EXPOSING INTER-VIRTUAL MACHINE NETWORKING TRAFFIC TO EXTERNAL APPLICATIONS

THESIS

Charles E. Byrd, CW3, USA

AFIT-ENG-MS-16-M-006

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-16-M-006

EXPOSING INTER-VIRTUAL MACHINE NETWORKING TRAFFIC TO EXTERNAL APPLICATIONS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Operations

Charles E. Byrd, B.S.C.I.S.

CW3, USA

March 2016

AFIT-ENG-MS-16-M-006

EXPOSING INTER-VIRTUAL MACHINE NETWORKING TRAFFIC TO EXTERNAL APPLICATIONS

THESIS

Charles E. Byrd, B.S.C.I.S.
CW3, USA

Committee Membership:

Barry E. Mullins, Ph.D.
Chair

Timothy H. Lacey, Ph.D.
Member

Michael R. Grimaila Ph.D., CISM, CISSP
Member

AFIT-ENG-MS-16-M-006

Abstract

Virtualization is a powerful and fast growing technology that is widely accepted throughout the computing industry. The Department of Defense has moved its focus to virtualization and looks to take advantage of virtualized hardware, software, and networks. Virtual environments provide many benefits but create both administrative and security challenges. The challenge of monitoring virtual networks is having visibility of inter-virtual machine (VM) traffic that is passed within a single virtual host. This thesis attempts to gain visibility and evaluate performance of inter-VM traffic in a virtual environment.

Separate virtual networks are produced using VMWare ESXi and Citrix XenServer platforms. The networks are comprised of three virtual hosts containing a Domain Controller VM, a Dynamic Host Configuration Protocol server VM, two management VMs, and four testing VMs. Configuration of virtual hosts, VMs, and networking components are identical on each network for a consistent comparison. Transmission Control Protocol (TCP) and User Datagram Protocl (UDP) traffic is generated to test each network using custom batch files, Powershell scripts, and Python code.

Results show standard virtual networks require additional resources (e.g., local Intrusion Detection System) and more hands-on administration for real-time traffic visibility than a virtual network using a distributed switch. Traffic visibility within a standard network is limited to using a local packet capture program such as *pktcap-uw, tcpdump, or windump*. However, distributed networks offer advanced options, such as *port mirroring* and *NetFlow*, that deliver higher visibility but come at a higher latency for both TCP and UDP inter-VM traffic.

Acknowledgements

I would like to extend my deepest appreciation to my committee for their guidance, direction, and education offered. Dr. Mullins, Dr. Lacey, and Dr. Grimaila are the consummate professionals that I am pleased to know and work alongside. I would also like to thank the rest of AFIT staff and faculty for the excellent education presented. The education and challenges offered far exceeds any other I have attended.

The most important people I would like to thank are my wife and kids. The understanding and encouragement they give is why I can do what I do. Last but not least, I want thank my Lord, Jesus Christ, for his grace and mercy that allows me to be where I am today.

Table of Contents

Page

Page

Page

List of Figures

List of Tables

List of Abbreviations

Abbreviation		Page
OSI	Open System Interconnection	11
SAN	Storage Area Network	14
NAS	Network Attached Storage	14
Dom0	Domain 0	18
DomU	Domain User	18
HAV	Hardware Assisted Virtualization	19
VT-x	Intel Virtualization Technology	21
AMD-V	Advanced Micro Devices Virtualization	21
FDC	Floppy Disk Controller	23
CIVSched	Communication-Aware Inter-VM Scheduling	25
vSS	vSphere Standard vSwitch	30
OVS	Open vSwitch	30
vDS	vSphere Distributed Switch	31
DVSC	Distributed Virtual Switch Controller	31
SPAN	Switch Port Analyzer	31
RSPAN	Remote SPAN	31
SUT	System Under Test	32
CUT	Component Under Test	32
VLAN	Virtual Local Area Network	34
SSH	Secure Shell	35
CLI	Command Line Interface	35
GUI	Graphical User Interface	35
vMA	vSphere Management Assistant	35
XAPI	Xen Application Programming Interface	35

Abbreviation		Page
IASE	Information Assurance Support Environment	36
SHB	DoD Secure Host Baseline Repository	36

EXPOSING INTER-VIRTUAL MACHINE NETWORKING TRAFFIC TO EXTERNAL APPLICATIONS

I. Introduction

1.1 Background

Virtualization history dates back to the 1960's but the technology as we know it today started in the late 1990's. The initial virtualization began with the IBM mainframe in the 1960's, and server virtualization was introduced by VMWare in 1999 [1]. Virtualization is an important factor in the Information Technology (IT) industry and most recently a high priority for the Department of Defense (DoD). The newly assigned director of the Defense Information Systems Agency (DISA), LTG Alan Lynn, has made virtualization a priority. The director of DISA's development and business center, Alfred Rivera, stated "Virtualization will be a major focus for the agency moving forward" [2]. The way ahead will require extensive research and development primarily focusing on the topic of security.

Virtualization covers the complete spectrum of IT resources, including software, hardware, and networking. The basic idea of virtualization is to consolidate physical resources into a virtual environment. This consolidation is a key benefit of virtualization but carries a higher level of risk management than traditional IT security. Risk evaluation associated with virtualization must account for infrastructure complexity and the ease of moving resources. The additional risk will require changes to security policies and procedures needed for administering the vast DoD IT infrastructure.

The increased use of virtualizaion by the DoD brings more reliance on the different vendors that offer it. VMWare and Citrix are two of the top providers supporting enterprise-level virtualizaion. The variety of deployments they support vary from a small Local Area Network (LAN) to large cloud-based infrastructures. The two platforms have similar functionality but different methods of implementation. Therefore, security policy deployment is platform dependent and must incorporate the network and system administrator expertise. Both VMWare and Citrix have virtual networking functionality built-in which minimizes network traffic on the physical network but increases processing requirements from the host CPU. The focus of this research is to determine the security implications of virtual networking by monitoring the visibility and performance of traffic that passes through the virtual infrastructure.

1.2 Motivation

With the change from physical to virtual network came a paradigm shift in network administration and a new set of challenges. Network administrators no longer have a complete view of traffic passing through the network and must rely on server administrators. Maintaining visibility by sending traffic to proper administrative roles can increase the security of a network. An Intrusion Detection System (IDS), either host or network based, also relies on network traffic for detecting security risks and intrusions. An IDS may miss an indicator of risk or vulnerability if it does not have complete visibility of network traffic. Providing visibility of inter-VM traffic offers an increased awareness of network administration and security.

