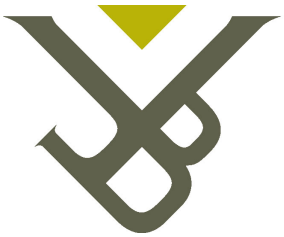

Vrije Universiteit Brussel

Faculty of Science and Bio-engineering Sciences
Computer Sience Department
Dean: Prof. dr. P. Geerlings

Security aspects in virtual networks

by

Laurent De Wilde

Promotor: Prof. dr. ir. Martin Timmerman

A thesis submitted in partial fulfillment for the degree of
Master Of Science in the Applied Computer Science

Academic year 2014 - 2015

Preface

This is where the preface will come. . . .

Laurent De Wilde, June 2015

Copyright declaration

Security aspects in virtual networks

by

Laurent De Wilde

A thesis submitted in partial fulfillment for the degree of
Master Of Science in the Applied Computer Science

Academic year 2014 - 2015

Promotor: Prof. dr. ir. Martin Timmerman
Faculty of Sience and Bio-engineering Sciences
VRIJE UNIVERSITEIT BRUSSEL

Computer Science Department
Dean: Prof. dr. P. Geerlings

Summary

Here comes the summary of the thesis....

Keywords

Security, virtual machines, virtual networks, Hyper-V, Windows, virtualization.

Security aspects in virtual networks

Laurent De Wilde

Supervisor(s): Prof. dr. ir. Martin Timmerman

Abstract—**Here comes a short abstract**

Keywords—**Security, virtual machines, virtual networks, Hyper-V, Windows, virtualization.**

I. Introduction

HERE comes the extended abstract.

II. Virtual networks

Some text ....

III. Conclusion

References

[1] Andrew Stuart Tanenbaum and Todd Austin *Structured Computer Organization*, vol. 6, pp. 2-7, New Jersey, U.S.A., 2013

Contents

Preface i

Copyright declaration ii

Overview iii

Extended abstract iv

Contents v

List of Figures vii

List of Tables viii

List of Abbreviations ix

1 Introduction 1
- 1.1 Virtualization . . . 1
  - 1.1.1 Abstractions . . . 1
  - 1.1.2 Abstraction and virtual machines . . . 3
- 1.2 Why use virtualization . . . 4
  - 1.2.1 Benefits of using machine virtualization . . . 4
  - 1.2.2 Challenges and disadvantages of using machine virtualization . . . 5
- 1.3 Security issues regarding virtualization . . . 6

2 Virtualization 7
- 2.1 Machine virtualization - Hypervisors . . . 7
  - 2.1.1 Bare-metal hypervisors . . . 8
  - 2.1.2 Hosted hypervisors . . . 14
- 2.2 Network virtualization . . . 14
  - 2.2.1 External virtual network . . . 15
  - 2.2.2 Internal virtual network . . . 16

2.2.3 Private virtual network 16
2.2.4 External network virtualization 17
2.3 Storage virtualization 19
2.3.1 Host-based storage virtualization 19
2.3.2 Virtual hard disks in Hyper-V 20

3 Security **21**
3.1 Known non-network related security issues 21
3.1.1 Virusses and malware problems 21
3.2 Known network related security issues 23
3.3 Possible solutions 23
3.4 Unknown security issues 24

4 Conclusions and recommendations **25**

A Test lab **26**

Bibliography **27**

List of Figures

1.1 Abstraction applied to disk storage 2
1.2 The OSI model 3
1.3 Virtualization of hard disks 4

2.1 Bare-metal hypervisor 8
2.2 Hyper-V Architecture 10
2.3 Xen architecture 12
2.4 VMware architecture 13
2.5 VMware simplified architecture 14
2.6 Microsoft Virtual Switch concept 16
2.7 External Virtual Network 17
2.8 Internal Virtual Network 18
2.9 Private Virtual Network 18
2.10 Logical Volume Management 19

3.1 VM based rootkit 22

List of Tables

List of Abbreviations

DMA	**D**irect **M**emory **A**ccess
ESXi	**E**lastic **S**ky **X** **i**ntegrated
OS	**O**perating **S**ystem
OSI	**O**pen **S**ystems **I**nterconnection
NIC	**N**etwork **I**nterface **C**ard
VM	**V**irtual **M**achine
VMBR	**V**irtual **M**achine **B**ased **R**ootkit
VMM	**V**irtual **M**achine **M**onitor
VMX	**V**irtual **M**achine e**X**tensions

Chapter 1

Introduction

In the introductory chapter, some basic aspects of virtualization will pass the revue.

1.1 Virtualization

1.1.1 Abstractions

Computer systems are built as hierarchies with interfaces that separate levels of abstraction. Those levels of abstraction hide lower-level implementation details, which allows for independant development of each seperate layer and thus simplifying the development and maintanance process [Smith and Nair, 2008].

To clearify this layered build of computer systems, two examples are provided: one applies to application design, whereas the second example applies to disk storage.

First example Imaging a higher-level programming language, called **Language1**. This language consists of classes, methods, variables, etc.... This Language5 is very convenient and understandable for people. However, since computer systems only understand machine language, let us call this **Language1**, one cannot simply feed this language to the low-level digital circuits and hoping for the program to execute.

In fact, the gap between what is conventient for people and what is convenient for computers tend to be very large. So the need to translate the higher-level code (Language5) to low-level bytecode (Language1) is needed. This is where abstractation layers come in.

Imaging a virtual machine **Machine1** that accepts Language5. The programs written in Language5 can then be translated of interpreted by a program written in Language1, which can be directly executed on the computer hardware [Tanenbaum and Austin, 2013].

However, for practical reasons, those two languages Language5 and Language1 cannot be too different. This means that intermediate Languages and thus virtual machines must exist to interpret of translate Language5 into a more lower-level language **Language4**. This process of continually and gradually translating higher-level languages into lower-level languages can further be executed until the final, lowest-level language (Language0) is reached [Tanenbaum and Austin, 2013].

Each virtual machine is a layer. So in the example given, 5 layers exist to translate a higher-level language into a lower-level one. The bottom-most language is the simplest one that computers understand (i.e.: bytecode) whereas the top-most language is the most sophisticated one.

Second example In the case of hard disk abstraction, the operating systems hides (i.e.: abstracts) the addressing details, that is, sectors and tracks, for the application software. This means that, from the application point-of-view, the disk appears as a set of files.

Programmers create, read, modify and delete files without knowing those low-level tracks and sectors. The figure below clearyfies the process.

