Host ~ $ grep KSM/boot/config-name -r‘
CONFIG_KSM = y
```

If the parameter *"CONFIG_KSM"* is output with "y", the KSM support is guaranteed. Whether the KSM support is also activated is queried as follows.

```
Host ~ $ cat /sys/kernel/mm/ksm/run
1
```

If the number "1" is output, the KSM support is activated. If a "0" is output, KSM support is not activated. The parameter can be adjusted in the file *"/etc/default/qemu-kvm"* (Thomas Ritzau and Robert Warnke, 23. Mär. 2010).

4.2.3.2 System requirements

For running the program *"virtual_pc.py"*, the analysis PC can be based on an Intel architecture and an AMD architecture, as described in Chapter 4.2.3.1.
For virtualization, the processor must also include the functions (- Intel->VT-x, - AMD->AMD-V). These additional functions are needed to pass commands to the CPU from the (guest) operating system directly to the CPU, which increases the speed extremely, because the commands do not need to be converted or edited.
In the BIOS/UEFI of the analysis PC, if the virtualization specified parameters for an Intel architecture are selectable, they must be activated.

- Intel Virtualization Technology
- VT-d Capability
- VT-d

to enable virtualization.
For an AMD architecture, the following settings have to be made, if they are selectable.

- Virtualistion
- SVM

The system requirements for a virtualization PC are similar to the LinuxMint system requirements, and for optimal performance, the operations system to be virtualized still needs to be considered.

The system requirements for LinuxMint 19 are shown in Table 4.6 (Clem, 2018):

• Processor:	32-bit PAE-capable x86 processor or 64-bit x86 processor (Linux Mint 64-bit requires a 64-bit processor) Linux Mint 32-bit works on both 32-bit PAE-capable processors and 64-bit processors processors)
• RAM:	1 GB of RAM (2 GB recommended for convenient use)
• HDD size:	15 GB of storage space (20 GB recommended)
• Grafical card:	Graphics card with a resolution of 1024 768 pixels (At lower resolutions: You can move windows with Alt + pressed left mouse button)
• Drive:	DVD-ROM drive or USB port

Table 4.6: The LinuxMint 19 system requirement

If this system is a Windows system, the system requirements are shown in Table 4.7 (Dirk Makowski, 2018).

System requirements:	Description
Windows XP:	PC with 300 MHz microprocessor (Intel Pentium / Celeron product family, AMD K6 / Athlon / Duron product family), 128 MiByte RAM, 1.5 GiByte free memory, Super VGA monitor (800x600), CD-ROM or DVD -ROM drive, keyboard and mouse
Windows Vista:	1 GHz processor, 32-bit (x86) or 64-bit (x64), 1 GiByte system memory, 40 GB hard disk with at least 15 GiByte of available space, DirectX 9 graphics (WDDM driver, 128 MiByte graphics memory, pixel memory). Shader 2.0, hardware-based, 32 bits per pixel), DVD-ROM drive, audio output
Windows 7:	1 GHz or faster processor, 32-bit (x86) or 64-bit (x64), 1 GiByte RAM (32-bit) / 2 GiByte RAM (64-bit), 16 GiByte (32-bit) / 20 GiByte (64-bit) free space, DirectX 9 graphics processor with driver for WDDM 1.0
Windows 8:	1 GHz processor or faster, 1 GiByte RAM (32-bit) or 2 GiByte (64-bit), 16 GiByte (32-bit) or 20 GiByte (64-bit) free hard disk space, DirectX 9 graphics card or with WDDM driver

Table 4.7: The Windows system requirements, part1

System requirements:	Description
Windows 10:	1 GHz or 1GHz processor for 32-bit or 2 GB for 64-bit, 16GB for 32-bit operating system or 20 GB for 64-bit operating system, DirectX 9 or higher with WDDM 1.0 driver, 800 x 600

Table 4.8: The Windows system requirements, part2

The system requirements of the analysis PC and the system to be virtualized must be added together for a smooth running virtualization.
In Table 4.9 is an example, as host system a Linux Mint 19 (64-bit) operation system and as guest system a Windows 10 (64-bit) operation system.

	Linux Mint 19 - Host	Windows 10 (64-bit) guest	**For the VM**	Statement
CPU	32-bit PAE x86 / 64-bit x86	1 GHz	> **1 Ghz 64 bit**	Since Windows 10 1 Ghz is needed and 64bit, at least one CPU with 1Ghz and 64 bit is needed.
RAM	2 GB	2 GB	**4 GB**	The host needs 2 GB of RAM and the guest also needs 2 GB of RAM. Since each system requires a certain amount of memory, 4 GB (2 + 2 GB) is needed
Graphic card	1024x768	800x600	**1024x768**	The resolution of the host is only important here, because the guest can emulate a resolution that is larger than that of the host.
Harddrive	20 GB	20 GB	> **40 GB**	Here it is like the RAM. Each system needs hard disk space so this must be added.

Table 4.9: System requirements added together

If the operating system for the analysis PC is Linux Mint 19, the following software package has already been installed:

- losetup
- parted
- fusermount
- Python 3 with TKinter (as GUI Framework for Python)

- NTFS-3G.

The following software packages still have to be installed:

- xmount
- pmount
- QEMU
- OVMF (UEFI).

Now the group customizations and configuration changes:

- Create user group FUSE.
- Include the current USER in the groups ”disk”, ”fuse”, and ”kvm”.
- Change the configuration files ”pmount.allow”, ”pmount.config” and ”fuse.config”.

Finally, copy the software package ”Virtual_PC” to the Analysis PC.
At this point, the Analysis PC should be fully installed and configured.

4.2.3.3 The way to virtualization

In this section, I describe the path from an image to its virtualization.
The police in Germany mainly produce an image file in E01 format (Encase Image File Format). That was the reason why we assumed an E01 format. So before QEMU can now be used, the evidence image still needs to be converted. For this purpose ”xmount” (explained in Chapter A.1.10) is used. Xmount converts the E01 image into a RAM/dd format and creates an overlay file <imagename>.ovl.
As previously described, xmount allows to use multiple types of input and output hard drives. Xmount creates a virtual file system using FUSE (Chapter A.4).

> File System in Userspace (FUSE) is a simple interface for userspace programs to export a virtual file system to the Linux kernel. FUSE provides a secure way to allow non-privileged users to build and deploy their own file system implementations.

After the image is converted, it needs some data for configuration, such as processor count and RAM size.
With the next command the number of processing engines will be displayed:

”grep -i processor /proc/cpuinfo | tail -n 1 | awk ’{print $3}”’,

and the RAM size: *”grep -i MemTotal /proc/meminfo | awk ’{printf ”%.0f”, $2 * 0.6 / 1024}”*.
Here, the current RAM memory in KB is determined and converted to MB and multiplied by the factor 0.6. The 0.6 factor states that 60% of the main memory is reserved for the virtual

machine.