1.3 Research Goals and Hypothesis

The main goal of this research is to determine visibility of inter-VM traffic and evaluate methods for capturing such data. Results define the visibility of traffic

and the approach taken on each virtual network. Researching the different options available in capturing network traffic can assist with the implementation of an effective security framework.

Other than security, size and performance are main criterias when determining the type of virtual network to deploy. The size selection is based on the two types of virtual networks available, standard for smaller and distributed for larger or enterprise level. Efficiency of a network relies heavily on the performance. Performance testing for different virtual networks is the secondary goal of this study. The goal is to capture and evaluate the performance of internal and external network traffic. The results can be an important distinction when determining the requirements of a specific type of virtual network. Additionally, the performance data can provide insight for the selection of a virtual platform.

It is believed that visibility into inter-VM traffic and performance is based upon the virtual platform. Each platform is expected to offer different methods for gaining traffic visibility and configurations for enhancing performance. It is hypothesized that distributed networks for each platform offer higher visibility and lower performance than a standard network.

1.4 Approach

The research approach encompasses a framework using DoD specifications. The network infrastructure built reflects a scaled-down DoD network that may provide results for visibility and performance of an actual network. The baseline for the virtual environment is constructed using VMWare and Citrix virtual platforms. All VMs instantiated use a DISA-approved Operating System (OS). Custom batch files, Powershell scripts, and Python code provide the means for network traffic generation.

1.5 Assumptions and Limitations

This implementation assumes that all connected devices are authorized to be on the network or added by an administrator. Tasks must be performed by a network or domain level administrator. The evaluation is limited to a confined network with no access to other networks.

Virtual networking traffic is broken down into several categories - production, management, storage, and vMotion. The two traffic categories of concern for this research are limited to production and management. Storage and vMotion traffic deal primarily with host communications and should be included in future works.

1.6 Contributions

This thesis contributes to the future of virtualization within the DoD community. Specific contributions include the ability to gain visibility of network traffic and obtain performance standards for virtual networks. As the DoD maintains both large and small networks, the process of gaining visibility and calculating performance provides network engineers and administrators the tools necessary to build virtual networks for different scenarios.

1.7 Thesis Overview

This thesis is organized into five chapters. Chapter 2 offers a detailed perspective of virtualization and the components that make up the technology. Chapter 3 provides the methodology used which includes physical infrastructure, virtual configuration, and tools to generate network traffic. Chapter 4 gives a summary of results and analysis for each evaluation. Chapter 5 concludes this study and suggest recommendations for future work.

II. Background and Related Research

This chapter describes virtualization and the different aspects of the technology. Section 2.1 defines virtualization and the components that make up the virtual architecture. Section 2.2 introduces virtual networking and mechanisms used for communications. Section 2.3 outlines different models used for implementation of a virtual environment. Section 2.4 covers the two privilege mechanisms used to regulate hardware access to physical and virtual resources. Section 2.5 describes the techniques that virtualization is built upon. Section 2.6 deals with security and vulnerabilities of virtualization. Section 2.7 concludes the chapter by providing details on related research conducted on Inter-VM traffic.

2.1 Virtualization

Virtualization has a long history that dates back to the 1960's when IBM made the first attempt to virtualize a mainframe OS [1]. Today it has become a powerful and fast growing technology that has been widely accepted throughout the computing industry. Computing concepts of virtualization use a software abstraction layer on a single computer to separate logical devices or applications from the physical hardware such as the Central Processing Unit (CPU), Random Access Memory (RAM), Network Interface Controller/Card (NIC) and storage devices. The technology allows for consolidation of multiple physical server resources onto a single computer called a virtual host. The advantage of consolidation provides savings in hardware cost, energy consumption, human effort, and management of resources.

A virtualization architecture provides the ability to reach a maximum utilization of server resources. Figure 1 displays a general architecture overview of a virtual host. Key components of the virtual host include Virtual Machines (VM), hypervisors,

and physical hardware. An understanding of each component is essential in how virtualization operates. The physical hardware is the basic resources available on the virtual host and is determinate on the virtual environment deployed. Sections 2.1.1 and 2.1.2 discusses the other components in further detail.

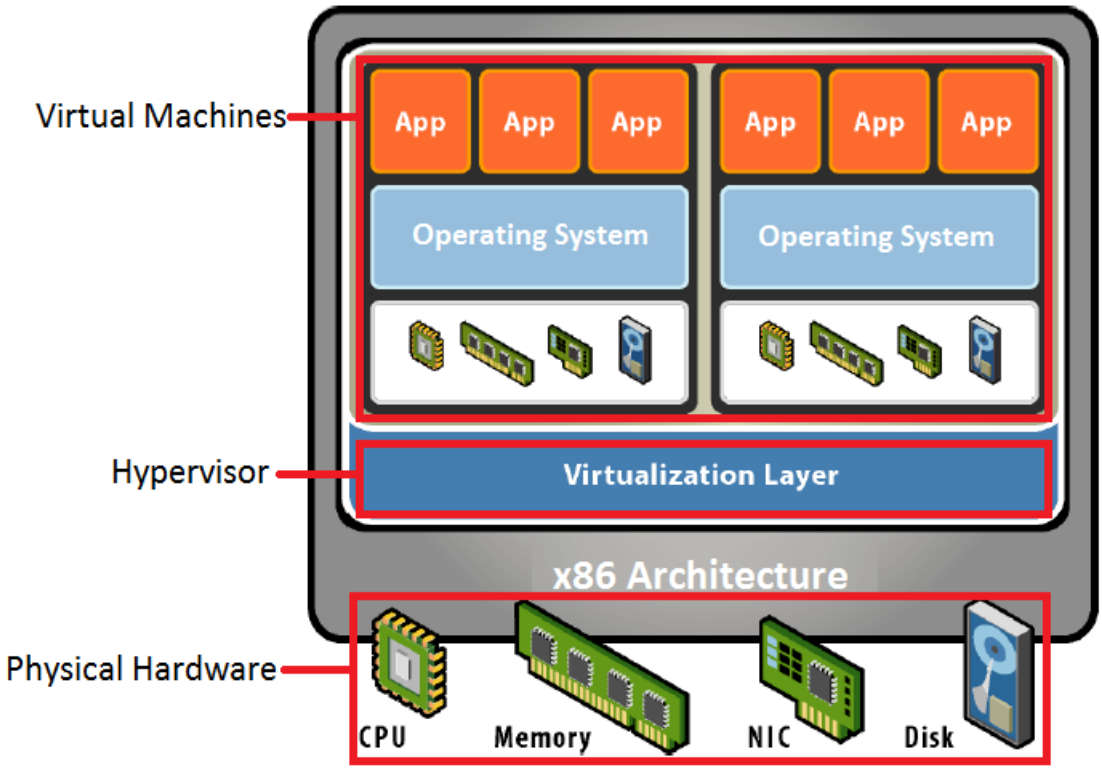


Figure 1. Virtual Host Architecture [3]

2.1.1 Hypervisor.

The hypervisor, also known as Virtual Machine Manager (VMM), is software that operates between physical hardware and virtual machines on the virtual host. The main responsibility of the hypervisor is to manage the interaction between VMs and allocation of CPU, network I/O, memory processing, and storage resources of physical hardware. The hypervisor must also balance the workload and secure the physical resources without interrupting or disturbing other VMs. The two variations of hypervisors are Type 1 and Type 2 as shown in Figure 2.