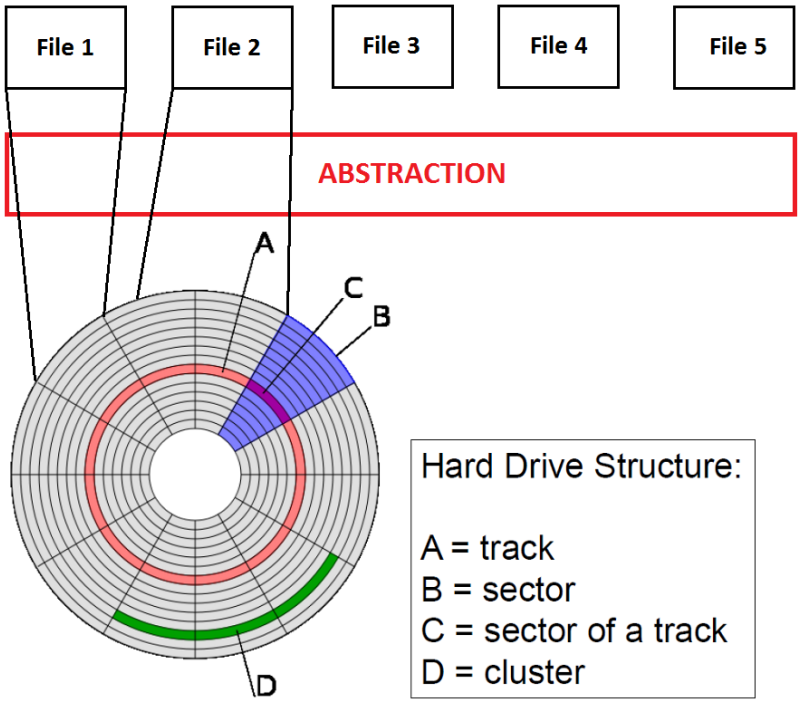

Figure 1.1: Abstraction applied to disk storage. The abstraction layer is in fact the operating system.

In fact, not only computer systems are built as hierarchies: also networking systems use several abstractation layers to communicate with each other. What follows is a brief explanation of the OSI model, that describes how network applications may communicate with each other [Briscoe, 2008].

The OSI model consists of 7 layers as illustrated in the figure below: Top-level applica-

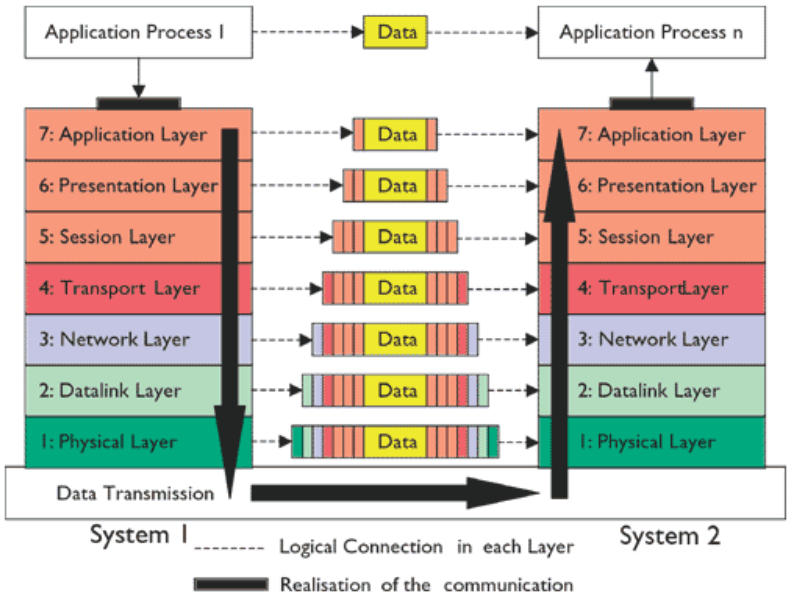

Figure 1.2: The OSI model is another example of abstraction. This time not in computer systems, but in networking.

tions do not communicate directly with each other. Instead, data is passed from one layer to another, starting at the application layer and proceeding to the bottom layer. There, the data is sent over the communication channel to the other host where the whole process takes place in reverse order [Beal, 2015].

The seven layers can be seen as abstraction layers. Each layer hides details of the level directly beneith it.

1.1.2 Abstraction and virtual machines

Virtualization exists in many forms. Not only there exist storage (disk) virtualization, but also network virtualization, virtualized applications and hypervisors. In the first case, virtualization does not necessarily aim to hide details [Smith and Nair, 2008].

Consider the figure below. Virtualization transforms a physical disk into two smaller disks. Each of those disks appears to have its own tracks, sectors and clusters. Furthermore,

the virtualization software uses the file abstraction described in the previous section to map a virtual disk onto the real physical disk.

Data that is to be written onto the virtual disk, is converted to a file write. Since the file resides on the real physical disk, data is actually written to the physical disk.

This is an example of abstraction and virtualization applied to disk storage. Obviously,

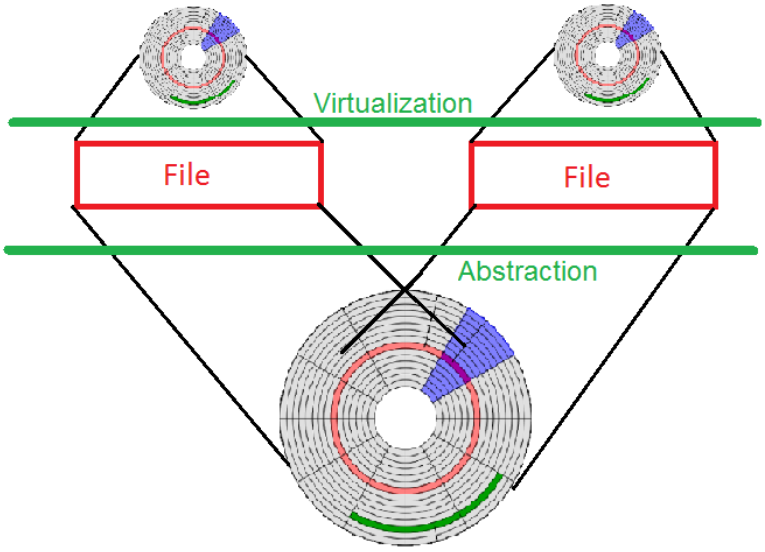

Figure 1.3: Virtualization applied to disk storage

the whole concept of (disk) virtualization can also be applied to physical machines. This is where hypervisors make their entry.

A hypervisor will abstract the physical resources, that is, CPU, memory, disk and network, from systems running on top of it [Vanover, 2013]. This means that those physical resources are shared between the virtual machines and thus allows that multiple virtual machines can run on a single physical machine.

Chapter 2 provides more details about virtualization and hypervisors in particular.

1.2 Why use virtualization

1.2.1 Benefits of using machine virtualization

- **Server consolidation** The most prominent advantage of using machine virtualization (where a hypervisor is used to abstract the physical machine resources) is server consolidation. In the case of a typical non-virtualized application server for example,

about 5% to 10% of the server's hardware is utilized. However, when a server hosts multiple virtual machines, utilization can reach values of 50% to 80% [Bigelow, 2009].

The whole point of machine virtualization is that the hardware of the physical machine is used more efficiently. It permits to get more out of the existing, physical hardware, because multiple virtual machines that act as real servers can run on top of one physical machine. This has an important consequence: fewer physical servers can be used to achieve the same goals. This means lower operating costs, e.g.: less power and air conditioning are needed [VMware, 2015b].

- **Easy cloning** In contrast to a physical server, which consists of a mixture of application files, OS files, driver files and user files, a VM exists as a single file [Bigelow, 2009]. This file can be duplicated, which leads to easy creation of exactly the same machine (server). Obviously, these images can be modified for each application [Vogel, 2014].

- **High availability** When running multiple virtual machines, the load can be distributed amongst them. When a VM fails, another VM can just be started with ensures minimal downtime or data loss [Vogel, 2014].

