Manuscript 1680

# Forensic Analysis of VirtualBox Application Artifacts

Author #1

## 1.0 INTRODUCTION

Virtualization is the process of dividing up physical hardware into separate virtual pieces, or "the process of creating a virtual representation of something, such as virtual applications, servers, storage, and networks"[15]. We focus on a particular virtualization technique: A Virtual Machine, (VM). A VM uses physical resources from its host acting as a separate entity. To cut costs, more companies are using virtualization technology as "it is the single most effective way to reduce IT expenses while boosting efficiency and agility for all size businesses" [15]. This is indicated by market growth in this sector. It is estimated that by 2023 the Server Virtualization Market will rise to $8 billion- a 7% increase from 2016 [6]. Since the enterprise sphere is shifting towards virtualization, so is the consumer market.

Our focus is Oracle's VirtualBox- a free, open-source, cross-platform virtualization application. VirtualBox allows a user to run multiple operating systems on a single Operating System (OS) by hosting VMs through a type II hypervisor [12]. The trend of the consumer market, the simplicity, and the wide availability of VirtualBox create a concern about the weaponization of the application to commit cyberattacks and cybercrime. This creates a question: How can a forensic investigator investigate a VirtualBox VM?

### 1.1 Purpose

Our purpose is to detail VirtualBox forensic artifacts related to the VM's resources and internal storage.

### 1.2 Background

Some publications deal with the general problem of investigating virtual machines. However, an exhaustive search of information published by Oracle or the academic community revealed no information about the VirtualBox Virtual Disk Image (VDI), the log structure, or directory structure. In this section, we review current material.

Diane Barrett and Gregory Kipper's Virtualization and Forensics A Digital Forensics Investigator's Guide to Virtual Environments provides an overview of how to investigate a virtual environment and the dynamic elements of a virtual environment. It covers VMware Server and Workstation, Microsoft Virtual PC, MojoPac, MokaFice, and others [1]. While this book does cover Sun VirtualBox, the predecessor to contemporary VirtualBox, it misses some forensics artifacts of VirtualBox. This isn't surprising as the book covers a wide range of VMs and is now over ten years old.

The main documentation found from Oracle is the VirtualBox User's Manual [12]. This is written for users (e.g. specifying how to create a VM and what settings may be appropriate) so it does not provide information about artifacts that would be useful to investigators and responders.

VirtualBox supports a variety of disk image formats. We investigate the file format of a VDI. There are other image file formats that investigators and responders may encounter. These are proprietary and it may be necessary to consult with the owning company in an investigation.

Microsoft has a virtual image format, the Virtual Hard Disk (VHD), described in "Virtual Hard Disk Image Format Specification" [7]. The specification describes the metadata and how this format stores data. This article is indispensable when handling VHD images.

VMware's virtual disk image (vmdk) is outlined in the technical note "Virtual Disk Format 5" [14]. The popularity of VMware solutions makes this valuable documentation.

"Virtual machine forensic analysis and recovery method for recovery and analysis of digital evidence" covers a forensic acquisition of a VirtualBox VM [16]. However, that paper uses the vmdk image format and not the VDI format. An interesting part of [16] is the explanation that if a VM is destroyed it is nonrecoverable.

Some forensics artifacts of VirtualBox are described in "Forensic Recovery of Evidence from Deleted Oracle VirtualBox Virtual Machines" by Cherilyn Neal at Utica College [8]. Three VM deletion scenarios are outlined and forensically analyzed. These scenarios are (1) the removal of a VM from the VirtualBox Manager, (2) reverting a VM to a previously saved state (Snapshot), and (3) the deletion of a VM. While [8] focuses on analyzing the physical, host machine we focus on analyzing the guest VM only. With this lens, we can confirm Neal's findings and more importantly find additional artifacts by isolating the guest OSs' resources. These include VirtualBox.log files, Selectorwindow.log files, Vbox.log files, complications with forensic tools, the VDI file format, and the interactions of snapshots.

"Design of Tool for Digital Forensics in Virtual Environment" argues for a forensic tool that could be used on VirtualBox to find evidence [13]. However, [13] does not address how to investigate a VirtualBox VM.

The description of the VDI image file in the 2008 VirtualBox Forum post [10] is out of date, however, we validate what is still current. See section 2.3.

### 1.3 Methodology, Scope, and Testing Equipment

Our methodology consisted of creating and running a Kali VM in the test environment. The test environment was then analyzed for activity related to inside the virtual environment. This is unlike previous studies as we did not focus on forensic artifacts from the test environment, e.g. Windows 10, but the VirtualBox application and the VM's resources themselves.

The scope consists of the various VirtualBox logs, the program's directory structure, snapshots, and the Virtual Disk Image file. Other logs that are not directly created by Virtualbox, such as Window's Event Logs, are outside the scope of this paper. These logs are detailed in various other digital forensic papers, but they do not directly log the events that occur within a VM.

The test environment included a Windows 10 desktop running various versions of VirtualBox. The Windows environment was chosen as it is the most likely OS to run VirtualBox. VirtualBox version 5.0-6.1 were examined. All these versions contained the same or similar structure that is documented in this paper.

Standard MD5sum hashing was used before and after any tests to ensure integrity. Well-known forensics tools were used: Access Data's Forensic Toolkit (FTK), FTK imager, LSoft's Active@ Disk Editor, Guidance Software's EnCase, X-ways, and HxD.

## 2.0 RESULTS

The VirtualBox application was then installed onto the test environment. A Kali Linux machine was then built as a test machine. Then, a couple of simple text documents and images were created as mock evidence. Multiple machines were created to isolate each new interest. This section contains descriptions of artifacts that were found. These artifacts are of interest to investigators and responders.

### 2.1 The Directory Structure of VirtualBox

When a user downloads and installs VirtualBox, a series of directories in the host machine are generated. These directories contain forensic artifacts about the VirtualBox interactions and the virtual environment. Three different directories are of interest. see Figure 1. (Note the directory locations in Figure 1 are the Windows defaults).