Type 1 hypervisors run directly on the server hardware without an OS beneath it. Because there is no intervening layer between the hypervisor and the physical hardware, this is also referred to as a bare-metal implementation. Without an intermediary, the Type 1 hypervisor can directly communicate with the hardware resources in the stack below it, making it more efficient than the Type 2 hypervisor [4]. Production systems primarily use Type 1 hypervisors for operations. VMWare ESXi, Microsoft Hyper-V, and Citrix XenServer are the most common Type 1 hypervisors.

A Type 2 hypervisor is a software application that operates using the existing OS and coordinates with the host OS for access to hardware resources. Type 2 hypervisors are typically used within a personal computing environment such as VMWare Fusion, VMWare Workstation, Oracle VirtualBox, and Microsoft VirtualPC.

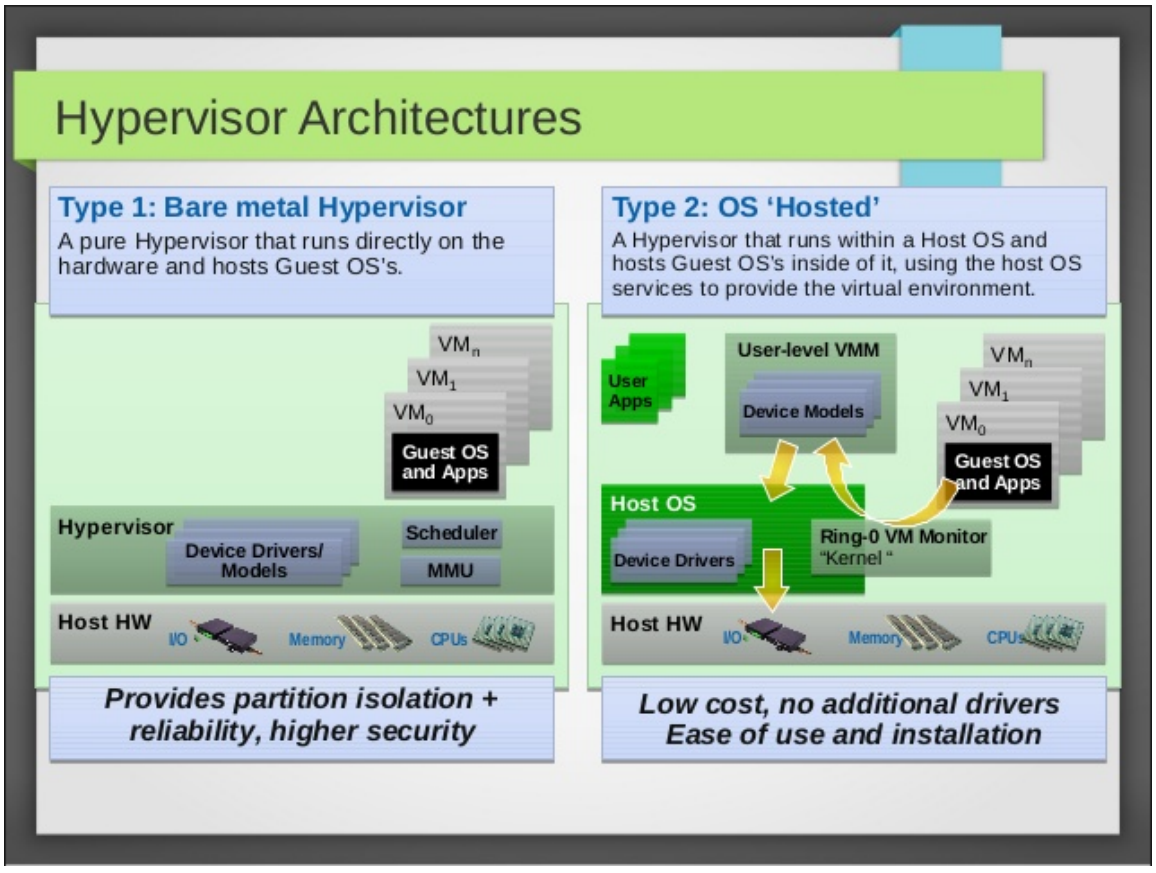


Figure 2. Hypervisor Types [5]

2.1.2 Virtual Machine.

Virtual machines, also known as guest machines, are the fundamental components of virtualization. They act as their own entity and operate similar to a physical machine. Resources from the physical host are shared between VMs and allocated by the hypervisor through the VM interface. Figure 2 also displays the relationship between VMs and where they reside within the hypervisor architecture. VMs consist of both configuration and virtual disk files which describes the resources available for utilization. A VM may be a virtual appliance or a fully functional OS running on top of a virtual layer.

The way to manage and maintain the provisioning of VMs is through clones, templates, and snapshots. Cloning of a VM creates an exact copy of the running system and only requires minimum configuration changes such as hostname and IP address. Similar to clones, templates use a framework of commonly used VMs to provision additional VMs. The difference between templates and clones is that a template cannot be operational and must be converted before use. Snapshots give the ability to take a picture of the VM state and revert back to a specific date and time. Snapshots are similar to incremental backups as they only save the information that has changed since instantiation or a previous snapshot.

Portability and the distribution of VMs use a standard called Open Virtualization Format (OVF). The goal of an OVF is to package a software application with an OS on which it is certified into a format that is easily transferred from a vendor. The final product from testing and development and into production is a pre-configured, pre-packaged unit with no external dependencies [6]. An OVF file provides the ability to export a VM for distribution or import to instantiate a new VM. The OVF standard also supports the Open Virtualization Appliance (OVA) file for exporting and importing virtual appliances. Virtual appliances are pre-configured virtual machines

that are self-contained, self-consistent, software applications that provide a particular service or services [6].