- **Scalability** Scalability can greatly be improved using virtual machines. Additional resources can quickly be allocated from the host to the guest [Microsoft, 2015a]: when a certain task requires more RAM, adding RAM to the virtual machine consist of editing a parameter on the hypervisor, whereas in the case of a physical machine, it can take minutes to add more RAM [Vogel, 2014].

1.2.2 Challenges and disadvantages of using machine virtualization

- **Longer recovery in case of physical hardware failure** and therefore longer downtime. When the physical machine breaks down because of a catastrophic hardware failure, all the virtual machines need to wait until the physical host is brung up online, after which the VM's can start booting. This means increased downtime [Vogel, 2014].

1.3 Security issues regarding virtualization

Consider the VMware ESXi Hypervisor. All virtual machines are isolated from each other [VMware, 2015a]. The advantages of this practise are for example the fact that if one VM fails, the remaining VM's remain accessible or the fact that if one VM gets infected with, let's say, a Trojan Horse, this will not affect the other VM's [Prowse, 2014].

However, this is not entirely true: cases have been reported in where virusses are able to break out VM's [Wang, 2009]. Additionaly, since VM are networked - either internally using a virtual switch or bridged with the physical network (or both), a new thread arises. Malware with a network component (e.g.: worms), travels where their routing tells - or allows - them to go [Marcin, 2011]. Therefore , it is perfectly possible that an infected VM might infect other VM's while the network administrator believed this could never happen.

The whole point is that one may not assume that VM's are really isolated from each other and that an infected VM cannot infect another one - even if they are not networked. Chapter 3 will cover security in more detail.

In this thesis, virtual networks will be tested against known and unknown security problems concerning virtual networks. Furthermore, securtity aspects will be highlighted in virtual networks.

This chapter briefly described some basic concepts of virtualization, especially machine virtualization. Advantages and disadvantages of virtualization have been discussed as well as a brief mention about security issues related to virtualization.
In the next chapter,)

Chapter 2

Virtualization

In this chapter, (the) three types of virtualization will be explained in detail: machine virtualization, network virtualization and storage virtualization.

Before digging deeper into security aspects in virtual networks, some types of virtualization must first be defined.

2.1 Machine virtualization - Hypervisors

In order to be able to run virtual machines, a hypervisor is needed. Chapter 1 already described the function of a hypervisor, also called a virtual machine manager [House, 2006]: a hypervisor abstracts the physical resources - CPU, memory, disks, network adapters, ...- from systems running on top of it. So it allows for a physical machine (that we will call a host) to share their hardware resources amongst virtual machines (that we will call guests) running as guests on top of the host (the physical hardware). Note that a hypervisor is nothing more than a piece of software [Kleyman, 2012] that ensures that VM's do not interrupt each other [House, 2006].

Each VM appears to have it's own processor, memory and disk. However, the hypervisor is actually controlling the VM's resources and allocating the resources to their needs [House, 2006].

There exist two types of hypervisors: **Type 1** and **Type 2** hypervisors [Siebert, 2006]. Type 1 hypervisors are the so-called **Bare-metal hypervisors** whereas type 2 hypervisors are known as **hosted hypervisors**. Both will be discussed in the upcoming sections.

From know on, the physical machine will be called the **host** and the virtual machine that runs on top of the host (through the hypervisor), will be called the **guest**.

2.1.1 Bare-metal hypervisors

The name 'bare-metal' ("without an operating system") comes from the fact that this type of hypervisor is deployed as a bare-metal installation. This implies that it is not required to first install a server operating system: the hypervisor is the first thing to be installed on the host [Siebert, 2006]. To be precise: the hypervisor is installed as the operating system of the host [Kleyman, 2012].

A bare-metal hypervisor runs directly on the host's hardware and therefore allow for direct access to the hardware recourses, which results in greater performance compared to hosted hypervisors [Siebert, 2006] as illustrated in the picture below. One point of remark: the VM's running on top of the hypervisor do not have direct access to the hardware recourses. Instead, they have a virtual view of for example the processor and run in a private memory address region that is unique for each guest [MSDN, 2015]. Only the hypervisor has direct access to the hardware.

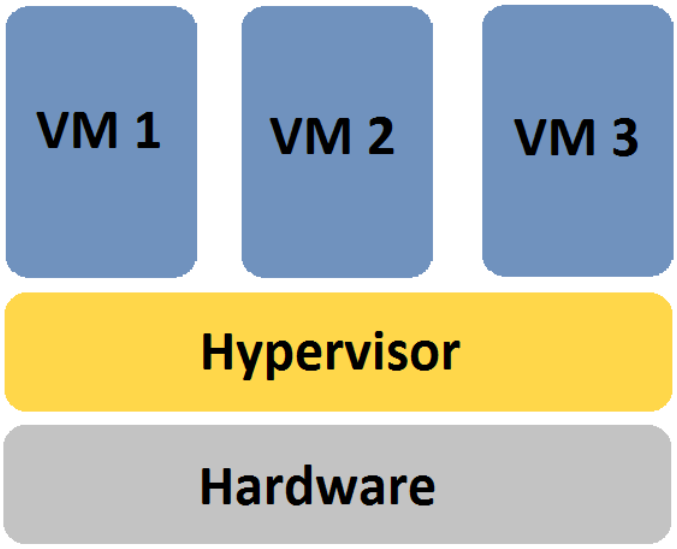

Figure 2.1: A schematic architecture of a typical bare-metal hyervisor. As one can see, the hypervisor runs directly on the host's hardware and therefore has direct access to the hardware.

Hyper-V

Hyper-V is a bare-metal hypervisor developed by Microsoft [Soper, 2012]. Hyper-V is the replacor of Microsoft Virtual PC in newer versions of Windows[Microsoft, 2015b] and was first included in Windows Server 2008, where system administrators could enable the

Hyper-V role [Shinder, 2008]. Hyper-V only supports x64 versions of Windows Server 2008 and Windows Server 2012 [MSDN, 2015].

The Hyper-V role provides the software infrastructure and management tools to create and manage a virtualized computer environment. As previously explained in section 1.3, each VM runs in a isolated computing environment [Microsoft, 2010]. In Hyper-V, this isolation is achieved by means of partitions [MSDN, 2015].

Architecture of Hyper-V In order for Hyper-V to work, a root partition (also known as parent partition) must be present on top of the hypervisor running a 64 bit version of Windows Server 2008 or Windows Server 2012. [MSDN, 2015]. It also runs the virtualization stack, contains the device drivers and therefore has direct access to the hardware resources as well [Smyth, 2009].

In essential, this root partition is nothing more than a virtual machine running Windows Server 2008 or Windows Server 2012 used to control the other guest VM's. The machine that installed the Hyper-V role, becomes the root partition. The root partition can be compared to the Dom0 VM of the Xen Project hypervisor explained in section 2.1.1.

The picture below visualizes the Hyper-V's architecture.