1. C:\Program Files\Oracle\VirtualBox
2. C:\Users\<USER>\.VirtualBox
3. C:\Users\<USER>\VirtualBox VMs

***Figure 1*** **VirtualBox Directories in Host**

The directory structure goes from high-level, environment-based artifacts to low-level, VM-based artifacts.

The first directory, C:\Program Files\Oracle\VirtualBox, are .dll files, drivers, licenses, templates, etc. These are the necessary files required to run VirtualBox.exe. The data gathered here does not pertain to a specific VM. However, the .dll files and VirtualBox.exe could be targets in a malware campaign. This area of the host OS has been targeted for dll injection attacks and other malware utilizing VirtualBox to avoid antivirus detection [2]. This type of malware starts up a VirtualBox instance to run the rest of the code inside the virtual environment to avoid detection. It then downloads another payload while maintaining secrecy. Because of this threat vector, this directory can be a target and of interest to incident responders.

The second directory, C:\Users\<USER>\.VirtualBox, there are log files regarding the virtual environment and the VMs. These files most frequently found are named “selectorwindow.txt”, “selectorwindow.log.#”, “vbox-ssl-cacertificate”, “VBoxSVC.txt”, “VirtualBox.xml”, and “VirtualBox.xml-prev” (where # is a number)

VirtualBox logs follow a proprietary structure. Each line that VirtualBox records, starts with a timestamp. The timestamp format is “HH:MM:SS.ms and is relative to the start of the vm”. However, “if [the user] prefer[s] to also have ‘wall clock time’ [the user] could set the environment variable before starting the VM” [3]. Assuming no alteration to the time variable, the logs and their timestamps could reconstruct events dealing with the host and guest operating systems. The log naming scheme is also explained in [3]: “the log files are rotated such that the most recent is always called vbox.log and the older ones are vbox.log[123]” Note: Due to this naming scheme, the most recent log file will be duplicated between vbox.log and the highest number of the vbox.log[#]. We also found that the naming scheme is also prevalent in all logs by VirtualBox, not just the vbox.log ones.

The selectorwindow.txt log files record information about the different tools found within VirtualBox. When a virtual environment’s network or media setting is modified, the event is recorded by these logs. These tools are found under the file tab of the VirtualBox Graphical User Interface (GUI) (Figure 2):

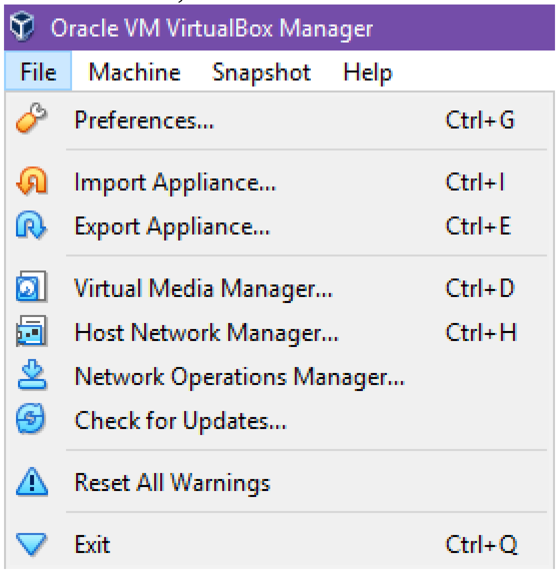
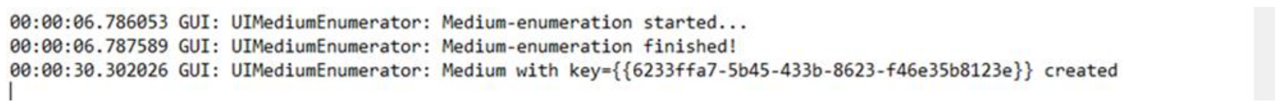

Oracle VM VirtualBox Manager
File
Machine
Snapshot
Help
Preferences... Ctrl+G
Import Appliance... Ctrl+I
Export Appliance... Ctrl+E
Virtual Media Manager... Ctrl+D
Host Network Manager... Ctrl+H
Network Operations Manager...
Check for Updates...
Reset All Warnings
Exit Ctrl+Q

***Figure 2*** **VirtualBox Selection of Tool**

The Virtual Media Manager allows for the conversion and handling of the different virtual images. The Host Network Manager is used to build the network between the VM(s) and the host. The Network Operations Manager- Focuses on the different networks that are currently running.

For example, selectorwindow.log can reveal an image/medium creation. See Figure 3. The timestamp shows this event occurred about 30 seconds after the start-up of VirtualBox.exe. The same line shows a UUID for the medium created. In this case, it was imported using the Virtual Media Manager under the name GUI: UIMediumEnumerator. This medium can now be used when creating a new VM. The key could then be traced to the corresponding VM’s XML file.