In the ”Virtual PC” program it is also possible to specify some parameters for the virtualization. Possible parameters would be ”boot”, ”cpu”, ”machine”, ”k” (language), ”bios”, ”no-quit”, ”vga” and transferred ”usb devices”. If more parameters are needed now or in the future, then the program would have to be adapted accordingly. With the parameters, specific settings can be done, among others, from the operating system of the image.

Now QEMU can be started. If QEMU does not start correctly, the configuration file can be customized with the ”Virtual PC” program interface. The adaptation consists in changing the specified parameters for the system to be virtualized, such as the graphics card used in the seized PC. After this QEMU will be restarted. This process can be repeated until there is no more error and the configuration file has been adapted to the evidence. QEMU can now be started with the specified operating system of the image.
After launching QEMU, the investigators can work and search with the evidence as if they were sitting right in front of the seized computer. If all the necessary data is spotted during the search, the virtual machine can be stopped again. If QEMU and the work with the virtual machine are finished, the temporary directory that was created before must be cleaned up. To do this, the dd file in the *”/tmp”* directory must now be cleared.
We have created and mounted the dd file with xmount. Xmount creates a virtual file system with FUSE. To unmount the file I used the command *”fusermount -u”*.
Fusermount (Chapter A.1.11) is the program for mounting and unmounting FUSE file systems (Daniel Baumann and Miklos Szeredi, 2018).
Once finished, the temporal files are cleaned automatically. The environment can be used again in the same state before the virtualization was started. This ensures that it can not be forgotten and came to an inadvertent mix of data.

4.2.3.4 Mounting the evidence filesystem

In this chapters the forensic integration of partitions form an evidence image into the analysis PC is explained. It was first determined which Linux commands could be used for a read-only partition deployment. I have then decided to use the ”pmount” command (Chapter A.1.8). As opposed to the standard command ”mount”, pmount allows normal users to mount removable media without root privileges and without having a matching entry in *”/etc/fstab”*.

To mount a partition now, the starting point and the size of the partition must first be determined. This work should be done by the command ”parted” (Chapter A.1.13). Parted belongs to the standard Mint installation.

Before parted can run, an E01 image has to be converted.
Parted needs a pure block device (disk) image as device. But by default, E01 files are compressed at the block level.

> The EnCase image format (E01), based on the Expert Witness Compression Format (EWF), contains a separate hash for each segment. It includes the hash file and certain information about the image. The information entered by the examiner is

> stored in the image file itself. An E01 file is more than just the image, it also contains metadata about the image file.
> Therefore, the E01 format of EnCase is not usable for ”parted” (Paul Cobbaut, 02.05.2015).

To convert, I use the program ”xmount” (Chapter A.1.10), which does not belong to the standard mint installation and has to be installed additionally.
With xmount, you can convert between multiple input and output hard disk image types.

> Xmount creates a virtual file system using FUSE (Chapter A.4). This virtual file system represents the input image. The virtual representation can be in Raw dd, DMG, VHD, VirtualBox’s virtual disk file format, or in VmWare’s VMDK file format. Input images can be RAW-DD, Expert Witness Compression Format (EWF), or Advanced Forensic Format (AFF) files. In addition, xmount also supports virtual write access to the output files. These output files are redirected to a cache file. Thus, acquired hard disk images can be booted with QEMU, KVM, VirtualBox, VmWare or similar (Cory Altheide and Harlan Carvey, 2011).

After ”xmount” has converted the E01 Fromat of the evidence images to a raw / dd format, ”parted” (Chapter A.1.13) can determine boot sectors and the size of partitions. With these facts, ”losetup” (Chapter A.1.12) can now set up a loop device.

> Losetup is used to link loop devices (a pseudo-device that makes a file accessible as a block device) with regular files or block devices. Losetup can disconnect loop devices and query the status of a loop device. If only the loopdev argument is specified, the status of the corresponding loop device is displayed (die.net, 2018b).

Now ”pmount” (Chapter A.1.8) can mount the found and created loops.
All of the evidence, mounted partitions are now made available in the file manager of the analysis PC and can be examined by the investigator.

4.2.3.5 Mounting Storage Media

Here is a brief description of how to mount storage media in ”Virtual PC”. First, it has to be found out which storage media are to be offered in ”Virtual PC” for storage. To do this, use the command ”lsblk” (ChapterA.1.4) to query all drives that exist in the system. The response can include both mounted and unmounted drives. Here you can also filter for a mount point. All drives with mount point are already mounted in the system. To make sure no mounted drives are displayed, you can disable the auto mounting from the system [13]. If now a portable storage device is plugged in as an USB disk, it will not be detected automatically. It has to be mounted manually if you want to see the drive in the system. Either with the command ”pmount” or via nemo, the Cinnamon File Manager [13]. The ”blkid” (ChapterA.1.5) command now gives us additional information about the drives, such as the label and type.
Now we have the addition that ”Virtual PC” wants, it will only show storage media that are

[13] `https://www.systutorials.com/241394/how-to-disable-auto-mounting-on-linux-mint-cinnamon/`

not mounted. All these storage media are displayed in a list and via a radio button, the corresponding drive can now be selected for data backup. This drive will be attached to the startup command of QEMU and integrated into the virtual machine. When shutting down the virtual machine, the storage medium is unmounted again and can then be mounted in the Analyzer PC to further process the data.

4.2.3.6 Saving Search Results

When we virtualize the evidence, any data store can be included that can then be used within the virtual machine to store found evidence data. If the evidence HDD is included as partition(s) in the file system of the analysis PC, the evidence data found by the investigator can be backed up on the embedded data store.
From several investigators came the proposal to add a virtual memory. Extraordinary situations arise out of the blue and in such a situation an external data storage device (e.g. a USB flash drive) is not always available.
Thats why a virtual memory stick has been built that can be used at any time. Evidence data found by the investigator can be stored on the virtual data store at any time. When the virtualization is finished, the virtual USB flash drive can be integrated into the file system of the analysis PC.
Now the found evidence data can be further processed on the analysis PC and follow-up actions can be carried out immediately.

The virtual memory stick can be created with the Bashscript ”Virtual_flash_drive.sh”, which is listed in the appendix in Chapter B.

Here is the most important part from the script:

```
IMG=”virtual_USB.img”
LABEL=”Virtual_USB”
dd if=/dev/zero of=${IMG} bs=1 count=0 seek=64G
parted −−script −−machine ${IMG} mklabel msdos mkpart primary ntfs 0% 100%
LOOP=$( losetup −−partscan −−show −−find ${IMG} )
mkfs.ntfs -f -Q -L ${LABEL} ${LOOP} -p1
losetup -d ${LOOP}
```

An image file *”virtual_USB.img”* is created with a virtual size of 64GB. Data up to 64GB can be stored in the file. With the command *”du -sh −−apparent-size virtual_USB.img”*, the virtual size can be queried. To query the physical size, use the *”u -sh virtual_USB.img”* command. The result is *”67M virtual_USB.img”*. The file ”virtual_USB.img” has a current physical size of 67 MB. The more data stored in the file, the larger the physical size becomes.