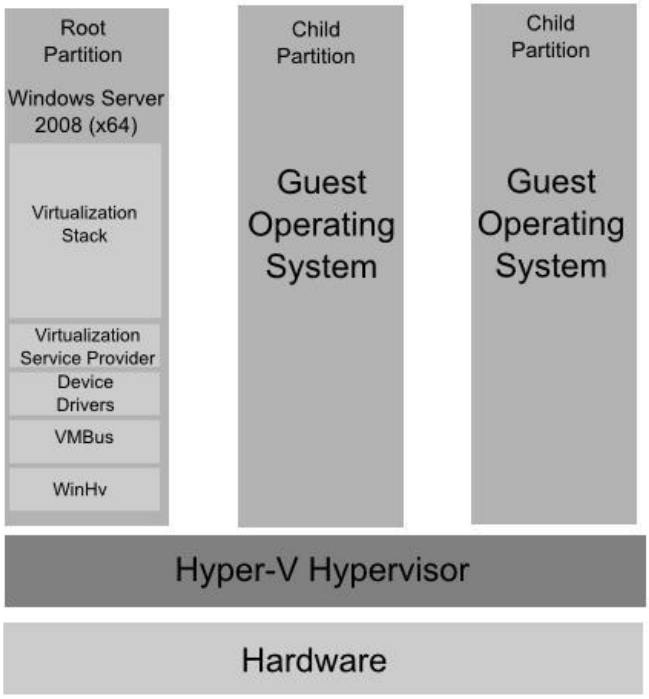

Figure 2.2: The architecture of the Hyper-V hypervisor. The root partion contains the Virtualization Stack which, in his turn, contains a collection of tools that provide the Hyper-V functionality, for example the virtual devices and the device drivers.

Xen

The Xen Project hypervisor is the only open source bare-metal hypervisor available at the time of writing. It consists of four main components being the Xen Project hypervisor, the guest VM's, the Control Domain and the Toolstack [Pavlicek, 2012].

Architecture of the Xen Project hypervisor The Xen hypervisor runs directly on the hardware and contains the scheduler. It is responsible for handling interrupts, CPU time and memory usage. The hypervisor is the first program to run after exiting the boot-loader [Pavlicek, 2012].

The Control Domain (Dom0) is a special type of virtual machine. Compared to the other guest VM's, this Dom0 has special privileges. E.g.: it is able to access the hardware directly and interacts with the other VM's. Therefore, the Dom0 contains the drivers for the hardware and a toolstack [Pavlicek, 2012].

One could compare this Dom0 with the root partition of Hyper-V.

The Dom0 is the first VM to be started by the hypervisor.

The Toolstack (xl is the prefered toolstack at the time of writing [Kurth, 2012]) allows users to manage their VM's [Pavlicek, 2012].

The guest VM's run on top of the Xen hypervisor and run their own operating system and application software. These are the actual VM's.
The picture below visualizes the Xen Project hypervisor's architecture [Kurth, 2012].

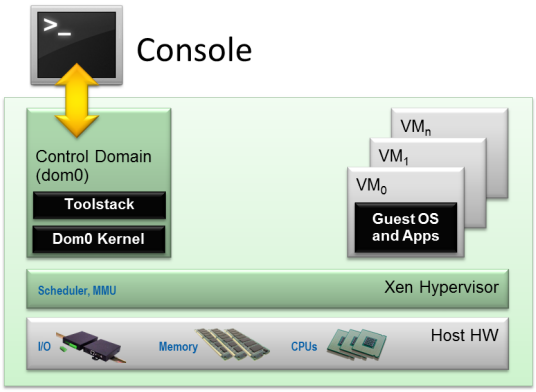

Figure 2.3: The architecture of the Xen Project hypervisor. It consists of four main components: the hypervisor itself, the guest VM's, the Control Domain and the Toolstack for managing the VM's.

VMware ESXi

VMware ESXi the bare-metal hypervisor developed by VMware. It is the successor of the ESX hypervisor [Grehl, 2015].

Architecture of the ESXi hypervisor The VMware ESXi hypervisor consists of the VMkernel, which is the underlying operating system and processes that run on top of the VMkernel.

The VMkernel contains the drivers, the management agents and applications and hardware monitoring components. The processes run on top of it are the Direct Console User Interface (DCUI), the virtual machine monitor (VMM) together with the helper processes VMX. Each VM has its own VMM and VMX process. On top of the VMkernel run the guest VM's.

The DCUI is responsible for low-level configuration and management. It is accessible through the server console and is used for basic, initial configuration.

The VMM is a process that provides an execution environment for the virtual machines, together with some helper processes knows as VMX processes.

Note that - in contrast to Hyper-V and Xen, there does not exist a fully functional console OS that manages the VM's. Instead, only a small POSIX kernel is included. This implies that the footprint of the hypervisor is very small - not more than 32MB.

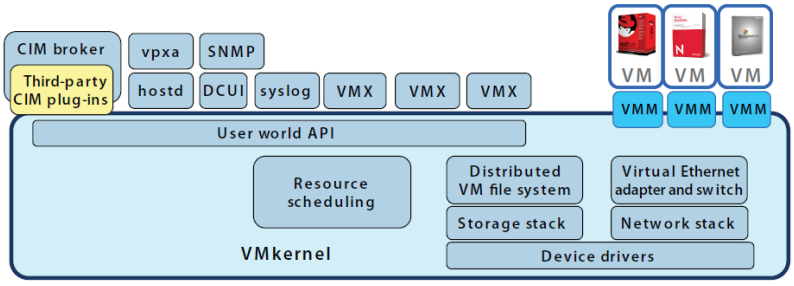

Figure 2.4: ...

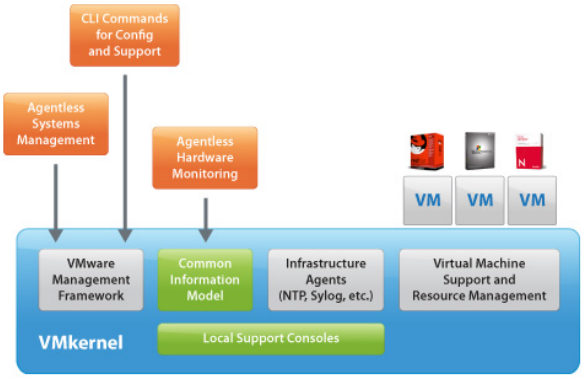

Figure 2.5: ...

The architectures of these 3 hypervisors are roughly equal .... In this thesis, Hyper-V will be used.

2.1.2 Hosted hypervisors

As described in section 2.1.1, bare-metal hypervisors run directly on top of the hardware. However, another type of hypervisor exist that runs on top of an existing operating system: the hosted hypervisors or type-2 hypervisors.

So basically, this means that an operating system is installed inside another operating system in contrast to the bare-metal hypervisor where each OS has direct access (through the hypervisor) to the hardware.

An example of such a hosted hypervisor is VirtualBox [Oracle, 2015].

2.2 Network virtualization

After having defined virtual machines and the hypervisors, it is now time to focus how a virtual machine is able to communicate with the outside world and to have a closer look how internal (i.e.: inside the hypervisor) network virtualization works.

When one wants to provide a VM with network capabilities, at lease one virtual network interface card (vNIC) has to be assigned to this VM - just as a physical computer requires a physical NIC as well [Technet, 2015a].

As Hyper-V will be used in this thesis, the focus of internal virtual networking will be stressed on Hyper-V.

One must not forget that a virtual network is actually just a software logic that is part of Hyper-V and sents and receives packets in OSI layer 2.

There exist three different types of virtual networks in Hyper-V: external virtual networks, internal virtual networks and private virtual networks [Technet, 2015b]. Each of them will be explained in the following section.