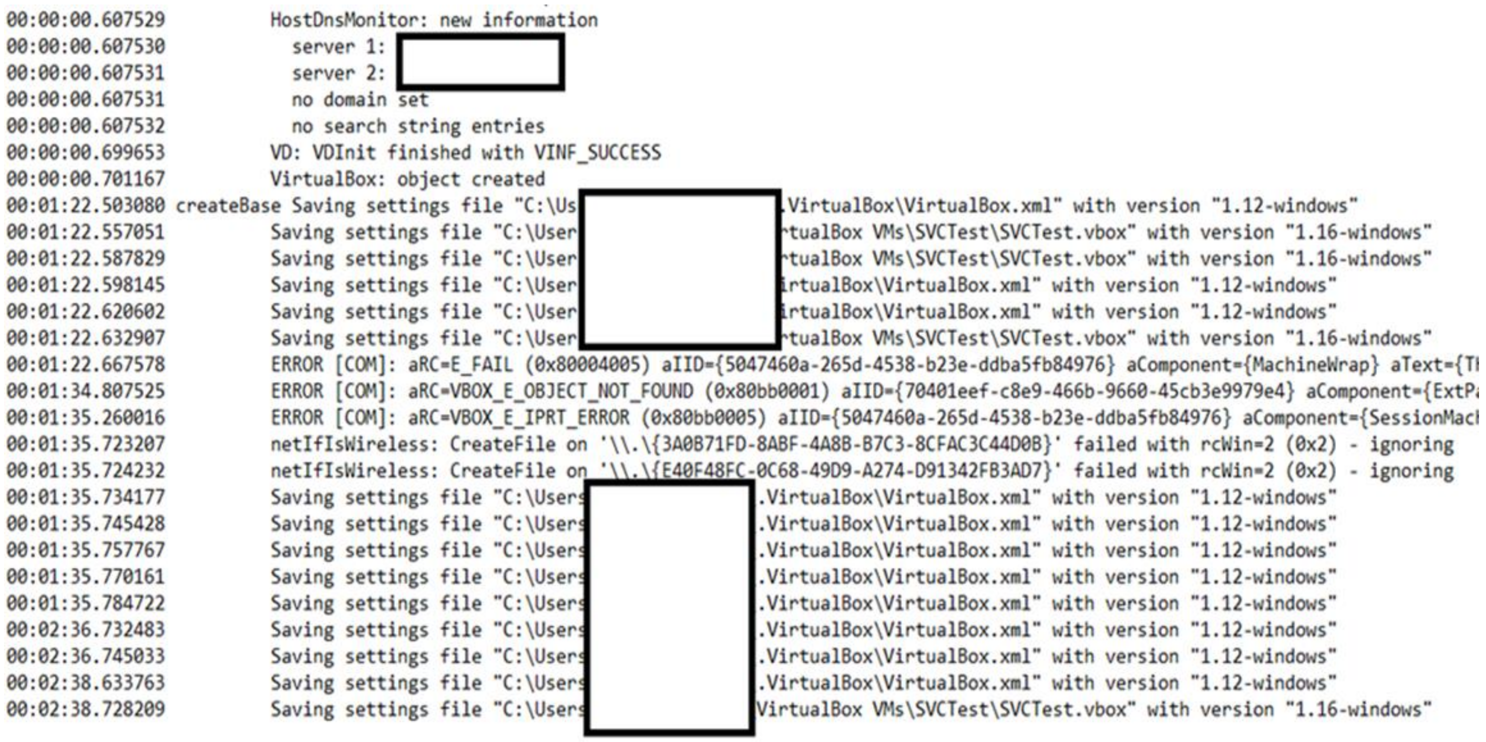


```
00:00:06.786053 GUI: UIMediumEnumerator: Medium-enumeration started...
00:00:06.787589 GUI: UIMediumEnumerator: Medium-enumeration finished!
00:00:30.302026 GUI: UIMediumEnumerator: Medium with key={{6233ffa7-5b45-433b-8623-f46e35b8123e}} created
|
```

***Figure 3*** **Selectorwindow.log noting the creation of a new image file**

The C:\Users\<USER>\.VirtualBox directory also contains the Thawte security certificate. This file is named “vbox-ssl-cacertificate.crt” by default. This certificate maintains the integrity of the VirtualBox installation and download [4]. If the integrity of VirtualBox is in question, this should be checked.

Another log of forensic importance is the VBoxSVC.log. This log records the virtualization events of VirtualBox. VBoxSVC stands for VirtualBox Service Process [13]. Anything VirtualBox does as a service is recorded in this log. This can include Host Information, Host Process ID, VirtualBox version, Host OS information (e.g. Windows service packs), location of the home directory for the log (\.VirtualBox by default), DNS/Networking Information, creation of VM, and when VM settings files are being updated. To demonstrate, when a new VM was created called “SVCTest”. VBoxSVC.log recorded the creation along with host information. See Figure 4.



```
00:00:00.607529          HostDnsMonitor: new information
00:00:00.607530            server 1: [redacted]
00:00:00.607531            server 2:
00:00:00.607531            no domain set
00:00:00.607532            no search string entries
00:00:00.699653          VD: VDInit finished with VINF_SUCCESS
00:00:00.701167          VirtualBox: object created
00:01:22.503080 createBase Saving settings file "C:\Us[redacted].VirtualBox\VirtualBox.xml" with version "1.12-windows"
00:01:22.557051          Saving settings file "C:\User          rtualBox VMs\SVCTest\SVCTest.vbox" with version "1.16-windows"
00:01:22.587829          Saving settings file "C:\User          rtualBox VMs\SVCTest\SVCTest.vbox" with version "1.16-windows"
00:01:22.598145          Saving settings file "C:\User          irtualBox\VirtualBox.xml" with version "1.12-windows"
00:01:22.620602          Saving settings file "C:\User          irtualBox\VirtualBox.xml" with version "1.12-windows"
00:01:22.632907          Saving settings file "C:\User          rtualBox VMs\SVCTest\SVCTest.vbox" with version "1.16-windows"
00:01:22.667578          ERROR [COM]: aRC=E_FAIL (0x80004005) aIID={5047460a-265d-4538-b23e-ddba5fb84976} aComponent={MachineWrap} aText={Tl
00:01:34.807525          ERROR [COM]: aRC=VBOX_E_OBJECT_NOT_FOUND (0x80bb0001) aIID={70401eef-c8e9-466b-9660-45cb3e9979e4} aComponent={ExtPa
00:01:35.260016          ERROR [COM]: aRC=VBOX_E_IPRT_ERROR (0x80bb0005) aIID={5047460a-265d-4538-b23e-ddba5fb84976} aComponent={SessionMacl
00:01:35.723207          netIfIsWireless: CreateFile on '\\.\{3A0B71FD-8ABF-4A8B-B7C3-8CFAC3C44D0B}' failed with rcWin=2 (0x2) - ignoring
00:01:35.724232          netIfIsWireless: CreateFile on '\\.\{E40F48FC-0C68-49D9-A274-D91342FB3AD7}' failed with rcWin=2 (0x2) - ignoring
00:01:35.734177          Saving settings file "C:\Users[redacted].VirtualBox\VirtualBox.xml" with version "1.12-windows"
00:01:35.745428          Saving settings file "C:\Users          .VirtualBox\VirtualBox.xml" with version "1.12-windows"
00:01:35.757767          Saving settings file "C:\Users          .VirtualBox\VirtualBox.xml" with version "1.12-windows"
00:01:35.770161          Saving settings file "C:\Users          .VirtualBox\VirtualBox.xml" with version "1.12-windows"
00:01:35.784722          Saving settings file "C:\Users          .VirtualBox\VirtualBox.xml" with version "1.12-windows"
00:02:36.732483          Saving settings file "C:\Users          .VirtualBox\VirtualBox.xml" with version "1.12-windows"
00:02:36.745033          Saving settings file "C:\Users          .VirtualBox\VirtualBox.xml" with version "1.12-windows"
00:02:38.633763          Saving settings file "C:\Users          .VirtualBox\VirtualBox.xml" with version "1.12-windows"
00:02:38.728209          Saving settings file "C:\Users          VirtualBox VMs\SVCTest\SVCTest.vbox" with version "1.16-windows"
```