2.2 Virtual Networking

Virtual networking is a collection of virtual machines that are logically connected to each other through a Virtual Switch (vSwitch). VMs obtain network access through one or more Virtual Network Interface Cards (vNIC) using a Virtual Interface (VIF) connected to a Physical Interface (PIF) on the host. Virtual networks can be configured where VMs use a bridge to connect directly to a physical network, use Network Address Translation (NAT) to share host IP addresses, or use host-only to allow for internal traffic only.

2.2.1 Virtual Switches.

The host hypervisor is responsible for managing the virtual network and connection to the physical network. The connection between the host and physical network is accomplished using a vSwitch. A vSwitch is a software-emulated virtual Ethernet switch typically implemented within the virtualization infrastructure [7]. Connectivity between VMs on the same or different host is also managed by a vSwitch.

A physical host can have multiple vSwitches where typically one NIC is assigned to each vSwitch. A vSwitch performs similar to a physical switch by forwarding packets to designated ports using the layer 2 address of the destination. There are two categories of vSwitches as illustrated in Figure 3 for standard and Figure 4 for distributed. A standard vSwitch is a standalone virtual switch on a host that connects internal VMs to each other or to the physical network. A distributed vSwitch acts as a virtual switch connecting multiple host for a more robust administration of the virtual network.

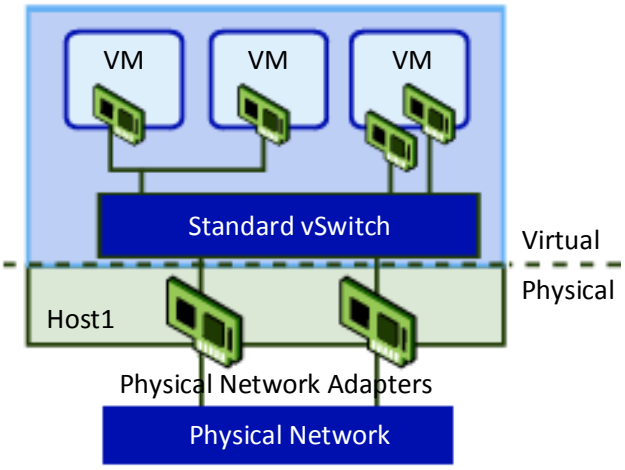


Figure 3. Standard vSwitch [8]

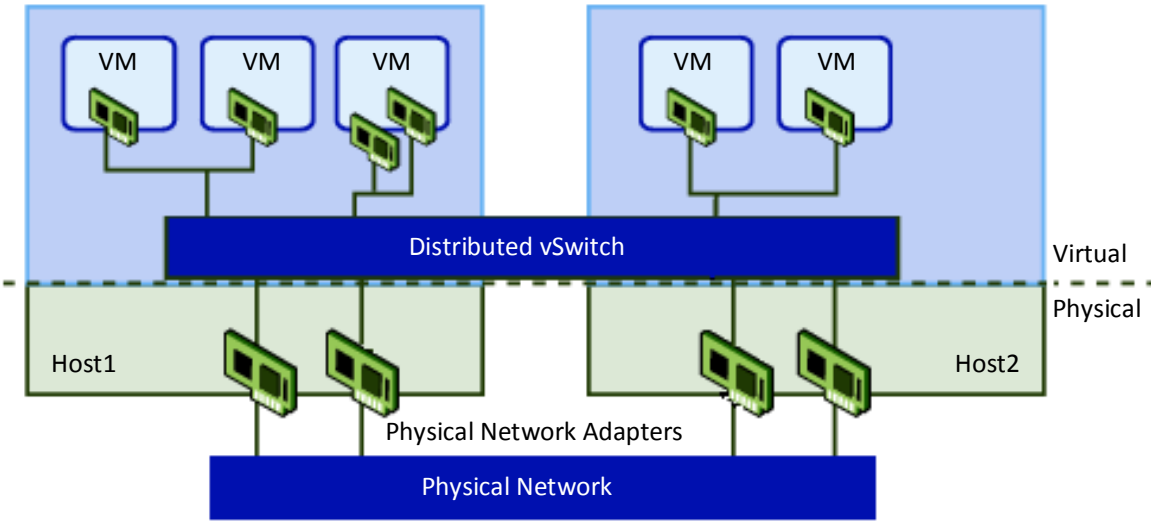


Figure 4. Distributed vSwitch [8]

2.2.2 Virtual Traffic.

Production and *Management* are categories of virtual traffic that pertain to this research. Production traffic is used to communicate between computing resources such as VMs, Exchange services, security devices, and other domain services. Internal and external traffic between VMs is an example of production traffic. Management traffic provides the ability to administer virtual hosts, VMs, and virtual networking. Man-

agement traffic is passed between virtual hosts using management software packages such as VMWare vCenter or Citrix XenCenter.

A consistent network traffic stream must be communicated between virtual hosts and VMs to ensure accurate results for the experiments. The network traffic generated employs the network-layer protocol Internet Control Message Protocol (ICMP) and the transport-layer protocols, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) in conjunction with the Internet Protocol (IP) protocol. Figure 5 outlines the layers within the Open System Interconnection (OSI) and TCP/IP reference models plus a list of example protocols associated with each layer.

TCP-IP	OSI	Example Protocols
Application	Application	HTTP, FTP, IRC, SSH, DNS
	Presentation	SSL, FTP, IMAP, SSH
	Session	SOCKETS
Transport	Transport	TCP, UDP
Internet	Network	ICMP, IP, IPSec, IGMP
Network Access	Data-Link	Ethernet, SLIP, PPP, FDDI
	Physical	Coax, Fiber, Wireless

Figure 5. OSI and TCP-IP Reference Model

2.3 The Kusnetzky Group Model of Virtualization

There are many forms of base virtualization that provide "Big Data," clusters, high-performance computing, extreme transaction processing, desktop virtualization, and the most common server virtualization. The Kusnetzky Group Model of Virtualization, illustrated in Figure 6, details the seven layers of virtualization and is a reference model that provides an easier way to understand the complex environment of virtualization [9]. The following sections discuss this model in detail.