2.2.1 External virtual network

Virtualization is all about abstracting existing hardware as explained in the first chapter. The same concept is used for virtual networking. Consider a host with an arbitrary number of cores and an arbitrary amount of RAM which hosts 50 VM's, but with only one physical NIC.

The requests performed by the virtual NIC's of the VM's can overhelm the physical NIC of the host. To overcome this problem, Microsoft developed an abstraction layer that uses the Virtual Switch concept [Kennedy, 2011].

This abstraction layer sits in between the the physical NIC and the vNIC's of the different VM's. It therefore abstracts the physical NIC. Microsoft calls this *External Networking* [Technet, 2015b] and is illustrated in the figure below [Kennedy, 2011].

When creating an external virtual network in Hyper-V, the following changes take place:

- The host creates a new virtual NIC that is used to connect to the physical network. So a minimum of two network adapters exist on the host.
- The Microsoft Virtual Switch Protocol is bound to the physical NIC.

Once created, the virtual network will work exactly the same as a phyiscal network, with that difference that the switch is a software-matic; additional ports (NICs) can be added dynamically. Following figure visualizes the external network concept of Hyper-V. The external network type has the following capabilities [Howard, 2008]:

- It allows for communication between a VM and and an external network so that all VM's are visible as seperate hosts on the external network as if they would be dedicated, physical hosts. One can compare this with Xen bridged networking [Jackson, 2012].

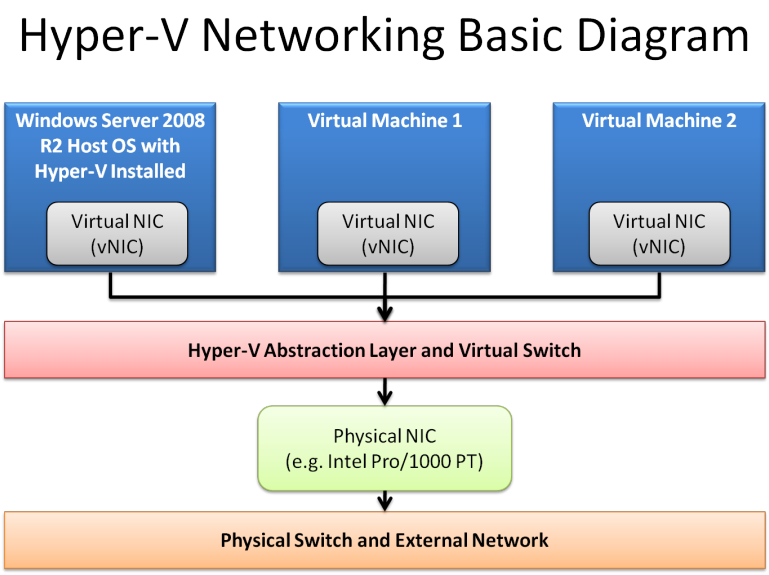

Figure 2.6: The concept of virtual networking as an abstraction layer on top of the physical NIC. The purple layer represents the virtual switch. Note that this is only occurs with external networking - internal and private networking do not use virtual switches.

- It allows for communication between VM's on the same host.
- It allows for communication between the VM's and the host.

2.2.2 Internal virtual network

In an internal virtual network, the phyiscal adapter plays no role in the network and is therefore not bounded. Thus access and communication to external networks (including the LAN outside the virtual machine) is impossible [Kennedy, 2011; Howard, 2008]. However, communication with the host is possible.
Some characteristics of internal networks are [Technet, 2015b]:

- It allows for communication between VM's on the same host.
- It allows for communication between VM's and the host.

The figure below visualizes this concept [Technet, 2015b]:

2.2.3 Private virtual network

When a private virtual network is used, VM's are only able to communicate with each other. No communication with the host exist, however. That is the difference with an

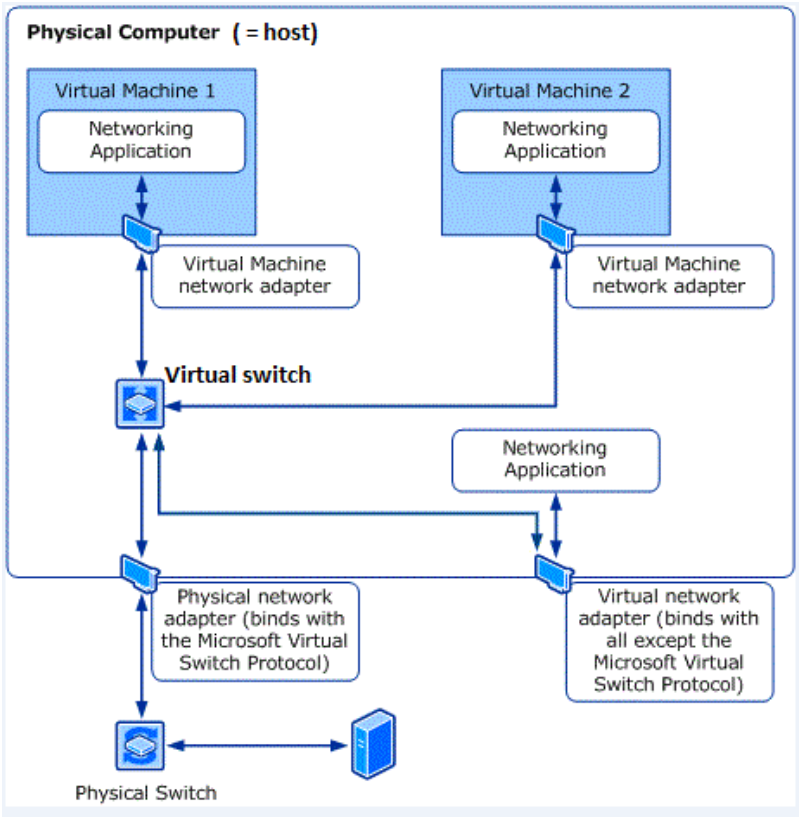

Figure 2.7: A diagram of the different components of external networking. The physical NIC binds with the virtual switch. The networking applications of the host communicate with the external network (the physical switch) through the newly created virtual NIC. Note that each VM is assigned a virtual NIC.

internal private network.
Some characteristics of internal networks are [Technet, 2015b]:

- Communication between VM's only.

he figure below visualizes this concept [Technet, 2015b]:

2.2.4 External network virtualization

VLAN's, ... Moet aan de prof vragen of dit ook moet.