***Figure 4*** **VBoxSVC.log displaying the creation of a new VM**

This information is can be important to a forensic investigator. For example, if the VM in Figure 4 then targeted and attacked a corporation, an investigator would be able to tie the VM to the Host OS. Additionally, they could build this log into his/her super timeline.

The final pair of files in the \.VirtualBox Directory are VirtualBox.xml and VirtualBox.xml-prev. These XML files are the settings for the VirtualBox application itself (as opposed to the Virtual Machines). The VirtualBox.xml document was found to be the current settings and the VirtualBox.xml-prev is previous. This confirms “[the xml-prev] file is the backup file for the associated VM…” [8]. These two settings files are imperative to use in an investigation because they hold key configuration data of the specific VM(s). The XML files contain the following:

•Setup information for VirtualBox.exe program

•Recent Virtual Hardware- Most recent Snapshot, Most recent Virtual CD.

•Network Information- Both Virtual Network Card and Public IP address in IPv4 and IPv6, and the relation between the host and guest, e.g. Host-only, Bridge Adapter, Network Address Translation (NAT).

•Dynamic Host Configuration Protocol (DHCP) information (if any)

•Machine Registry- List of VMs with their Universally Unique Identifier (UUID) and file path to the machine-specific .vbox file

•Location of the VM-specific files. By default, labeled "VirtualBox VMs"

The most important piece of information within the XML files is the list of VMs it provides, called the Machine Registry. For each VM created with the VirtualBox program, there is a machine registry entry. Each entry contains the UUID, which is used to call each machine, and the file path for the VM's specific files.

Following this path, an investigator will see a list of directories corresponding to the name of each VM. These should match each VM in the Machine Registry. If a VM listed in the registry is not found it can be considered a missing VM and should be located immediately (Note: the path in the Machine Registry is set during VM creation. If those files are moved elsewhere, the file path does not adjust unless rerouted within VirtualBox).

In C:\Users\<USER>\VirtualBox VMs\<VM NAME> (or following the Machine Registry path from VirtualBox.xml) are the .vbox files which are the configuration files for the VM. They are similar in structure to the VirtualBox.xml file. These files include useful information: snapshot(s), network configuration, virtual image location, USB connections, and the other physical and virtual devices connected to the VM.

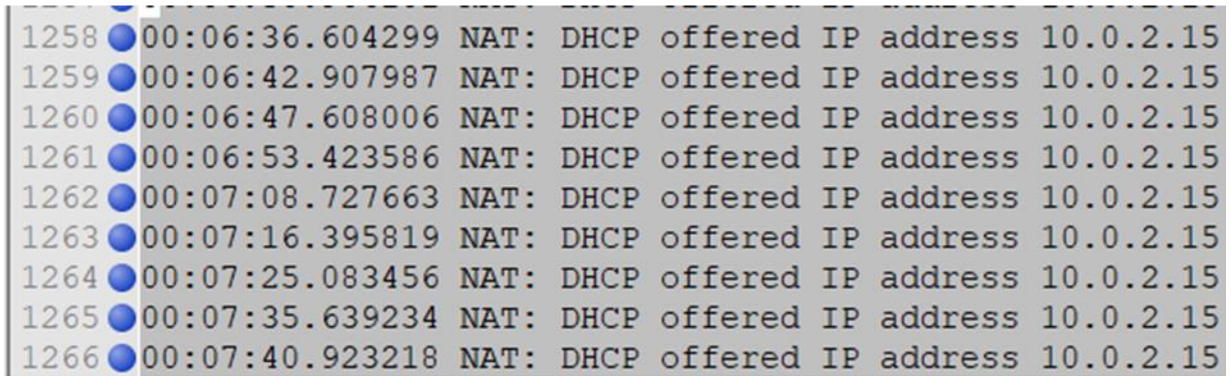
There are also VM-specific logs found within the log directory of the specific VM. There are two files called "Vbox.txt" and "VboxHardening.txt". These logs are specific to a VM and follow the outline listed below, obtained from [3]. See Figure 5.

***Figure 5*  Outline of Vbox.txt/VM-Specific Logs [3]**

| **Section** | **Content** | **Starts around** |
|---|---|---|
| **Header** | VirtualBox version, Host OS information and Host Hardware information | Beginning of file |
| **Guest's Configuration (CFGM) Dump** | A listing of the configuration information of guest (guest virtual hardware) | *** CFGM dump *** |
| **Host Information (CPUID dump)** | Low-level CPU information of Host and what will be reflected to Guest | *** CPUID dump *** |
| **Creating the VM** | Information about the creation of the virtual machine environment | *** End of CPUID dump *** |
| **Powering on or Loading from saved state** | When powering on you'll see very little in the logs at this stage.  If loading from a saved state, information from the Saved State Manager about matching previous state to current environment. | Changing the VM state from 'CREATED' to 'LOADING' |
| **Resuming/Running** | Once the saved state is loaded, or the vm is booted, the guest code is executed. | Changing the VM state from 'POWERING_ON' to 'RUNNING' or from 'LOADING' to 'RESUMING' or from 'SUSPENDED' to 'RESUMING' |
| **Guest lifetime** | This part of the log contains entries concerning the lifetime of the Guest. | Changing the VM state from 'RESUMING' to 'RUNNING' |
| **Powering off or suspending** | A dump of the guest state at the time the vm was powered off. | *** Guest state at power off *** |
| **Statistics** | The statistics collected during the session are dumped out. | *** Statistics *** |

Using Figure 5, an investigator can determine when VM's were started, what resources each VM accessed, host information, and how long each VM was running for.