4.2.3.7 Summary

For the experimental work, a laptop with Linux Mint 19 OS was used and configured in a way that is needed for the virtualization. Starting with mounting tools experiments came to the result to use QEMU and a Python program for the configuration. Some BIOS-settings had to be set. With a self-written program, named ”Virtual PC” I was able to virtualize successful Windows and Linux images. Apple Mac images were also tested, but I found no way for a needed simply virtualization. That was the reason, why I have discarded to virtualize Mac images. To be sure that ”unskilled persons” were able to work with my self-written program ”Virtual PC” and the used environment, the next experiment was started. Ten ”unskilled persons” were short instructed to settings and work with the program. Additional wishes like adding a virtual data storage to save results were added to my program.

Chapter 5

Discussion, Conclusion and Further Work

5.1 Discussion

In this Chapter the research questions posed in Chapter 1.3 will be answered. The questions are discussed in the same order as they are named.

Main Research Question 1:
Can a person, unskilled in IT forensics (such as an investigator) virtualize an image from seized PCs with the required hardware and software and without professional help?

The use of virtualization in forensics is nothing new. But while working, in a extended stressful period, I came up with the idea that the virtualization by unskilled staff can be done. In this period, we had far too few staff so that a lot of evidence could not be processed in the available time.

During my Literature Research, I learned then that others have worked on this topic of using unskilled persons for forensic work, such as Ben Hitchcock, Nhien-An Le-Khac, Mark Scanlon (2016) [1].

After researching various ways of virtualization, the virtualization method used has been implemented with QEMU. Previously, I have doubted that a forensically unskilled person can perform this work. However, my systematic group study has shown that forensically unskilled investigators are able using my new software, after a brief introduction, to virtualize an image in a forensic way.

With a rate of 93.33% successfully visualized images, the research question 1 can be answered with a definite positive answer that is yes. In the case of the 6.67% that were unsuccessful in the first attempt, with some slight changes and done help have been also successfully visualized.

[1]`https://www.sciencedirect.com/science/article/pii/S1742287616300044`

The result of the second attempt shows a quota of 100%.
The result is, as described, more than acceptable and fully meets my expectations. From years of experience with the handling of evidence and working with investigators, I have written this script in a way that I have expected such a result. Thus, the result of this research offers the prosecution authorities additional opportunities for the future. Unskilled persons can not only work in this area. Further work can develop these potential fields of application.

Research Question 2:
Can I find a commercial or noncommercial way or a program that will be able to do the virtualization with a previously defined test series from PC images?

Virtualization is typically an approach to pooling and sharing technology resources to simplify management and increase asset utilization so that IT resources can better meet business needs. In this research images of seized PCs should be virtualized, with different methods and approaches.

According to my research, both commercial and non-commercial software is virtualized to images. But none of the researched programs fulfills all self-imposed goals as described in Chapter 1.1. The program that fulfills the most criteria is QEMU. This makes QEMU the best option for this research under Linux. As described in Chapter 2.5.2, QEMU was developed to virtualize and emulate hardware (in collaboration with KVM).
The reason that QEMU can be controlled from the command line and therefore, it is predestined for this thesis.
Thereby, when starting the virtual machine different hardware such as e.g. Graphics cards, PCI and ISA systems, CPUs, sound cards and other interfaces (USB, RS232, etc.), BIOS systems (BIOS, UEFI, etc.) can be set with parameters, that worked also relatively fast to find and set the correct configuration for an image.

From statistically most used operating, the following systems have been incorporated and tested in the experimentel part of my research:

- Windows 10/8/7/Vista/XP
- Linux (all Linux installations that do not use a self-built kernel)

Because the created start commands could not be saved with QEMU, I decided during this research to program my own configuration tool, which should be able to perform this task. I have created for this part a configration tool and named it ”Virtual PC”. With this tool the assembled configurations for an image can be tried out quickly and then saved. After finishing the configuration with ”Virtual PC”, the virtualization can be started from ”Virtual PC”.

At this point, the second research question must be denied. But by using the self-programmed application ”Virtual PC”, I was able to do this and also to answer the second question with yes.

Research Question 3:
When developing my own virtualization method, are there any special features of the hardware to consider when virtualizing images?

This question about special functions or special hardware refers both to the analys PC and to the computers that have been seized and thus to the image created by it.

For virtualization to be carried out on an analysis PC, virtualization-specific parameters must be set in the BIOS/UEFI as described in Chapter 4.2.3.1. Further features or hardware we do not put to the analysis PC.

Image can also make demands on virtualization. If an image has been created by a confiscated PC that has been operating with a UEFI-BIOS, a UEFI program, the OVMF package, must be installed on the analysis PC in order to virtualize the image (Chapter 4.2.3.2). On some confiscated PCs, e.g. a USB dongle must be used (often used to confirm the software licenses). But this can easily be specified as USB transfer parameters in QEMU (Chapter 4.2.3.3).

Some functions haven't been realized, such as secure boot. Secure Boot is part of the UEFI specification, which is intended to guarantee the authenticity or integrity of important software parts of the firmware[2].
Not all possible hardware has been considered here, such as Apple Computer with the Mac OS. These are points that we want to address in the Chapter 5.3.

5.2 Conclusion

In this chapter we will try to answer the research questions mentioned in Chapter 1.3 and discuss whether we have reached the goal of this research project.
The answers are based on the experimental work, both my empirical study and that of my systematic group study. Further, the answers described in Chapter 4.2 are based on the discussion of the results in Chapter 5.1.
There is one main research question and two further research questions, as described in Chapter 1.3.

[2] `https://www.thomas-krenn.com/en/wiki/UEFI_Secure_Boot`

Main Research Question 1:
Can a person, unskilled in IT forensics (such as an investigator) virtualize an image from seized PCs with the required hardware and software and without professional help?

Nowadays almost all law enforcement agencies in different countries have the same problems. Forensic IT specialists are completely utilized. There are too many cases with a lot of evidence objects for far too few forensic IT specialists. One way to fix this problem led to the idea of employing unskilled personnel for previously defined work. This personnel can be deployed as quickly as possible even in extreme situations or specific threat situations, such as Terrorism or hostage taking. In these cases, it is possible for a normal investigator, defined as digital forensic ”unskilled person”, to use this method and the resulting ”Virtual PC” (Chapter 4.2.3) configuration tool to quickly and securely access the data of the compromised PC and provide the initial results. This thesis has shown that unskilled staff was able to do this working with the provided environment (in the part ”Main Research Question 1” in Chapter 5.1).

My systematic group study (Chapter 4.2.2) has shown that an extension of the experiment should be possible. The investigators have demanded that not only a virtualization should be possible, but also the pure forensic mount of partitions should be provided quickly and easily. In some situations, experienced investigators get their needed results faster using this method.