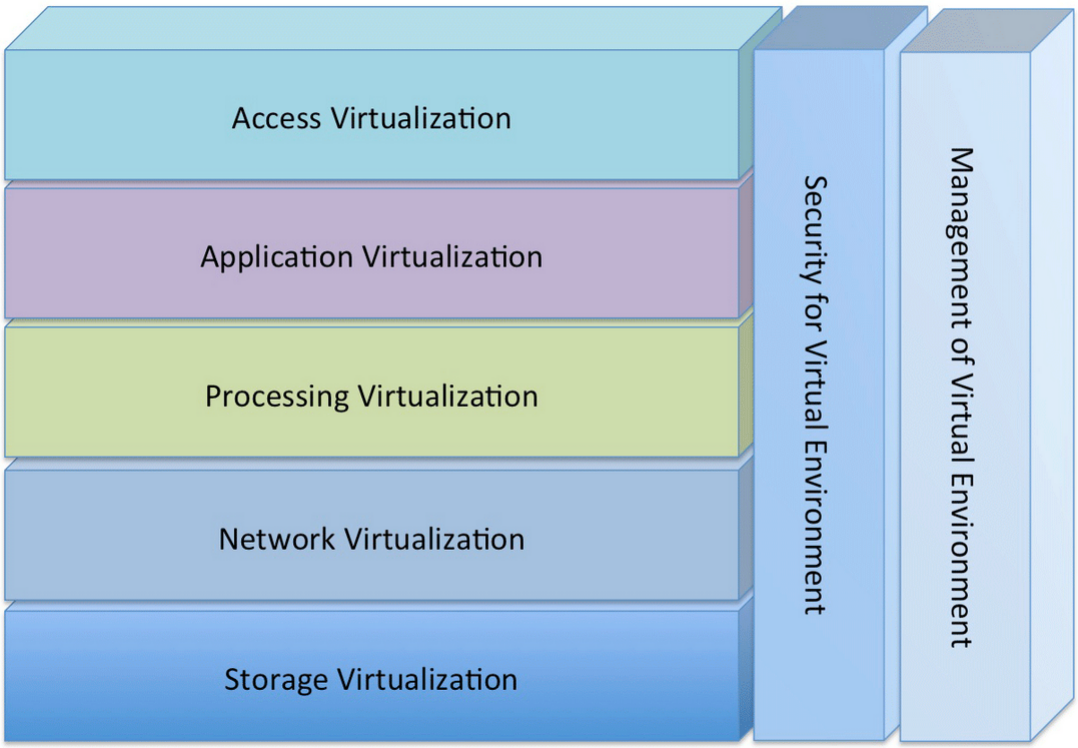


Figure 6. The Kusnetzky Group Model of Virtualization
[9]

2.3.1 Access Virtualization.

Access virtualization is designed to make available applications and workloads in a virtual environment. A remote device is used to display the user interface of a remote application and accept input from the user. The remote application is not dependent on the device making the call. An example would be using a thin client that requires applications and services from another computer, such as servers or mainframes. The ability to access applications and services on multiple devices running different OSs is becoming more prevalent as technology changes. Microsoft Terminal Services, Citrix XenApp, and Linux X-Windows are just a few applications that provide access virtualization. The benefits of access virtualization are greater agility, device independence, improved availability, greater security, cost reduction, and access to Software as a Service (SaaS) applications in a cloud environment [9].

2.3.2 Application Virtualization.

Application virtualization entails the encapsulation of both client-side, known as streaming, and server-side, known as remote applications. Encapsulation creates a protected environment that isolates applications from the OS and other host applications. Isolation minimizes application compatibility issues and allows applications to operate on different versions of the same OS. Client-side applications download required components to the local system and request additional components as needed. Server-side applications operate from the server and run multiple instances allowing more clients to access the applications simultaneously. The server-application provides an interaction with the clients through available interfaces or agents. The main difference between access and application virtualization is that application virtualization still relies on a portion of the application to be capable of running on the device. Microsoft App-V, VMWare ThinApp, and Citrix XenApp are all applications that provide application virtualization. The benefits of application virtualization are greater application isolation, OS independence, improved availability, improved performance or scalability, and cost reduction [9].

2.3.3 Processing Virtualization.

Processing virtualization has five forms: parallel processing monitors, workload management monitors, high availability/fail over/disaster recovery monitors, virtual machine software, and OS virtualization and partitioning. Using the five forms, processing virtualization takes on two different types, make a single system appear to be many or many systems appear to be one.

Processing virtualization does one of three things:

1. Encapsulates host OS so that many virtual systems can run on a single system
2. Links multiple systems together so that workloads will fail over if a system fails

3. Links systems together so an application or data can be spread across all of them for performance or scalability

Server virtualization is an example of processing virtualization where multiple servers with different functions can be isolated within a single physical host. Microsoft Hyper-V, VMWare ESXi, and Citrix XenServer are all products that provide processing virtualization. The benefits of processing virtualization are greater application isolation, OS independence, improved availability, improved performance or scalability, cost reduction, optimization [9].

2.3.4 Network Virtualization.

Network virtualization creates an artificial view of the network that hides the physical network from clients and servers. It provides functionality of network routing, NAT, and network isolation. Cisco, HP, IBM, and Juniper systems all provide network virtualization within a number of network or general purpose servers. The benefit of network virtualization is improved network reliability, performance, and security [9].

2.3.5 Storage Virtualization.

Storage virtualization provides tools to present an artificial view of a storage environment. Storage servers must directly support the variety of storage virtualization being implemented. An example is a remote device or service providing data looks as if it is directly attached to the local system. Storage devices include CD/DVD drives, traditional drives, or solid state technology. Storage virtualization is typical configured using Storage Area Network (SAN) if attached locally or Network Attached Storage (NAS) if accessed over the network. EMC, Hitachi, HP, IBM, and NetApp provide industry-standard servers supporting storage virtualization. The benefits of

storage virtualization are high availability/fail over/disaster recovery, improved storage performance, and making storage resource server available to everyone [9].

2.3.6 Security for Virtual Environment.

Security for Virtual Environment touches all layers of the Kusnetzky Group Model of Virtualization. It refers to the tools necessary to control access and use of all other layers of virtualization technology. There are two approaches used for securing a virtual environment. The first is to add a small piece of software, known as an agent, to a virtual device. This requires the device to perform security processes locally but is a daunting task for managing in a large environment. The second approach is to capture streams of network communication between virtual resources and direct it through a security appliance server. This approach removes any processing on devices and allows for detecting worms, viruses, malware, and other vulnerabilities circulating in the environment. CA, Cisco, HP, IBM, and Juniper Systems provide security products for virtual environments which protect both computing and networking. The goal of security in a virtual environment is simple, protect all information and computing resources so that only authorized use is made of them [9].