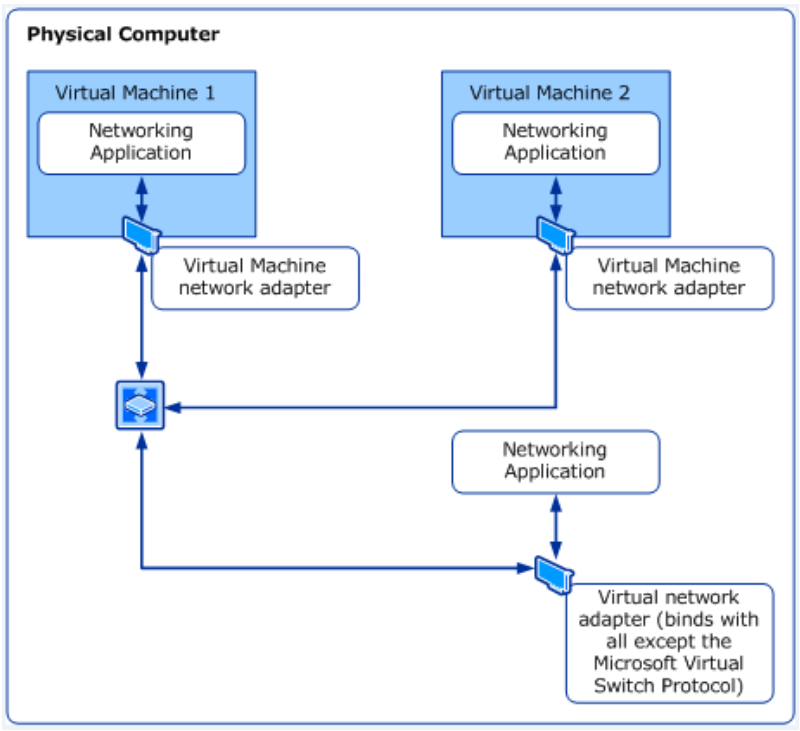

Figure 2.8: ...

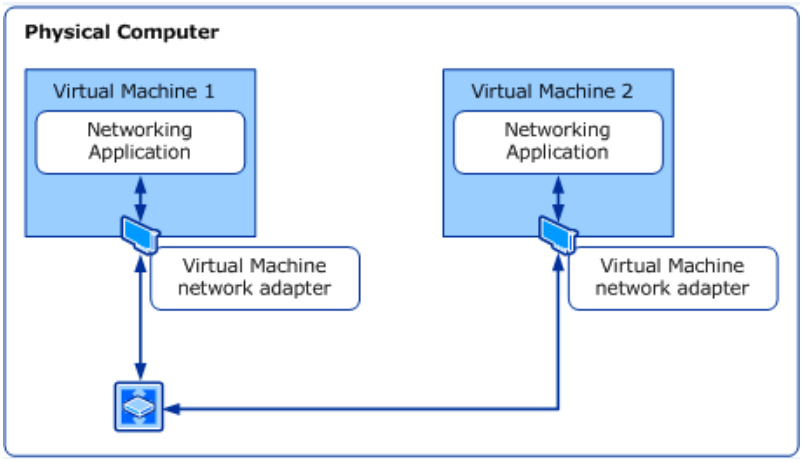

Figure 2.9: ...

2.3 Storage virtualization

2.3.1 Host-based storage virtualization

Host-based storage virtualization adds, as any virtualization technique (see chapter 1), a layer of abstraction between the host OS and the physical disks. Each mayor OS provides a piece of software called a logical volume manager to provide such a layer of abstraction [Garrison, 2011].
Whereas in traditional disk management, the OS looks for physical disks and partitions that reside on those physical disks, with a logical volume manager disks can be abstracted. This way, a logical disk can span multiple physical disk and appear in the OS as one disk.

A logical volume manager adds more flexibility in a way that logical volumes can be resized and moved around. This is not a straightforward job when physical disks and partitions are used [TLDP, 2002].
Microsoft's implementation of logical volume management is called Logical Disk Manager. In the next section, virtual hard disks in Hyper-V will be discussed.
The next figure illustrates the concept of Logical Volume Management.

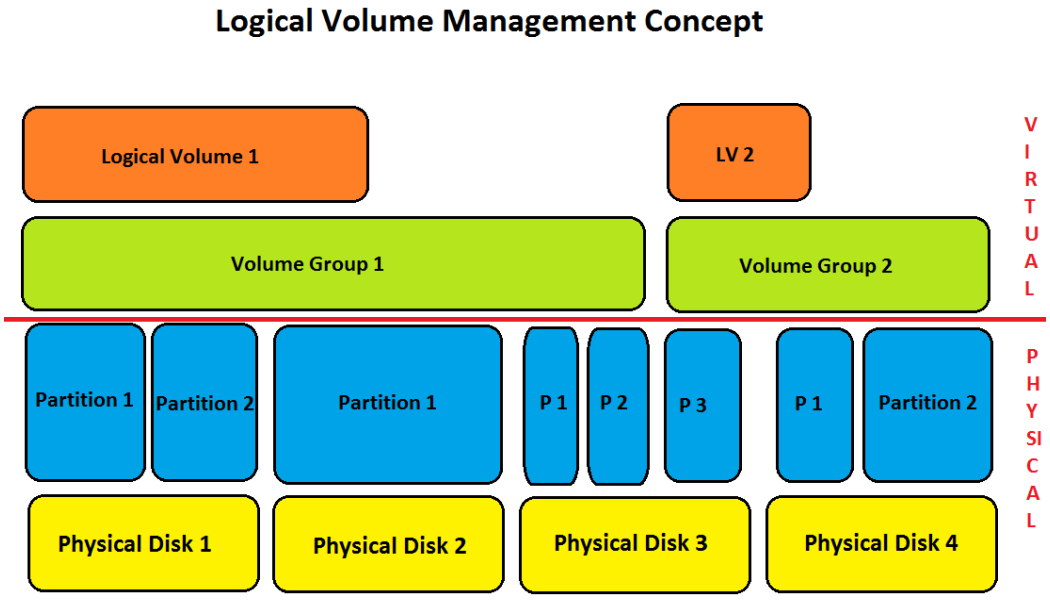

Figure 2.10: LVM adds a layer of abstraction between the OS and the phyical disks / partitions. In the figure, a volume group spans 5 physical partitions and disks. On top of the volume group lies a logical volume (partition). As one can see, there is space left on the volume group and thus expanding the logical volume (partition) is possible. With a phyiscal partition, this would not be possible without losing data.

2.3.2 Virtual hard disks in Hyper-V

A virtual hard disk (with extension .vhd) enables one to create a virtual disk which resides on the physical disk as a single file [MSDN, 2012]. A VHD has the same capabilities as a physical disk and thus are used the same way. They are able to host file systems and support standard disk operations [MSDN, 2012].

Chapter 3

Security

In this chapter, some security issues concerning virtual machines and virtualization in general are covered, which will be the basis for further research in this MA thesis.

3.1 Known non-network related security issues

3.1.1 Virusses and malware problems

The main question we ask ourself here is the fact that if it is possible that virusses and rootkits can be transmitted from a guest VM to the host without the VM having networking capabilities?

As seen in the first section, virtual machines are all isolated from each other. Thus, it might seem odd that breakouks are likely to occur. However, with recent advances in technology as x86 Virtualization and I/O MMU Virtualization it turns out this is in fact possible.

I/O MMU Virtualization and DMA With I/O MMU Virtualization, for example Intel VT-d [Ott, 2009] or AMD-Vi [AMD, 2009] it becomes possible to directly assign an I/O device such as a graphical adapter to a VM [Burger, 2012]. But it also enables Direct Memory Access (DMA). One does not have to be a security expert to see that an infected VM with Direct Memory Access can infect the memory of the host.

x86 Virtualization and rootkits Noticable rootkits [1] are the **Blue Pill** rootkit developed by Joanna Rutkowska and the **SubVirt rootkit**, developed by Microsoft. These rootkits are two examples of VM-based rootkits (VMBR), which install an additional VMM between the host hardware and an existing operating system [Rutkowska, 2006; King et al., 2006], after which this new VMM can be used to host malicious software as illustrated in figure 3.1.1. Additionally and in contrast to usual rootkits, hypervisor-level rootkits remain undetectable, because they cannot be accessed by software [Rutkowska, 2006; Ben-Yehuda, 2013].