Research Question 2:
Can I find a commercial or noncommercial way or a program that will be able to do the virtualization with a previously defined test series from PC images?

In this research question, it’s about finding a virtualization software that meets all the requirements from the Chapter 1.1, whether it’s commercial or not. The optimal would be a noncommercial software, then we would reach all our internal goals. Unfortunately, there was no tool that met all these requirements. The second choice fell on QEMU. Since not all requirements could be met here, I used my self-programmed configuration tool ”Virtual PC” (Chapter 2.5.2) to correct this.
Further description and discussion can be found in in the part ”Research Question 2” in Chapter 5.1.

Research Question 3:
When developing my own virtualization method, are there any special features of the hardware to consider when virtualizing images?

As discussed in part ”research question 3” in Chapter 5.1, there is something special about the development of my own virtualization methodology that hardware must be aware of when virtualizing images. Solutions have been found for various functions or hardware. But there was also hardware for which no solutions were found during this research, such as Apple’s Mac OS.

5.3 Further Work

Within this research, not all possible topics could be considered. During the preparation, a Windows-based system was also considered. At least a Linux based system was preferred, but there are ways of implementing this program on a Windows system.
Encryptions files offers further work.
During my work, I only used images that were not encrypted. Thus, here is potential to expand the work so that Virtual PC (Chapter 4.2.3) can work with encrypted images or images with encrypted partitions. Furthermore, password-protected data or operating systems have not been considered. This is also a point that could be incorporated into the program.
Another improvement is the programming of additional file systems that have not yet been considered. Like Apple’s Mac OS, for example, which is still ahead of Linux in the popularity scale (Figure 4.3). This would certainly improve the capabilities of the program ”Virtual PC” and increase the performance and flexibility of the unskilled person (Chapter 1.2) even further.

5.4 Summary

With the used environment and the self-written program ”Virtual PC” images from Windows and Linux OS were successfully mounted and virtualized. Even an unskilled person that were police investigators has been able to work with the program and environment. Additional functions have been programmed, like virtual storage. This was a wich voiced during the experiment with the unskilled investigator.
The program ”Virtual PC” can be customized and supplemented with additional functions that open a variety number of possibilities for future work.

Bibliography

Preston Miller and Chapin Bryce, 2016. Learning Python for Forensics, english edition Edition. Packt Publishing, ISBN: 1783285230.

Admin, Top Password Software, Inc, 2018. How to set a disk or volume read-only in windows 10/8/7. Webpage, last checked: 16.12.2018.
URL `https://www.top-password.com/blog/set-a-disk-or-volume-read-only-in-windows/`

Alexander Geschonneck, 2014. Computer Forensik, 6th Edition. dpunkt.verlag, ISBN: PDF 978-3-86491-489-8.

Alieda Blandford, 2016. Google, public libraries, and the deep web. Webpage, last checked: 24.07.2018.
URL `https://ojs.library.dal.ca/djim/article/view/2015vol11Blandford`

Anton Shilov, 2017. Toshiba announces 14 tb pmr mg07aca hdd: 9 platters, helium-filled, 260 mb/s. Webpage, last checked: 05.08.2018.
URL `https://www.anandtech.com/show/12134/toshiba-announces-14-tb-pmr-mg07aca-hdd-9-platters-heliumfilled-260-mbs`

Arne Arnold, 2015. Per notfall-cd einfach die windows-registry bearbeiten. Webpage, last checked: 11.10.2018.
URL `https://www.pcwelt.de/tipps/Per-Notfall-CD-einfach-die-Windows-Registry-bearbeiten-Registry-9688907.html`

Arnold Robbins, October 2005. Unix in a Nutshell, 4th Edition, fourth edition Edition. O'Reilly Media, Inc., ISBN: 0596100299.

Ben Hitchcock, Nhien-An Le-Khac, Mark Scanlon, 2016. Digital investigation. Elsevier Journal 10, 75–85.

Brian Carrier, 2005. File System Forensic Analysis, edition 01 Edition. Addison Wesley Professional, ISBN: 978-0321268174.

Bundesamt für Sicherheit in der Informationstechnik, BSI, 2011. Leitfaden it-forensik. PDF, last checked: 13.01.2019.
URL `https://www.bsi.bund.de/SharedDocs/Downloads/DE/`

BSI/Cyber-Sicherheit/Themen/Leitfaden_IT-Forensik.pdf;jsessionid=43B7AC366A216DE24A519AA8FD76A0F1.1_cid341?__blob=publicationFile&v=2

Clem, 2018. The linux mint blog. Webpage, last checked: 05.02.2019.
URL https://blog.linuxmint.com/?p=3599

Cory Altheide and Harlan Carvey, 2011. Digital Forensics with Open Source Tools, n/a Edition. Syngress, Elsevier.

Dan Nanni, N/A. How to check cpu info on linux. Webpage, last checked: 15.04.2019.
URL https://jin-yang.github.io/reference/linux/monitor/check-cpu-info-linux.pdf

Daniel Baumann and Miklos Szeredi, 2018. Ubuntu manpage repository. Webpage, last checked: 15.02.2019.
URL http://manpages.ubuntu.com/manpages/trusty/man1/fusermount.1.html

David Wolski, 2014. Virtualisierung: So arbeitet die technik unter der haube. Webpage, last checked: 16.08.2018.
URL https://www.pcwelt.de/ratgeber/Virtualisierung__So_arbeitet_die_Technik_unter_der_Haube-Der_PC_im_PC-8903176.html

David Wolski, 2015. Die besten virtualisierer für linux im überblick. PDF, last checked: 16.08.2018.
URL https://www.pcwelt.de/ratgeber/Die-besten-Virtualisierer-fuer-Linux-im-Ueberblick-Linux-Virtualisierer-9580443.html

David Wolski, 2016. Fünf beliebte virtualisierer für linux im vergleich. Webpage, last checked: 15.08.2018.
URL https://www.computerwoche.de/a/fuenf-beliebte-virtualisierer-fuer-linux-im-vergleich,3218056,2

debian, 2011. Pmount.conf(5). Webpage, last checked: 16.12.2018.
URL https://manpages.debian.org/experimental/pmount/pmount.conf.5.en.html

Diana Nadeborn, 2018. It-strafrecht, digitale beweismittel. Webpage, last checked: 05.07.2018.
URL http://it-strafrecht.org/zugriff-auf-elektronische-beweismittel/

die.net, 2018a. grep(1) - linux man page. Webpage, last checked: 12.12.2018.
URL https://linux.die.net/man/1/grep

die.net, 2018b. losetup(8) - linux man page. Webpage, last checked: 11.12.2018.
URL https://linux.die.net/man/8/losetup

die.net, 2018c. pmount(1) - linux man page. Webpage, last checked: 13.12.2018.
URL https://linux.die.net/man/1/pmount

Dirk Makowski, 2018. Winhistory.de. Webpage, last checked: 05.08.2018.