We found that networking information is also stored in the Vbox.txt and the VboxHardening.txt logs.  This relationship is based on the network topology chosen for the VM and the host machine in VirtualBox settings.  An example of a NAT typology between the VM and the Host is displayed in Figure 6.



```
1258 ●00:06:36.604299 NAT: DHCP offered IP address 10.0.2.15
1259 ●00:06:42.907987 NAT: DHCP offered IP address 10.0.2.15
1260 ●00:06:47.608006 NAT: DHCP offered IP address 10.0.2.15
1261 ●00:06:53.423586 NAT: DHCP offered IP address 10.0.2.15
1262 ●00:07:08.727663 NAT: DHCP offered IP address 10.0.2.15
1263 ●00:07:16.395819 NAT: DHCP offered IP address 10.0.2.15
1264 ●00:07:25.083456 NAT: DHCP offered IP address 10.0.2.15
1265 ●00:07:35.639234 NAT: DHCP offered IP address 10.0.2.15
1266 ●00:07:40.923218 NAT: DHCP offered IP address 10.0.2.15
```

***Figure 6*** **Vbox.txt Network Typology Logging**

Finally, there is the image file or the virtual disk for the VM. By default, this format is in the Virtual Disk Image (VDI) format. Image files are discussed in the next section.

### 2.2 VirtualBox Supported Images

VirtualBox supports 4 different virtual disk image formats [12]:

•VDI- Virtual Disk Image, Oracle's proprietary image file, the default

•VHD- Virtual Hard Disk, Microsoft's proprietary image file

•VMDK- Virtual Machine Disk, VMware's proprietary image file

•HDD- Image files of Parallels version 2 (HDD format)

In an investigation, the image file format determines the tools that may be used with the file. Furthermore, the metadata in the image file formats varies. Although the image file may be investigated by mounting it, there are reasons not to take this approach. First, in this approach, the investigator will miss important metadata within the header of the image file. It also may contaminate the evidence. A direct approach avoids these problems. A direct approach requires documentation of the disk image format. This is provided for VDI files in the next section as VDI is the default (and probably most common) format for VirtualBox VMs.

### 2.3 Outline of Virtual Disk Image Format

The first sector (512 bytes) of the VDI file is the most important. See Figure 7. Note: VDI Header values are little-endian encoded [10].

***Figure 7* Data Offset Table for Oracle VDI files [10]**

| Offset | Description | Notes |
|---|---|---|
| [0x00-0x27] | The header in ASCII. Typically, "<<<Oracle VM VirtualBox Disk Image>>>" | This depends on which version of VirtualBox. Older versions tend to use <<Sun VirtualBox Disk Image>> |
| [0x40-0x43] | "0x 7F 10 DA BE" | File Signature |
| [0x44-0x47] | Version Number. 01 00 01 00 is Version 1.1 | It is the version of VDI. Version 1.1 as of writing. |
| [0x48-0x4B] | Header Size: 0x190 | Size of Header. 0x190 is the default. |
| [0x4C-0x4F] | Image Type | 1=Dynamic<br>2=Static<br>4=Snapshot<br>Note: 3 is unknown |
| [0x50-0x53] | Image Flags | |
| [0x54-0x153] | Image Description | |
| [0x154-0x157] | Offset Blocks | Number of Blocks to start of Virtual Disk |
| [0x158-0x15B] | Offset Data | Offset to Start of Virtual Disk<br>Default is 00 00 20 00 |
| [0x168-0x16B] | Sector Size | By default, 00 02 00 00 = 512 bytes |
| [0x170-0x177] | Disk Size (in bytes) | |
| [0x178-17B] | Block Size | |
| [0x17C-17F] | Block Extra Data | By default, zero |
| [0x180-0x183] | Blocks in HDD | Number of Blocks in the VDI. |
| [0x184-0x187] | Allocated Blocks | Total blocks have data in them. Static should equal Blocks in HDD number. Dynamic will be different. |
| [0x188-0x197] | UUID of Self | The UUID of the VDI itself |
| [0x198-0x1A7] | UUID of the last Snapshot | |

*Figure 7* **Data Offset Table for Oracle VDI files [10]**

| | | |
|---|---|---|
| [0x1A8-0x1B7] | UUID Link | UUID of VDI File before in the Chain |
| [0x1B8-0x1C7] | UUID Parent | |
| Jump to 0x200000 | | |
| [Value in Offset Data] | Data | The disk itself starts here with the MBR or GPT (depending on the partitioning scheme) |

Like most files, a VDI file has a signature. However, the signature of a VDI file is not located in the first few bytes but at offsets 0x40 (four bytes). Preceding the signature (starting at offset 0) is an ASCII header. The contents of the header vary and should be noted. However, the actual signature (as 0x40 to 0x43) does not vary. The most important artifacts here are the UUIDs as they relate to snapshots (detailed in section 2.5) and the start of the virtual disk is found at the offset data value in the header, [0x158-0x15B], for VirtualBox VDI files.

## 2.4 Snapshots

Snapshots are where "you can save a particular state of a virtual machine for later use" [12]. This enables system administrators to regulate machine settings across an enterprise. However, snapshots could also be used to attempt to hide evidence. For example, an individual could have a collection of child exploitation images on a VirtualBox VM. Reverting to a clean snapshot then could hide the existence of the child exploitation images on the VM's disk image. Therefore, investigating snapshots is imperative. The snapshots are stored under the VM-specific directory in a subdirectory called "Snapshots". Within the Snapshots directory, there will be two files for each snapshot that was taken: a .sav file and a .vdi file. Here the snapshot VDI file's filename is a UUID. The same key is referenced in the Vbox, machine-specific, files. Any reference to this UUID refers to this snapshot. We found that whenever a VM is reverted to a previous snapshot or a snapshot is taken that the UUIDs will change, see section 2.5. The .sav file's filename is a timestamp in the Coordinated Universal Time (UTC). It has the following format:"YYYY-MM-DDTH-MM-SS-MsMsMsMsMsMsMsMsMsZ.sav". Note: "T" is signaling the switch between the date and the time.