One possible solution to minimize (the bluepill attack becomes impracticable) this problem from happening is to disable x86 virtualization [2], despite the loss in performance. Also bear in mind that an anti-virus scanner on the host does not protect a guest VM against virusses [Avis, 2013].

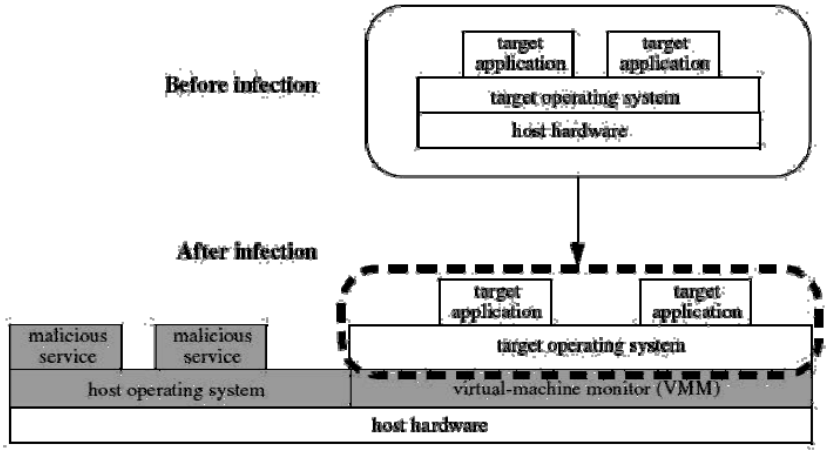

Figure 3.1: A schematic view of a virtual machine based rootkit (VMBR). A VMM, for example Hyper-V, is installed underneath an existing OS. This means that the rootkit is now the hypervisor.

[1] **Rootkit:** a mostly malicious program that is designed to hide itself for detection and thus for the fact that an OS has been compromised. It is used to gain *root* access (hence the name) to the compromised OS to perform, for example, eaves-dropping [Ben-Yehuda, 2013; Kassner, 2008].

[2] **x86 virtualization:** also known as hardware-assisted virtualization [Vangie, 2007], improves the performance when full virtualization is used. It adds hardware support to run VM's more efficiently. Normally, the machine code of the guest OS is translated to machine code of the host OS, by means of binary translation. With x86 virtualization - or hardware-assisted virtualization, the need for such binary translation disappears. Hardware-assisted virtualization must be supported by the CPU [Jeong, 2013]. Therefore, for example Intel and AMD developed respectively Intel-VT and AMD-V for that purpose [Intel, 2015]

3.2 Known network related security issues

When VM's are directly connected to an external network (e.g.: using Hyper-V's external network mode), then the VM is threated as any other physical machine regarding the transmission of virusses. Just a virusses spread between physical machines using files (e.g.: via mail attachments or software downloads) that are transferred through the network [Avis, 2013], the same can also happen with VM's who are connected to an external network, since when a virtual network is connected to a physical NIC, it is exposed to the same threads and security risks as that physical NIC and thus as a normal physical network[Technet, 2015c].

3.3 Possible solutions

Use of firewalls

The built-in Windows Firewall in Hyper-V does not interfere - and thus does not protect guests - with guest traffic in any way. Packets will just pass the Windows Firewall without being analyzed, because of the physical adapter bound to the virtual switch is unbound from anything that Windows Firewall has access to.

However, extensions to the Hyper-V Virtual Switch are available that take care of these problems. A network packet filter and an intrusion detection or firewall are two out of four examples of such extensions [Remde, 2012].

Obviously, one can always completely isolate virtual networks from each other - and from physical networks - in order to protect the host. This can be done using VLAN's - or IP subnets - : the hypervisor (host) does not have to be on the same VLAN as the guests and thus placing it in a seperate VLAN is perfectly possible [Siron, 2014].

Securing the guests

As already described in the first section, virtual machines are isolated from each other, in such a way they cannot access each other's physical resources such as RAM.

However, in 2012, a security issue was found in 64 bit virtualization software running on Intel CPU's. With this vulnerability, when a system exception occured, it became possible to escape from the local guest OS into the host OS with elevated privileges, with all the consequences.

Therefore, a guest has to be secured and securing a guest is just like securing a phyiscal machine.

Antimalware solution

Attention has to be payed when installing antimalware solutions on the host OS. Either antimalware solution poses a threat to the host: most of the antimalware software dislike XML files, however, these files defines the VM's, so deleting them will cause the VM's to disappear.

This means that one has to be very careful when installing antimalware solutions and in particulary take care of exclusions.

3.4 Unknown security issues

This is what the thesis is about.

Chapter 4

Conclusions and recommendations

Here comes the conclusion.

Appendix A

Test lab

Bibliography

AMD (2009). *Advanced Micro Devices, Inc. AMD I/O Virtualization Technology (IOMMU) Specification License Agreement*. AMD. Retrieved on March 7, 2015.

Avis, C. E. (2013). Vmware or microsoft: Protecting vm's - agents or agentless? *http://blogs.technet.com/b/chrisavis/archive/2013/08/22/vmware-or-microsoft-protecting-vm-s-agents-or-agentless.aspx*. Retrieved on March 8, 2015.

Beal, V. (2015). The 7 layers of the osi model. *http://www.webopedia.com/quick_ref/OSI_Layers.asp*. Retrieved on February 24, 2015.

Ben-Yehuda, M. (2013). Machine virtualization, efficient hypervisors and stealthy malware. *http://www.mulix.org/lectures/vmsecurity/vmsec-cyberday13.pdf*. Retrieved on March 8, 2015.

Bigelow, S. J. (2009). Understanding the befefits of a virtual machine. *http://searchservervirtualization.techtarget.com/tip/Understanding-the-benefits-of-a-virtual-machine*. Retrieved on February 25, 2015.

Briscoe, N. (2008). Understanding the osi 7-layer model. *http://memberfiles.freewebs.com/61/55/58745561/documents/OSI.pdf*. Retrieved on February 24, 2015.

Burger, T. (2012). IntelÂ® virtualization technology for directed i/o (vt-d): Enhancing intel platforms for efficient virtualization of i/o devices. *https://software.intel.com/en-us/articles/intel-virtualization-technology-for-directed-io-vt-d-enhancing-intel-platforms-for-efficient-virtualization-of-io-devices*. Retrieved on March 7, 2015.

Garrison, J. (2011). What is logical volume management and how do you enable it in ubuntu? *http://www.howtogeek.com/howto/36568/what-is-logical-volume-management-and-how-do-you-enable-it-in-ubuntu/*. Retrieved on March 3, 2015.

Grehl, F. (2015). Vmware esxi release and build number history. *http://www.virten.net/vmware/esxi-release-build-number-history/#esxi3.5*. Retrieved on February 27, 2015.

House, M. (2006). Hypervisor. *http://searchservervirtualization.techtarget.com/definition/hypervisor*. Retrieved on February 26, 2015.

Howard, J. (2008). Hyper-v: What are the uses for different types of virtual networks? *http://blogs.technet.com/b/jhoward/archive/2008/06/17/hyper-v-what-are-the-uses-for-different-types-of-virtual-networks.aspx*. Retrieved on March 2, 2015.