2.3.7 Management of Virtual Environment.

Management of Virtual Environment is the second layer that touches all layers of the Kusnetzky Group Model of Virtualization. It refers to the tools necessary to install virtual environments and to watch, analyze, control, automate, and optimize what they are doing. The management of virtual environment has become a critical layer for virtualization. Management involves many aspects of both virtualized client-side and server-side resources. Both resources include the creation of VM images while setting proper parameters, and installation/update of the OS. Managing client-side

resources also include configuring user data, application installation, and adjusting application configurations. Managing virtualized server resources include workload management, automation of virtual environment workload, policy management, and orchestration of server workloads. CA, HP, IBM, VMWare, and Xen all have products for managing network environments. Management of virtual environment focuses on automating functions to reduce the need for manual intervention and creates an optimal environment that complies with the organizational service-level goals and policy [9].

2.4 Virtual Privileges

Operating systems are developed to run directly on bare-metal hardware and have the privileges of owning the computer system [3]. In a virtual environment, the guest OS must maintain these privileges while adding the virtual layer. Virtualization regulates this access to physical hardware with one of two privilege mechanisms. The x86 architecture utilizes Protection Rings, while Linux and other variances of Unix use Domain privileges.

2.4.1 Protection Rings.

Protection rings are used to isolate the OS from untrusted user applications. In an x86 architecture, there are four different levels (rings) that range from 0 to 3. These levels determine which processor commands are executed. Ring 0 is where the OS kernel operates and has the highest level privilege. Ring 0 also has unrestricted access to physical resources. Rings 1 and 2 are allocated for device drivers. All other code, such as applications, typically operate in ring 3 with less privileges and cannot directly access any physical resources [10]. Figure 7 displays the different privilege levels with corresponding x86 architecture components.

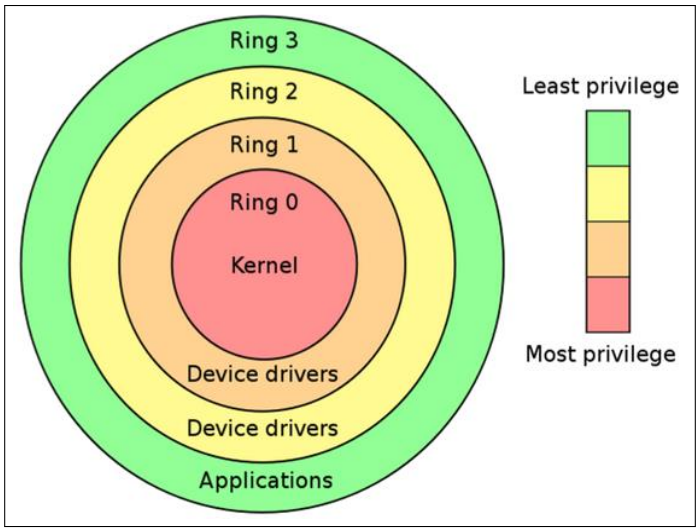


Figure 7. The Protective Rings [11]

The separation of physical and virtual rings provide virtual machines the environment similar to physical machines. Figure 8 shows the difference between a physical and virtual machine in regards to protective rings. An OS on a physical machine operates directly on top of the hardware in ring 0. A VM OS typically operates above ring 0 where the hypervisor or VMM resides. The VMM manages the interaction between VM and hardware resources.

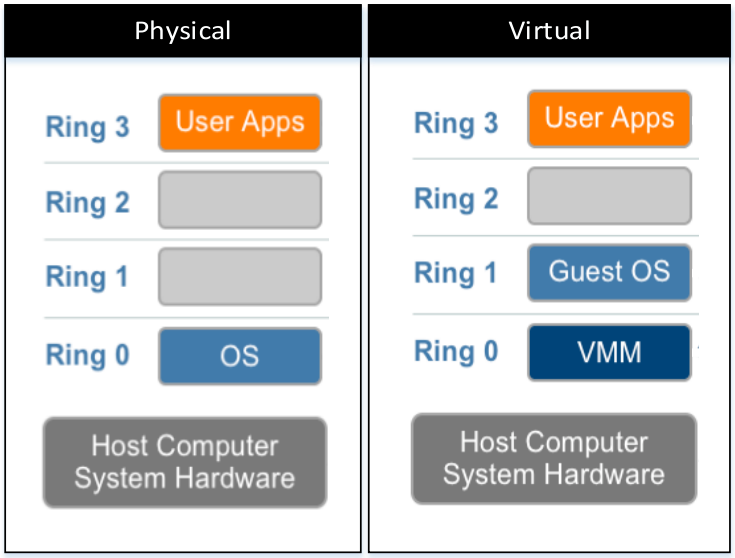


Figure 8. Physical and Virtual Protective Rings [3]

2.4.2 Domains.

Domains are VMs running a Linux OS and instantiated on a virtual host within which a guest OS executes. Domain 0 (Dom0) is a privilege domain started by the Xen hypervisor at boot time. Dom0 is also referred to as the driver domain, the privileged domain, or the control domain [12]. Dom0 is responsible for running all device drivers for the hardware. Domain User (DomU), also known as guest domain, is an unprivileged domain that has no special access to hardware and is assigned to guest machines on the virtual host. The kernel for each DomU comes from the Dom0 filesystem [13]. The DomU frontend for device drivers connects to the backends of the Dom0 drivers. Application and user requests are sent directly to physical resources (e.g., CPU and RAM) for processing only. The interactions between Dom0, DomU, hypervisor, and physical hardware is shown in Figure 9.