We found that both the VDI and .sav files contain information about the directory structure and data from the VM's filesystem. We found that manual forensics is a way to extract evidence from the snapshot files as we were unable to mount or convert the snapshot's VDI for analysis. This is consistent with Neal's finding that a VM cannot be generated using only a snapshot's VDI file ([8]). Both the snapshot and the machine's main VDI file are required for this method.

## 2.5 Snapshot Universally Unique Identification Numbers (UUID)

The most important piece of data within the VDI format is the UUID tree generated. Within each VDI file, in the header, there is a hexadecimal value that is used to identify relations between the VM's main VDI disk and other VDI files e.g. Snapshots. This can be found in the table in Figure 7. These UUIDs create a VDI chain that can be used to piece together which snapshot came first and which one comes next. A simple example of this can be shown in Figure 8.

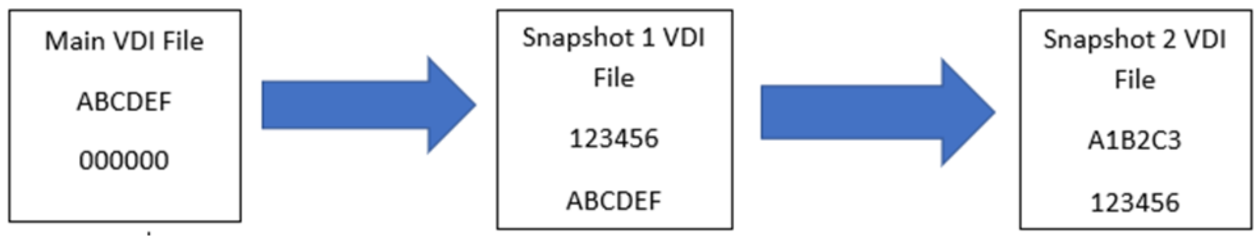


***Figure 8*** **Snapshot Relation with UUIDs**

Additionally, this use of UUIDs is applied to multiple paths. See Figure 9.

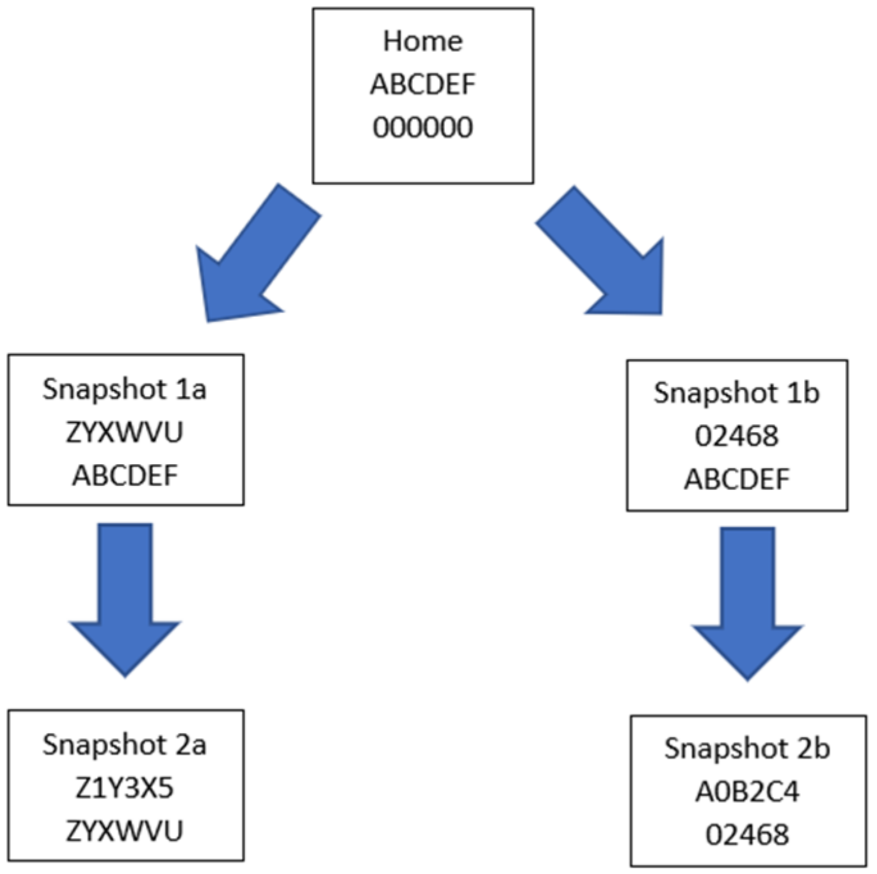


***Figure 9*** **Snapshot Tree Relations with UUIDs**

The "home" also known as the main VDI file will only have one UUID- the reference to itself. The other UUID entries are null. This file will also be a machine VDI and not a snapshot. The second signature is the link UUID connecting other VDI files based on their UUIDs.

The snapshot tree can be used by a forensic investigator to determine the relations of each VDI file. Then, a timeline can be built with the host filesystem timestamps, .sav file timestamps, and log timestamps to make a super timeline. Additionally, we found that the UUIDs continue to change IDs with each reversion of a snapshot. Building a tree during dead analysis will determine these snapshot relations.

## 3. EFFECTS ON FORENSICS WITH VDI FORMAT

The VDI file is a key artifact for a forensic investigator as it contains the VM filesystem. Because of this, we found exporting any VM image, VDI format or not, and analyzing it separate from its host is most effective. This allows forensic tools and investigators from mistaking the physical host's filesystem and the guest's filesystem.