URL `https://www.winhistory.de/index.php`

DistroWatch, 2018. Top ten - distributionen. Webpage, last checked: 11.08.2018.
URL `https://distrowatch.com/dwres.php?resource=major`

Dr. Sabine Vogt, 2017. Das darknet rauschgift, waffen, falschgeld, ausweise das digitale kaufhaus der kriminellen? Webpage, last checked: 08.07.2018.
URL `https://www.kriminalpolizei.de/ausgaben/2017/juni/detailansicht-juni/artikel/das-darknet.html`

Dr. Vincenzo Ciancaglini, Dr. Marco Balduzzi, Robert McArdle, and Martin Rösler, 2015. Below the surface: Exploring the deep web. Wevpage, last checked: 17.08.2018.
URL `https://documents.trendmicro.com/assets/wp/wp_below_the_surface.pdf`

elysium, 3. Januar 2017. Der groe linux distribution-report: Die 10 beliebtesten distributionen 2016. Webpage, last checked: 25.07.2018.
URL `https://games4linux.de/der-grosse-linux-distribution-report-die-10-beliebtesten-distributionen-2016/`

F. Kalhammer, 2001. Das mounten von dateisystemen. Webpage, last checked: 01.08.2018.
URL `http://www.linux-praxis.de/linux1/filesystem4.html`

Falk Gaentzsch and Prof. Norbert Pohlman, 2015. Virtualisierung. Webpage, last checked: 15.07.2018.
URL `https://norbert-pohlmann.com/app/uploads/2015/12/Virtualisierung-Prof-Norbert-Pohlmann.pdf`

futurezone, 2018. Grösste ssd der welt vorgestellt. Webpage, last checked: 22.07.2018.
URL `https://futurezone.at/produkte/100-terabyte-groesste-ssd-der-welt-vorgestellt/400008000`

Gavin Thomas Garrad and Stepan Maluchev, 15.05.2015. Ourbox, bachelor thesis. Webpage, last checked: 11.10.2018.
URL `https://brage.bibsys.no/xmlui/handle/11250/294994`

Graham Williams, 1995-2018. Gnu/linux desktop survival guide. Webpage, last checked: 13.10.2018.
URL `https://www.togaware.com/linux/survivor/Standard_Groups.html`

Harlan Carvey, 2009. Windows Forensic Analysis DVD Toolkit 2E, 1st Edition. Syngress, Elsevier, ISBN: 978-1-59749-422-9.

Heinrich Schwietering, 2018. Ubuntuusers, grundlagen. Webpage, last checked: 10.08.2018.
URL `https://wiki.ubuntuusers.de/Partitionierung/Grundlagen/`

Horst Dumstorff, November, 2017. Project thesis, forensic data backup of mobile apple phones. NTNU.

Jack Wallen, 2017. How to work with remote filesystems using sshfs. Webpage, last checked: 24.08.2018.
URL `https://www.techrepublic.com/article/how-to-work-with-remote-filesystems-using-sshfs/`

Janez Puhan, 2015. Operating Systems, Embedded Systems, and Real-Time Systems, 1st Edition. FE Publishing, ISBN: 78-961-243-275-1.

Johannes Plötner and Steffen Wendzel, 2012. Linux, Das umfassende Handbuch, 5th Edition. Rheinwerk Computing, ISBN: 978-3-8362-1822-1.

Karim Yaghmour and Jon Masters and Gilad Ben-Yossef and Philippe Gerum, 2008. Building Embedded Linux Systems, second edition Edition. OReilly Media, Inc, ISBN: 978-0-596-52968-0.

Linux Mint, 2017. Official user guide. PDF, last checked: 04.10.2018.
URL `https://linuxmint.com/documentation/user-guide/Cinnamon/english_18.0.pdf`

Marcus Fischer, 2010. Ubuntu GNU/Linux, 7th Edition. Rheinwerk Computing, ISBN: 978-3-8362-1945-7.

Margaret Rouse, 2015. Computerkriminalitt (cybercrime). Webpage, last checked: 10.10.2018.
URL `https://www.searchsecurity.de/definition/Computerkriminalitaet-Cybercrime`

Marjie T. Britz, Ph.D., 2013. Computer Forensics and Cyber Crime, third edition Edition. Pearson Education, Inc., ISBN: 978-0-13-267771-4.

Mark L. Mitchell and Alex Samuel and Jeffrey Oldham, 29 Jun 2001. Advanced Linux Programming, 1st Edition. CodeSourcery, LLC, ISBN: 0735710430.

Martin Pitt, 2018. Linux man page. Webpage, last checked: 24.07.2018.
URL `https://linux.de.net/man/1/pumount`

Martin Pitt and Vincent Fourmond, 2018. Linux man page. Webpage, last checked: 24.07.2018.
URL `https://linux.de.net/man/1/pmount`

Matthew E. Hoskins, 2006. Sshfs: Super easy file access over ssh. Webpage, last checked: 10.09.2018.
URL `https://www.linuxjournal.com/article/8904`

Matthew Portnoy, 2012. VIRTUALIZATION ESSENTIALS, 1st Edition. John Wiley and Sons, Inc., ISBN: 978-1-118-17671-9.

Michael Hale Ligh, Andrew Case, Jamie Levy and AAron Walters, 2014. The Art of Memory Forensics, 1st Edition. John Wiley and Sons, Inc., ISBN: 978-1-118-82509-9.

Michael Kerrisk, 2018. Linux man pages online. Webpage, last checked: 15.10.2018.
URL `http://man7.org/linux/man-pages/index.html`

Michael Kofler and Ralf Spenneberg, 2012. KVM für die Server- Virtualisierung, 2014th Edition. ebooks.kofler, ISBN: 978-3-902643-22-3.

Microsoft, N/A. Mount-diskimage. Webpage, last checked: 18.08.2018.
URL https://docs.microsoft.com/en-us/powershell/module/storage/mount-diskimage?view=win10-ps

Miklos Szeredi, 2018. Ubuntu, trusty (8) mount.fuse.8.gz. Webpage, last checked: 19.08.2018.
URL http://manpages.ubuntu.com/manpages/trusty/man8/mount.fuse.8.html

Moritz Wanke, 2016. Die grösste sd-karte der welt: Sandisk stellt 1-tbyte-karte vor. Webpage, last checked: 10.07.2018.
URL https://www.chip.de/news/Die-groesste-SD-Karte-der-Welt-SanDisk-stellt-1-TByte-Karte-vor_100397250.html

Moy, T., and Tyrkus, M. J. (Eds.), 2016. American law yearbook. Farmington Hills, ISBN: 1784722.

Paul Cobbaut, 02.05.2015. Linux system administration. Webpage, last checked: 10.10.2018.
URL linux-training.be/linuxsys.pdf

PowerQuest Corporation, 2002. Zum verständnis von festplatten. PDF, last checked: 01.07.2018.
URL http://digilib.happy-security.de/files/Grundwissen_-_Festplatten.pdf

QEMU, 2018. Qemu the fast! processor emulator. Webpage, last checked: 21.12.2018.