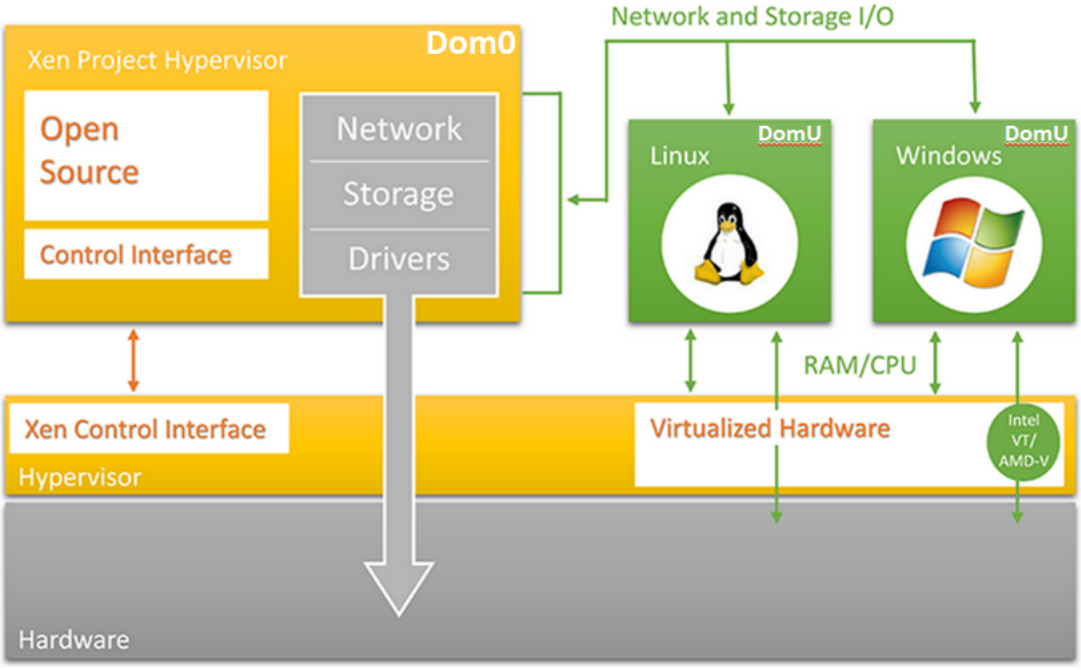


Figure 9. Domain - Xen Architecture Design [14]

2.5 Virtualization Techniques

Hardware management in a virtual environment can be a difficult task. Administrators must understand the difference between physical and virtual hardware when managing the environment. There are several methods developed on the configuration of OS and hypervisor to emulate physical hardware. Hypervisors handle privileged level instructions differently depending on which method is used. The three most commonly used virtualization techniques are Full Virtualization, Paravirtualization, and Hardware Assisted Virtualization (HAV).

2.5.1 Full Virtualization.

Full virtualization isolates the VM OS from the physical hardware using a virtualization layer. Binary translation is used to intercept VM OS code, translate the code, and then execute code using the host machine hardware. The binary image of the OS is manipulated at runtime while user-level code is directly executed on the processor for high performance virtualization [15]. The VM OS does not require any modifications due to OS being unaware of being virtualized. Full virtualization allows the virtual host to virtualize any x86 OS and offers the best solution for security and portability [3]. Figure 10 displays the full virtualization protective ring layers and how they communicate. The binary image of the VM OS in ring 1 communicates with physical hardware through a hypervisor in ring 0. It is the hypervisors responsibility to perform translation of binary code, and allocate hardware resources. Ring 3 has direct access to physical hardware for processing of user applications and requests but cannot perform any modifications.

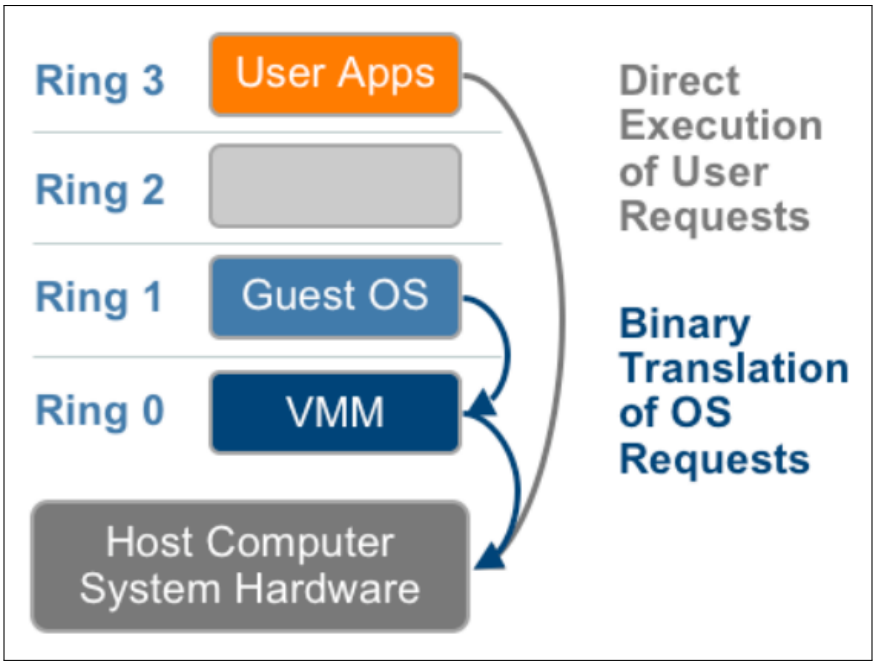


Figure 10. Full Virtualization [3]

2.5.2 Paravirtualization.

Paravirtualization in the English term means "alongside virtualization". It refers to the communication directly between hypervisor and VM OS kernel for improved performance. In paravirtualization, the OS knows it is virtualized and only a portion of the system resources are abstracted. Modification to the OS kernel is used to replace non-virtualizable instructions with hypercalls which communicate directly with the hypervisor [3]. The hypervisor provides hypercall interfaces for other critical kernel operations like interrupt handling, memory management and time keeping [16]. The concept of paravirtualization has a lower virtualization performance overhead than full virtualization but portability is an issue [3]. Figure 11 displays the paravirtualized ring layers. The hypervisor and paravirtualized VM OS work in conjunction with each other within ring 0. Ring 3 operates the same as full virtualization giving user applications and requests direct access to physical hardware for processing only.

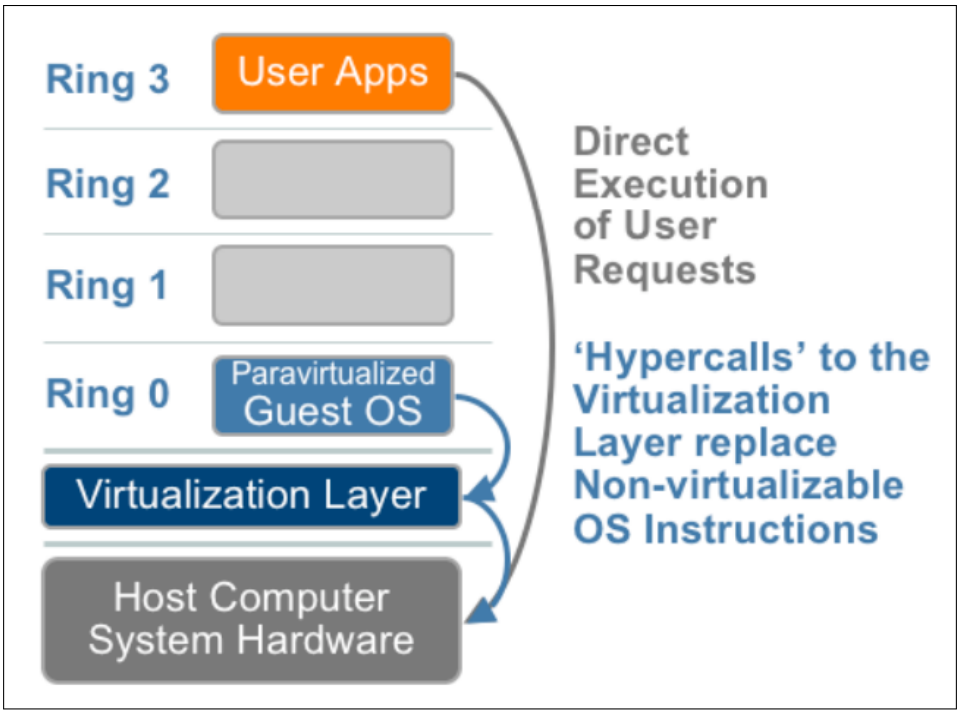


Figure 11. Paravirtualization [3]