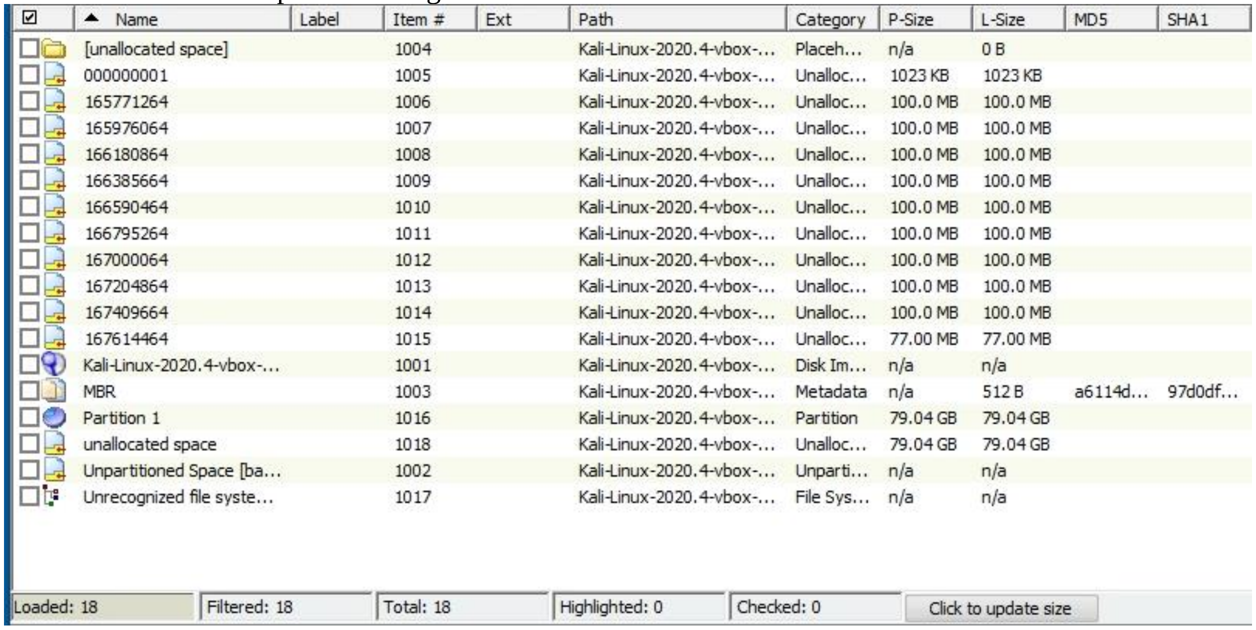
### 3.1 Forensic Problems with VDI Files

As critical as a VDI file is to a forensic investigator, we found many problems analyzing the format.

A VDI file may be mounted to do a forensic analysis of a VM ([8]). This approach requires an uncorrupted VDI file and will modify evidence. We recommend creating at least a working copy if utilizing this method.

After exporting, the investigator also must determine if a tool interprets the image as a file or a filesystem. For example, Active@Disk Editor treats a VDI file as a file, not a filesystem. In contrast, X-ways interprets a VDI file as a filesystem. This occurs because VM disks use dynamic space allocation by default, not static space allocation. For example, a tool that interprets a VDI as a filesystem will report a 128GB disk and filesystem while a tool that cannot, will report only a 13 GB file.

Furthermore, we found that multiple industry-standard forensic suits had issues importing the VDI file. We found that AccessData's Forensic Toolkit (FTK) can add VDI as a type of evidence, but the VM's filesystem will not be parsed. FTK does interpret parts of the VDI file such as the Master Boot Record or determines a partition, but the VM's filesystem is not interpreted. This can be seen in figure 10. Similarly, EnCase mounts a VDI as a raw image, however, it is called "unused disk space area".



| ☑ | Name | Label | Item # | Ext | Path | Category | P-Size | L-Size | MD5 | SHA1 |
|---|---|---|---|---|---|---|---|---|---|---|
| ☐ | [unallocated space] | | 1004 | | Kali-Linux-2020.4-vbox-... | Placeh... | n/a | 0 B | | |
| ☐ | 000000001 | | 1005 | | Kali-Linux-2020.4-vbox-... | Unalloc... | 1023 KB | 1023 KB | | |
| ☐ | 165771264 | | 1006 | | Kali-Linux-2020.4-vbox-... | Unalloc... | 100.0 MB | 100.0 MB | | |
| ☐ | 165976064 | | 1007 | | Kali-Linux-2020.4-vbox-... | Unalloc... | 100.0 MB | 100.0 MB | | |
| ☐ | 166180864 | | 1008 | | Kali-Linux-2020.4-vbox-... | Unalloc... | 100.0 MB | 100.0 MB | | |
| ☐ | 166385664 | | 1009 | | Kali-Linux-2020.4-vbox-... | Unalloc... | 100.0 MB | 100.0 MB | | |
| ☐ | 166590464 | | 1010 | | Kali-Linux-2020.4-vbox-... | Unalloc... | 100.0 MB | 100.0 MB | | |
| ☐ | 166795264 | | 1011 | | Kali-Linux-2020.4-vbox-... | Unalloc... | 100.0 MB | 100.0 MB | | |
| ☐ | 167000064 | | 1012 | | Kali-Linux-2020.4-vbox-... | Unalloc... | 100.0 MB | 100.0 MB | | |
| ☐ | 167204864 | | 1013 | | Kali-Linux-2020.4-vbox-... | Unalloc... | 100.0 MB | 100.0 MB | | |
| ☐ | 167409664 | | 1014 | | Kali-Linux-2020.4-vbox-... | Unalloc... | 100.0 MB | 100.0 MB | | |
| ☐ | 167614464 | | 1015 | | Kali-Linux-2020.4-vbox-... | Unalloc... | 77.00 MB | 77.00 MB | | |
| ☐ | Kali-Linux-2020.4-vbox-... | | 1001 | | Kali-Linux-2020.4-vbox-... | Disk Im... | n/a | n/a | | |
| ☐ | MBR | | 1003 | | Kali-Linux-2020.4-vbox-... | Metadata | n/a | 512 B | a6114d... | 97d0df... |
| ☐ | Partition 1 | | 1016 | | Kali-Linux-2020.4-vbox-... | Partition | 79.04 GB | 79.04 GB | | |
| ☐ | unallocated space | | 1018 | | Kali-Linux-2020.4-vbox-... | Unalloc... | 79.04 GB | 79.04 GB | | |
| ☐ | Unpartitioned Space [ba... | | 1002 | | Kali-Linux-2020.4-vbox-... | Unparti... | n/a | n/a | | |
| ☐ | Unrecognized file syste... | | 1017 | | Kali-Linux-2020.4-vbox-... | File Sys... | n/a | n/a | | |

Loaded: 18 | Filtered: 18 | Total: 18 | Highlighted: 0 | Checked: 0 | Click to update size

*Figure 10* **FTK Failing to Process the VM's Filesystem**

Active@Disk Editor can open a VDI as a file, but then manual parsing and interpretation will be required. X-ways does interpret a VDI file and the VM filesystem. We recommend using X-ways to do forensics with the VDI for automatic and seamless interpretation.