URL https://www.qemu.org/

Robert Love, Stephen Figgins, Ellen Siever, Arnold Robbins, 2009. Linux in a Nutshell, 6th Edition. O'Reilly Media, Inc.

Robert Schanze, 2018. Top 10 linux-distributionen 2018 im vergleich. Webpage, last checked: 09.09.2018.
URL https://www.giga.de/extra/linux/gallery/top-10-linux-distributionen-2018-im-vergleich-s/

Sebastian Helmer, 2012. Geschichte der datenträger. Webpage, last checked: 23.07.2018.
URL https://www.fmi.uni-jena.de/fmimedia/Fakultaet/Institute+und+Abteilungen/Abteilung+f%C3%BCr+Didaktik/GDI/20112012/homepage/Geschichte+der+Datentr%E2%80%9Eger.pdf

SilkeCCM, 2018. Geschichte der festplatte. Webpage, last checked: 04.07.2018.

Sphinx, 2017. Linux mint installation guide. Webpage, last checked: 13.08.2018.
URL https://linuxmint-installation-guide.readthedocs.io/en/latest/index.html

Stefan Meier, 2016. Digitale forensik in unternehmen. Webpage, last checked: 03.07.2018.
URL https://epub.uni-regensburg.de/35027/1/Dissertation_Veroeffentlichung_Stefan_Meier_A5_digital.pdf

Stefan Weil, 2018. Qemu version 3.0.0 user documentation. Webpage, last checked: 06.01.2019.
URL `https://qemu.weilnetz.de/doc/qemu-doc.html`

Stephen G. Kochan and Patrick H. Wood, 1987. UNIX TEXT PROCESSING, second printing - 1988 Edition. OReilly and Associates, Inc., ISBN: 0-672-46291-5.

SUSE, 2018. Accessing file systems with fuse. Webpage, last checked: 08.07.2018.
URL `https://doc.opensuse.org/documentation/leap/startup/html/book.opensuse.startup/cha.fuse.html`

Sven Schefer, 2011. Ssd - solid state drive. PDF, last checked: 11.07.2018.
URL `https://wr.informatik.uni-hamburg.de/_media/teaching/wintersemester_2010_2011/sds-1011-schefer-ssds-ausarbeitung.pdf`

Thomas Ritzau and Robert Warnke, 23. Mär. 2010. QEMU-KVM and libvirt, n/a Edition. qemu-buch.de, ISBN: 978-3-8370-0876-0.

Thuy Anh Nguyen, 2015. Die festplatte. Webpage, last checked: 04.07.2018.
URL `https://medienwissenschaft.uni-bayreuth.de/dimensionen/entwuerfe/geschichte/festplatte/geschichte.html`

Tuxera, 2017. ntfs-3g third generation read/write ntfs driver. Webpage, last checked: 03.08.2018.
URL `http://man7.org/linux/man-pages/man1/gpasswd.1.html`

Ubuntu Documentation, 2009. Fuseiso. Webpage, last checked: 1.07.2018.
URL `https://help.ubuntu.com/community/FuseIso`

WikiSysop, 2017. Kernel virtual machine. Webpage, last checked: 20.12.2018.
URL `https://www.linux-kvm.org/page/Main_Page`

Appendix

A The used OS Linux, the commands, additional configurations and Linux Users and Groups

A.1 Used Linux programs

Here is a list of all the programs that are explicitly needed for this research.
If these programs have not been installed during a standard installation of a Linux distribution, each program can be subsequently installed.

A.1.1 mkdir

Mkdir is the command to create a directory with an absolute or relative pathname (Arnold Robbins, October 2005).
(GNU coreutils 8.21 package)

A.1.2 cp

The cp command makes a copy of a file or copies multiple files into a directory (Arnold Robbins, October 2005).
(GNU coreutils 8.21 package)

A.1.3 rm

The rm command removes files and directories. There is no way to reverse this process (Arnold Robbins, October 2005).
(GNU coreutils 8.21 package)

A.1.4 lsblk

lsblk lists information about all available or the specified block devices. The command prints all block devices (except RAM disks) in a tree-like format by default. (Michael Kerrisk, 2018).
(util-linux package)

A.1.5 blkid

The blkid program is the command-line interface. It can determine the type of content (e.g. filesystem or swap) that a block device holds, and also the attributes (tokens, NAME=value pairs) from the content metadata (e.g. LABEL or UUID fields). (Michael Kerrisk, 2018).
(util-linux package)

A.1.6 grep

Grep is a search command. Grep can search for PATTERN in each FILE. A FILE of - stands for standard input. If no FILE is given, recursive searches examine the working directory, and nonrecursive searches read standard input. By default, grep prints the matching lines (die.net, 2018a).

If we use the program in this way with the following parameters, *grep MemTotal /proc/meminfo*, the Outputs gives the total memory available to system after reserved (Mark L. Mitchell and Alex Samuel and Jeffrey Oldham, 29 Jun 2001).

If we use the program in this way with the following parameters, *grep -i processor /proc/cpuinfo*, the program cpuinfo outputs only the number of processors (Dan Nanni, N/A).
(GNU grep 2.16 package)

A.1.7 mktemp

Mktemp can create a temp file oder a directory. By default, mktemp create a file in the */tmp* directory and return its name on standard output.

If mktemp can successfully generate a unique filename, the file (or directory) is created with file permissions such that it is only readable and writable by its owner (unless the -u flag is given) and the filename is printed to standard output.
The mktemp utility takes the given filename template, if indicated, and overwrites a portion of it to create a unique filename. The template may be any filename with some number of Xs appended to it, for example */tmp/tfile.XXXXXXXXXX*. If no template is specified a default of tmp.XXXXXXXXXX is used.

If the parameter *"−−suffix="zzz""* is used the file ends with this characters, if *"-d -suffix = "zzz""* if used the directory ends with this characters (Arnold Robbins, October 2005).
(GNU coreutils 8.21 package)

A.1.8 pmount

Pmount (”policy mount”) is a wrapper around the standard mount program that allows normal users to mount removable media without a matching */etc/fstab* entry. It also supports encrypted devices that use dm-crypt and have LUKS metadata. It will automatically recognize encrypted drives. If a LUKS-enabled cryptsetup is installed, pmount will use it to decrypt the device first. Before decrypting the password is automatically queried to then mount the associated unencrypted device.

If you start pmount without arguments, a list of mounted removable disks is printed (Martin Pitt and Vincent Fourmond, 2018).

A.1.9 pumount

Pumount (”policy umount”) is a wrapper around the standard umount program that allows normal users to umount removable devices without a matching */etc/fstab* entry.
It also supports encrypted devices that use dm-crypt and have LUKS metadata. It will automatically recognize encrypted drives. If a LUKS-capable cryptsetup is installed, pumount will umount the mapped device instead and call cryptsetup to close the decrypted device afterwards.