Intel (2015). Intel virtualization technology (intel vt). *http://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html*. Retrieved on March 7, 2015.

Jackson, I. (2012). Xen networking. *http://wiki.xenproject.org/wiki/Xen_Networking#Bridging*. Retrieved on March 2, 2015.

Jeong, S. (2013). In-depth overview of x86 server virtualization technology. *http://www.cubrid.org/blog/dev-platform/x86-server-virtualization-technology/*. Retrieved on March 7, 2015.

Kassner, M. (2008). 10+ things you should know about rootkits. *http://www.techrepublic.com/blog/10-things/10-plus-things-you-should-know-about-rootkits*. Retrieved on March 8, 2015.

Kennedy, P. (2011). Hyper-v networking and virtual switches overview. *http://www.servethehome.com/hyperv-networking-virtual-switches-overview/*. Retrieved on March 2, 2015.

King, S. T., Chen, P. M., Wang, Y.-M., Verbowski, C., Wang, H. J., and Lorch, J. R. (2006). Subvirt: Implementing malware with virtual machines. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 314–327, Oakland, CA. Institute of Electrical and Electronics Engineers, Inc.

Kleyman, B. (2012). Hypervisor 101: Understanding the virtualization market. *http://www.datacenterknowledge.com/archives/2012/08/01/hypervisor-101-a-look-hypervisor-market/*. Retrieved on February 26, 2015.

Kurth, L. (2012). Choice of toolstacks. *http://wiki.xen.org/wiki/Choice_of_Toolstacks#Default_.2F_XL*. Retrieved on February 26, 2015.

Marcin (2011). How secure are virtual machines really? *http://security.stackexchange.com/questions/3056/how-secure-are-virtual-machines-really-false-sense-of-security*. Retrieved on February 26, 2015.

Microsoft (2010). Overview of hyper-v. *https://technet.microsoft.com/en-us/library/cc816638%28WS.10%29.aspx*. Retrieved on February 27, 2015.

Microsoft (2015a). Advantages of hyper-v. *https://msdn.microsoft.com/en-US/library/cc768521%28v=bts.10%29.aspx*. Retrieved on February 27, 2015.

Microsoft (2015b). Virtuele machines uitvoeren onder windows 8.1 met client hyper-v. *http://windows.microsoft.com/nl-be/windows-8/hyper-v-run-virtual-machines*. Retrieved on February 27, 2015.

MSDN, M. (2012). Virtual disk. *https://msdn.microsoft.com/en-us/library/windows/desktop/dd323684%28v=vs.85%29.aspx*. Retrieved on March 3, 2015.

MSDN, M. (2015). Hyper-v architecture. *https://msdn.microsoft.com/en-us/library/cc768520%28v=bts.10%29.aspx*. Retrieved on February 27, 2015.

Oracle (2015). *VirtualBox User Manual*, volume 6. Oracle.

Ott, D. (2009). Understanding vt-d: Intel virtualization technology for directed i/o. *https://software.intel.com/en-us/blogs/2009/06/25/understanding-vt-d-intel-virtualization-technology-for-directed-io/*. Retrieved on March 7, 2015.

Pavlicek, R. (2012). Understanding hosted and bare-metal virtualization hypervisor types. *http://wiki.xen.org/wiki/Xen_Overview*. Retrieved on February 26, 2015.

Prowse, D. L. (2014). *CompTIA Security+ SYO-201 Cert Guide*, volume 3. Prentice Hall, New Jersey.

Remde, K. (2012). Extend your hyper-v virtual switch in windows server 2012. *http://blogs.technet.com/b/kevinremde/archive/2012/10/20/31-days-of-our-favorite-things-extend-your-hyper-v-virtual-switch-in-windows-server-2012-part-20-of-31.aspx*. Retrieved on March 9, 2015.

Rutkowska, J. (2006). Introducing blue pill. *http://theinvisiblethings.blogspot.be/2006/06/introducing-blue-pill.html*. Retrieved on March 8, 2015.

Shinder, D. (2008). 10 things you should know about hyper-v. *http://www.techrepublic.com/blog/10-things/10-things-you-should-know-about-hyper-v/*. Retrieved on February 27, 2015.

Siebert, E. (2006). Understanding hosted and bare-metal virtualization hypervisor types. *http://searchservervirtualization.techtarget.com/tip/Understanding-hosted-and-bare-metal-virtualization-hypervisor-types*. Retrieved on February 26, 2015.

Siron, E. (2014). 7 keys to hyper-v security. *http://www.altaro.com/hyper-v/7-keys-to-hyper-v-security/*. Retrieved on March 9, 2015.

Smith, J. E. and Nair, R. (2008). The architecture of virtual machines. *http://www.ittc.ku.edu/∼kulkarni/teaching/archieve/EECS800-Spring-2008/smith_nair.pdf*. Retrieved on February 24, 2015.

Smyth, N. (2009). An overview of the hyper-v architecture. *http://www.virtuatopia.com/index.php/An_Overview_of_the_Hyper-V_Architecture*. Retrieved on February 27, 2015.

Soper, T. (2012). Getting to know hyper-v: A walkthrough from initial setup to common scenarios. *https://technet.microsoft.com/library/ee256064%28WS.10%29.aspx*. Retrieved on February 27, 2015.

Tanenbaum, A. S. and Austin, T. (2013). *Structured Computer Organization*, volume 6. Prentice Hall, New Jersey.

Technet, M. (2015a). Configure networking. *https://technet.microsoft.com/en-us/library/cc770380.aspx*. Retrieved on March 2, 2015.

Technet, M. (2015b). Configuring virtual networks. *https://technet.microsoft.com/nl-be/library/cc816585%28v=WS.10%29.aspx*. Retrieved on March 2, 2015.

Technet, M. (2015c). Virtual network security. *https://technet.microsoft.com/en-us/library/cc720393(v=ws.10).aspx*. Retrieved on March 9, 2015.

TLDP (2002). What is logical volume management? *http://www.tldp.org/HOWTO/LVM-HOWTO/whatisvolman.html*. Retrieved on March 3, 2015.

Vangie, B. (2007). Understanding hardware-assisted virtualization. *http://www.webopedia.com/DidYouKnow/Computer_Science/hardware_assisted_virtualization.asp*. Retrieved on March 7, 2015.

Vanover, R. (2013). Introduction to virtualization, abstraction is key. *http://www.techrepublic.com/blog/data-center/introduction-to-virtualization-abstraction-is-key/*. Retrieved on February 24, 2015.

VMware (2015a). Security and virtual machines. *https://pubs.vmware.com/vsphere-4-esx-vcenter/index.jsp#security_for_esx_systems/c_security_and_virtual_machines.html*. Retrieved on February 25, 2015.

VMware (2015b). Virtualization advantages. *http://www.vmware.com/virtualization/virtualization-basics/virtualization-benefits.html*. Retrieved on February 25, 2015.

Vogel, D. (2014). The benefits and challenges of virtual machine hosting. *http://www.datapipe.com/blog/2014/04/23/benefits-challenges-virtual-machine-hosting/*. Retrieved on February 25, 2015.

Wang, Z. J. (2009). How to recover virtualized x86 instructions. *https://www.virusbtn.com/conference/vb2009/abstracts/Wang.xml*. Retrieved on February 26, 2015.