As X-Ways may not be available to an investigator, we give a means of handling a VDI in the next section.

### 3.2 Investigating a VDI

We recommend an investigator analyze the VDI header to determine all the original metadata and

Snapshot trees, see Figure 7. Then, convert to a different virtual disk image with VirtualBox's Media Manager [9]. We found that VHD or vmdk formats are more accepted by popular forensics tools. However, VirtualBox Media Manager has no hashing function built-in to ensure image integrity, but we found no alterations of the virtual filesystem in our testing. Furthermore, the metadata associated with the VHD or vmdk format will not be correct [7]. From there, an investigator can easily treat the converted image file as a stand-alone machine image. If the investigator prefers a raw format, then most forensic imagers, such as FTK imager, can image a VHD or vmdk to raw or dd format.

## 4.0 FUTURE EXPLORATION

Virtualization continues to expand. As it does digital forensic investigators must adapt to new virtualization techniques and trends. The preceding covers VirtualBox artifacts an investigator should analyze. Investigating other virtualization managers besides VirtualBox would be worthwhile.

There are also other areas associated with aspects of VirtualBox that were outside the scope of this project. One is the various .dll files, driver information, Secure Sockets Layer (SSL) certificates, found under "C:\Program Files\Oracle\VirtualBox". Another is the network forensics implementation between the VM and the host machine. The sectors between the VDI header and the Data (offset 0x1FF-0x200000 by default) have data (bytes which are not null), but we did not investigate this. Furthermore, there may be more forensics artifacts within the snapshot files: VDI and .sav files. Finally, a way to convert between VDI to other image formats in a forensically sound manner has not been developed. We encourage the development of a forensically sound converter. Research or development in any of these areas may uncover forensically useful information.

## 5.0 CONCLUSION

Individuals are also utilizing virtual environments for their flexibility and ease of use. Investigators must be able to analyze a VM itself, not just the host machine. Virtualization is a method for a criminal to hide evidence. Popular digital forensics tools are not able to process both the physical environment and the virtual environment simultaneously. Because of this issue, a cybercriminal can hide evidence behind a VirtualBox VM. We addressed this problem by outlining the VirtualBox log structure, the VDI file, and the snapshot UUID relation tree. These artifacts give investigators a means to analyze both a physical machine and a VirtualBox VM.

## 6. ACKNOWLEDGEMENTS

<REDACTED>

## 7. REFERENCES

[1] Barrett, D., & Kipper G. (2010). Virtualization and forensics: A digital forensics investigator's guide to virtual environments. S. Liles (Ed.). Burlington, MA: Elsevier Inc.

[2]Cimpanu, C. (2020). Ransomware deploys virtual machines to hide itself from antivirus software. Zero Day. Retrieved from https://www.zdnet.com/article/ransomware-deploys-virtual-machines-to-hide-itself-from-antivirus-software/

[3] Coter, S. (2017). Oracle vm virtualbox: log files and how-to manage them. Oracle. Retrieved from https://blogs.oracle.com/scoter/virtualbox-log-files-v2

[4] DigiCert. (2019). Digicert trusted root authority certificates. DigiCert. Retrieved from https://www.digicert.com/digicert-root-certificates.htm

[5] Dykstra J. & Sherman A.T. (August 2013). Design and implementation of FROST: digital forensics tools for the openstack cloud computing platform. Digital

Investigation, 10, S87-S95. https://doi.org/10.1016/j.diin.2013.06.010

[6] Market Research Future. (December 2019). Server virtualization market research report-global forecast 2023. *Market Research Future.* Retrieved from https://www.marketresearchfuture.com/reports/server-virtualization-market-3981

[7] Microsoft. (2006). Virtual hard disk image format specification. Retrieved from https://www.microsoft.com/en-us/download/confirmation.aspx?id=23850

[8] Neal C. (December 2013). Forensic recovery of evidence from deleted oracle virtualbox machines. Utica College. Retrieved from https://programs.online.utica.edu/sites/uti/files/Neal_6_Gonnella_Forensic_Recovery_of_Evidence_from_Deleted_Oracle_VirtualBox_Virtual_Machine.pdf

[9] Openstack. (2021). Converting between image formats. Retrieved from https://docs.openstack.org/image-guide/convert-images.html#vboxmanage-vdi-virtualbox-to-raw

[10] Oracle. (2008). Re: All about VDIs. Retrieved from https://forums.virtualbox.org/viewtopic.php?t=8046

[11] Oracle. (2012). VDI file signature. Retrieved from https://forums.virtualbox.org/viewtopic.php?f=1&t=47732

[12] Oracle Corporation. (2019). Oracle vm virtualbox user manual. Retrieved from https://www.virtualbox.org/manual/

[13] Rajguru S., Chaus K. D., Pathak A., Boramani A.K. (2017). Design of Tool for Digital Forensics in Virtual Environment. International Journal of Computer Applications 163 (4).

[14] VMware. (2011). Virtual disk format 5.0. Retrieved from https://www.vmware.com/support/developer/vddk/vmdk_50_technote.pdf

[15] VMware. (2019). Virtualization. VMware. Retrieved from https://www.vmware.com/solutions/virtualization.html

[16] Wahyudi E., Riadi I., Prayudi, Y. (2018). Virtual machine forensic analysis and recovery method for recovery and analysis digital evidence. International Journal of Computer Science and Information Security, 16 (2).