Pumount expects only one argument, the device that should be unmounted. This will umount device from a directory below /media if policy is met (Martin Pitt and Vincent Fourmond, 2018).
(pmount package)

A.1.10 xmount

In computer forensics, most images are stored in popular formats such as ”Data Definition” (dd), ”Advanced Forensic Format” (AFF), ”Expert Witness Format” (EWF) and ”The Encase image” (E01).

To convert these formats, the Linux tool xmount is used.
Xmount allows you to convert on-the-fly between multiple input and output harddisk image types.

This tool creates virtual images in mount points. Xmount can import images into the common formats ”Data Definition” (raw format) (dd), ”Advanced Forensic Format” (AFF), ”Expert Witness Format” (EWF) and ”The Encase image” (E01) and can be converted into formats converts, which can then be used by QEMU or other virtualization solutions. xmount uses the ”file system in user space” (FUSE).
Xmount also supports virtual write access to the output files that is redirected to a cache file. This makes it possible to boot acquired harddisk images using QEMU, KVM, VirtualBox, VmWare or alike (Cory Altheide and Harlan Carvey, 2011).

(xmount package)

A.1.11 fusermount

Fusermount is a command of the simple interface File system in USErspace (FUSE). FUSE is for userspace programs to export a virtual filesystem to the Linux kernel.

As normal user you can't mount anything that the administrator hasn't somehow given you permission to mount. There are many ways to escalate privileges through mounting.
For this reason, only root can call the system call "mount".

So fusermount aims to provide a secure method for non privileged users to create and mount their own filesystem implementations (SUSE, 2018).
(fuse package)

A.1.12 losetup

If you want to mount an image file as a normal device, attach a loop device to a regular file or block device. To do that use losetup.
After creating a loop device, mount this device to your file system (die.net, 2018b).
(util-linux package)

A.1.13 parted

Parted is a command line tool that allows you to easily manage hard disk partitions. You can add, delete, shrink and extend disk partitions along with the file systems located on them.
In this case you can find out a start sector and the size of a partition within an image (Paul Cobbaut, 02.05.2015).
(parted package)

A.2 Additional configuration for Linux

A.2.1 /etc/pmount.allow

For a smooth operation a configuration settings must still be made. To be able to run the pmount command smoothly, the file *"/etc/pmount.allow"* and the file *"/etc/pmount.conf"* have to be adapted.

In the *"/etc/pmount.allow"* all devices must be entered, which a user may mount with pmount.

e.g .:
/dev/loop[0-9]
/dev/sd[b-z]
/dev/sd[b-z][0-9]

(die.net, 2018c).

A.2.2 /etc/pmount.conf

From a strict point of view, pmount supports various operations that can be considered unsafe, but on a PC in general are relatively safe or at least acceptable.
All these operations are not allowed by default.
In the *"/etc/pmount.conf"* file all these operations have to be explicitly allowed to the user.

If loop_allow is true, then users can mount personal files using loopback devices.
If you want to allow users to mount loopback devices, you should also specify here a comma-separated list of whitelisted loop devices that the users can use.

e.g .:
loop_allow = yes
loop_devices = /dev/loop0, /dev/loop1, /dev/loop2

(debian, 2011).

A.2.3 /etc/fuse.conf

FUSE has a number of global options that can be used with all modules. The most important ones are *"-o allow_other"* and *"-o allow_root"*. This allows access to the mounted file system for other users.

However, for these options to be used, the configuration file *"/etc/fuse.conf"* must be created:

user_allow_other

(Miklos Szeredi, 2018; Daniel Baumann and Miklos Szeredi, 2018; SUSE, 2018).

A.3 Linux Users and Groups

Users and Groups are, to control access to the system's files, directories, and peripherals.

In addition for this research, users and group affiliation must also be considered.
Linux is a multi-user system in which the users have a group affiliation.
Each user belongs to at least one group, but may also belong to several groups.

The command *"gpasswd -a <user><group>"* allows users to be included in specific groups. This command can only be executed with administrator rights (Michael Kerrisk, 2018).

For this research you have to be in the following groups to ensure a complete function:

A.3.1 Group disk

The disk device nodes are group accessible to disk so that programs that need access to them will set their group ID to be disk. This group has "write access" to all the raw disk devices (/dev/hd* and /dev/sd*), so assigning users to group disk is both dangerous and a security risk (Graham Williams, 1995-2018).

A.3.2 Group fuse

FUSE is a Linux kernel module that makes it possible to move file system drivers from kernel mode to user mode. It also allows non-privileged users to "mount" their own file systems.
The fuse group lets you limit which users are allowed to use FUSE-based filesystems. This is important because FUSE installs setuid programs, which always carry security implications. On a highly secured system, access to such programs should be evaluated and controlled (Matthew E. Hoskins, 2006).

A.3.3 Group kvm

The Kernel Virtual Machine (KVM) is a virtualization technology. KVM is an extension of the Linux kernel, enabling hardware-accelerated virtualization of guest operating systems.
KVM is a relatively small kernel module that only works in combination with the emulation software QEMU.
KVM requires a CPU with hardware virtualization capabilities and turns the QEMU emulator into a hardware virtualization system.
So the user who rmkdiruns QEMU must be in the KVM group so that QEMU can perform its performance and function (Michael Kofler and Ralf Spenneberg, 2012).

A.3.4 Missing user group

If a group is not present/created, the administrator can create this group, such as: for example, the group "fuse" that does not always exist (Marcus Fischer, 2010).

groupadd <group>

A.4 Filesystem in Userspace (FUSE)

The abbreviation FUSE stands for Filesystem in userspace. It is a kernel module. It provides a virtual file system for unprivileged users. A FUSE file system should not be confused with a VFS (Virtual File System). VFS systems contain all file systems contrary to FUSE. FUSE file system is not involved in data storage.

A FUSE file system represents a level of abstraction. A program that uses FUSE can access a FUSE file system with the normal syscalls (such as read (), write (), close (), or open ()), just like any other. In the background, however, these accesses are redirected. Other, separate actions will be carried out, of which the user and the accessing program will not notice (Johannes Plötner and Steffen Wendzel, 2012).

B Virtual flash drive

```
#!/bin/bash
#
#########################################################
#                                                       #
#         script to make a virtual flash drive          #
#                                                       #
#             This script shoud be run as root          #
#                                                       #
#                   from Horst Dumstorff                #
#                                                       #
#                                                       #
# to check the real size of the image:                  #
# $ du -sh ${IMG}                                       #
#                                                       #
# to check the virtual size of the image:               #
# $ du -sh --apparent-size ${IMG}                       #
#                                                       #
# to copy the image file in real size                   #
# $ cp -av --sparse=always image.img aim_image.img      #
#                                                       #
# #######################################################

# input image
IMG="virtual_USB.img"

echo -e "\n -> img name:"$IMG" created"

# label of the USB flash drive
```

```
if [ $1 != '' ]; then
  LABEL=$1
else
  LABEL="Virtual_USB" # if not named
fi

echo -e "\n -> label:"$LABEL" created\n"

# create image file
# (https://www.linuxnix.com/what-you-should-know-
about-linux-dd-command/)
dd if=/dev/zero of=${IMG} bs=1 count=0 seek=64G
echo -e "\n -> img file created"

# manipulating partition tables
# (https://www.gnu.org/software/parted/manual/parted.html)
parted --script --machine ${IMG} mklabel msdos mkpart
primary ntfs 0% 100%
echo -e "\n -> parted done"

# create loop device
# (https://www.systutorials.com/docs/linux/man/8-losetup/)
LOOP=$( losetup --partscan --show --find ${IMG} )
echo -e "\n -> LOOP var created"

# create an NTFS file system
# (https://linux.die.net/man/8/mkfs.ntfs)
mkfs.ntfs -f -Q -L ${LABEL} ${LOOP} -p1
echo -e "\n -> NTFS file system created"

# delete loop device
# (https://www.systutorials.com/docs/linux/man/8-losetup/)
losetup -d ${LOOP}
echo -e "\n -> loop device deleted"
```

C Application ”Virtual PC”

Virtual PC documentation

API Documentation

May 17, 2019

Contents

1 Module virtual_pc

1.1 Functions

vp_start_gui()

Starting point when module is the main routine.

create_New_Toplevel(*root*, **args*, ***kwargs*)

Starting point when module is imported by another program.

destroy_New_Toplevel()

1.2 Variables

Name	Description
py3	**Value:** `True`
w	**Value:** `None`

1.3 Class New_Toplevel

1.3.1 Methods

___init___(*self*, *top*=`None`)

get_VIRTUAL_METHOD(*self*)

get_FILE_NAME(*self*)

get_FILE_DIR(*self*)

set_TEMP_DIR(*self*, *tempdir*)

get_TEMP_DIR(*self*)

get_USB_IMAGE(*self*)

get_virtual_USB(*self*)

get_virtual_USB_drive(*self*)

set_choosen_drive(*self*, *listelement*)

get_choosen_drive(*self*)

set_selected_ListElement(*self*)

call_cmd(*self*, *cmd*)

call_cmd_format1(*self*, *cmd*, *strg*=`None`, *replace*=`None`)

call_cmd_format2(*self*, *cmd*, *strg*=`None`, *replace*=`None`)

get_unmounted_drives(*self*)

set_vitualPCConfigXML(*self*, *vitualPCConfigXML*)

get_vitualPCConfigXML(*self*)

1.4 Class chooseDeviceDialog

```
tkinter.simpledialog.Dialog ─┐
                             virtual_pc.chooseDeviceDialog
```

1.4.1 Methods

__init__(*self*, *parent*, *Liste*, *Toplevel*)

buttonbox(*self*)

body(*self*, *master*)

populate(*self*)

ok(*self*, *event*=`None`)

apply(*self*)

get_selected_RB(*self*)

onFrameConfigure(*self*, *event*)

Reset the scroll region to encompass the inner frame

set__Liste(*self*, *Liste*)

get__Liste(*self*)

2 Module virtual_pc_config

2.1 Class ConfigDialog

tkinter.simpledialog.Dialog ─┐
virtual_pc_config.ConfigDialog

2.1.1 Methods

buttonbox(*self*)

__init__(*self*, *master*, *gui*, *state*, *file*=`None`)

body(*self*, *master*)

This class configures and populates the toplevel window. top is the toplevel containing window.

apply(*self*)

cancel(*self*)

ok(*self*)

writetoxml(*self*)

readfromxml(*self*, *filename*)

c_load_file_evidence(*self*)

c_load_file_medium(*self*)

2.1.2 Class Variables

Name	Description
NEW	**Value:** `"new"`
OPEN	**Value:** `"open"`
SAVE	**Value:** `"save"`
SAVEAS	**Value:** `"saveas"`

3 Module virtual_pc_config_dialog_support

3.1 Functions

c_load_file_medium()

3.2 Variables

Name	Description
py3	**Value:** `False`
__package__	**Value:** `None`

4 Module virtual_pc_support

4.1 Functions

set_Tk_var()

get_RB_Method()

append_Storagelist(*liste*, *Liste*)

get_Storagelist()

B_Choose_Storagedrive()

set_Evidence_File_in_List()

set_Mediume_File_in_List()

set_List_Element_VM_Medium()

get_Evidencelist()

get_Storagemediumlist()

B_Choose_Evidencelist()

RB_Select_Virtual()

RB_Select_mount()

event_Info()

call_cmd(*cmd*, *a*=`None`, *b*=`None`)

mount_storage_device(*dd_path*=`None`)

umount_storage_device()

xmount_evidence_image(*xcache*, *xvol*, *ximage*, *temp_dir*)

unmount_evidence_image()

mount_evidence_device()

start_qemu(*parameter*, *sb_usb*, *addpara*=`None`)

set_mount_button_ENABLED()

set_mount_button_DISABLED()

mount_virtual_usb()

is_virtual_usb()

unmount_virtual_usb()

virtualize_image(*usb_drive*)

event_OK()

event_new()

event_open()

event_quit()

event_save()

event_save_as()

init(*top*, *gui*, **args*, ***kwargs*)

destroy_window()

5 Module virtual_pc_xml

5.1 Variables

Name	Description
__package__	**Value:** `None`

5.2 Class VitualPC_Config_XML

5.2.1 Methods

__init__(*self*)

setProject_evidence(*self, project_evidence*)

setProjectname(*self, projectname*)

setProjectart(*self, projectart*)

setProjectdir(*self, projectdir*)

setEvidencefile(*self, evidencefile*)

setVm(*self, vm*)

setVm_parameter(*self, vm_parameter*)

setVm_add_parameter(*self, vm_add_parameter*)

setMedium(*self, medium*)

setMedium_name(*self, medium_name*)

setMedium_art(*self, medium_art*)

setMedium_dir(*self, medium_dir*)

setMedium_file(*self, medium_file*)

getProject_evidence(*self*)

getProjectname(*self*)

getProjectart(*self*)

getProjectdir(*self*)

getEvidencefile(*self*)

getVm(*self*)

getVm_parameter(*self*)

getVm_add_parameter(*self*)

getMedium(*self*)

getMedium_name(*self*)

getMedium_art(*self*)

getMedium_dir(*self*)

getMedium_file(*self*)

__str__(*self*)

5.3 Class VitualPC_XML

5.3.1 Methods

writeXML(*configfile, filename*)

readXML(*filename*)

Index

11