2.5.3 Hardware Assisted Virtualization.

Hardware Assisted Virtualization minimizes the host system involvement of managing privilege instructions. Privileged as well as sensitive calls are set to automatically trap to the hypervisor and handled by hardware, removing the need for either binary translation or paravirtualization. First generation of HAV technology includes Intel Virtualization Technology (VT-x) and Advanced Micro Devices Virtualization (AMD-V) which did not become available until 2006 [3]. Both hardware technologies target privileged instructions with a new CPU execution mode which allows the VMM to run in a new root mode below ring 0 as shown in Figure 12. The new root mode makes ring 0 available for use by only unmodified VM OSs. Ring 3 operates same as previous virtualization techniques.

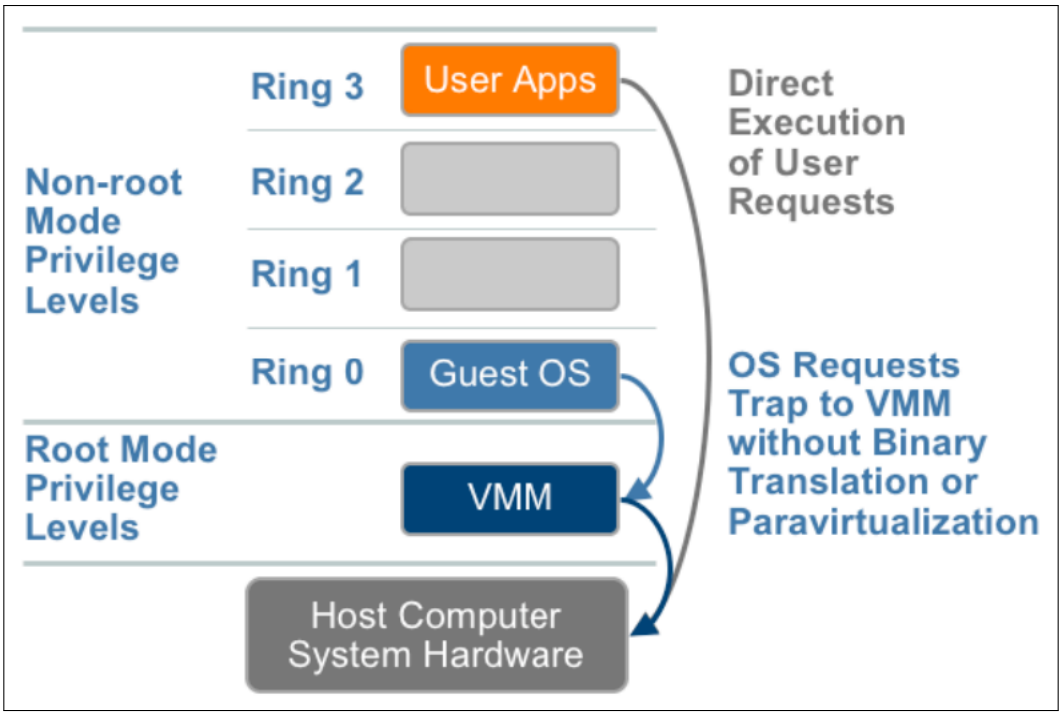


Figure 12. Hardware Assisted Virtualization [3]

2.6 Virtual Security

Virtual security in a networking environment brings additional attack vectors that need to be guarded against. Isolation is the biggest key to virtual network security. Production traffic should be separated from management traffic by allocating a separate physical connection and virtual switch.

Security threats to a virtualized environment fall into a number of categories, centered mostly on the hypervisor. The categories are Hyperjacking, VM escape, VM hopping, and inter-VM traffic [17].

Hyperjacking is the process of injecting a roque hypervisor and taking control of the virtual host. A popular example of hyperjacking was demonstrated in 2006. Joanna Rutkowska used hardware virtualization to developed the undetectable rootkit called "Blue Pill" which executed within a VM to create the Blue Pill hypervisor [18]. Blue Pill was used to implement a network backdoor, keylogger, and other malware.

VM escape is defined as an exploit in which the attacker runs code on a VM that allows an OS running within the VM to break out and interact directly with the hypervisor [19]. This violates the isolation concept discussed earlier which opens the host OS, VMs, and network to further attacks. A well-known VM escape vulnerability is called "Venom" [20]. Venom uses the Floppy Disk Controller (FDC), which is added to new VMs by default. Even if the virtual floppy drive is disabled, the vulnerable FDC code remains active and exploitable by attackers on Xen or Quick Emulator hypervisors. The vulnerability may allow attackers to escape from the confines of an affected VM guest and obtain code-execution access to the host which potentially give adversaries significant elevated access to the host local network and adjacent systems [20]. Figure 13 shows the steps how Venom maneuvers within a host and throughout a network. The first step is to attack the virtual floppy drive code on a VM. If successful, the attacker escapes the confines and moves laterally within the host or network [20]. In 2012, the US-CERT published a warning about another VM escape vulnerability. The vulnerability is a malware called "ring3 attacker" which attacked ring0 (kernel) and caused privilege escalation [21].

VM hopping, also referred to as hyper or guest jumping, is the process of moving from one VM to another using vulnerabilities within the infrastructure or hypervisor [17]. Due to the ability to hop and control other VMs, this attack violates the heart of "confidentiality, integrity and availability" security [22].

Inter-VM traffic is the focus of this research and further discussed in Section 3.3. The security issue with inter-VM traffic is the inability to gain full visibility of packets being passed within the same host. Unlike virtual networks, traditional networks have the ability to monitor and send all network traffic to a security device for intrusion detection.

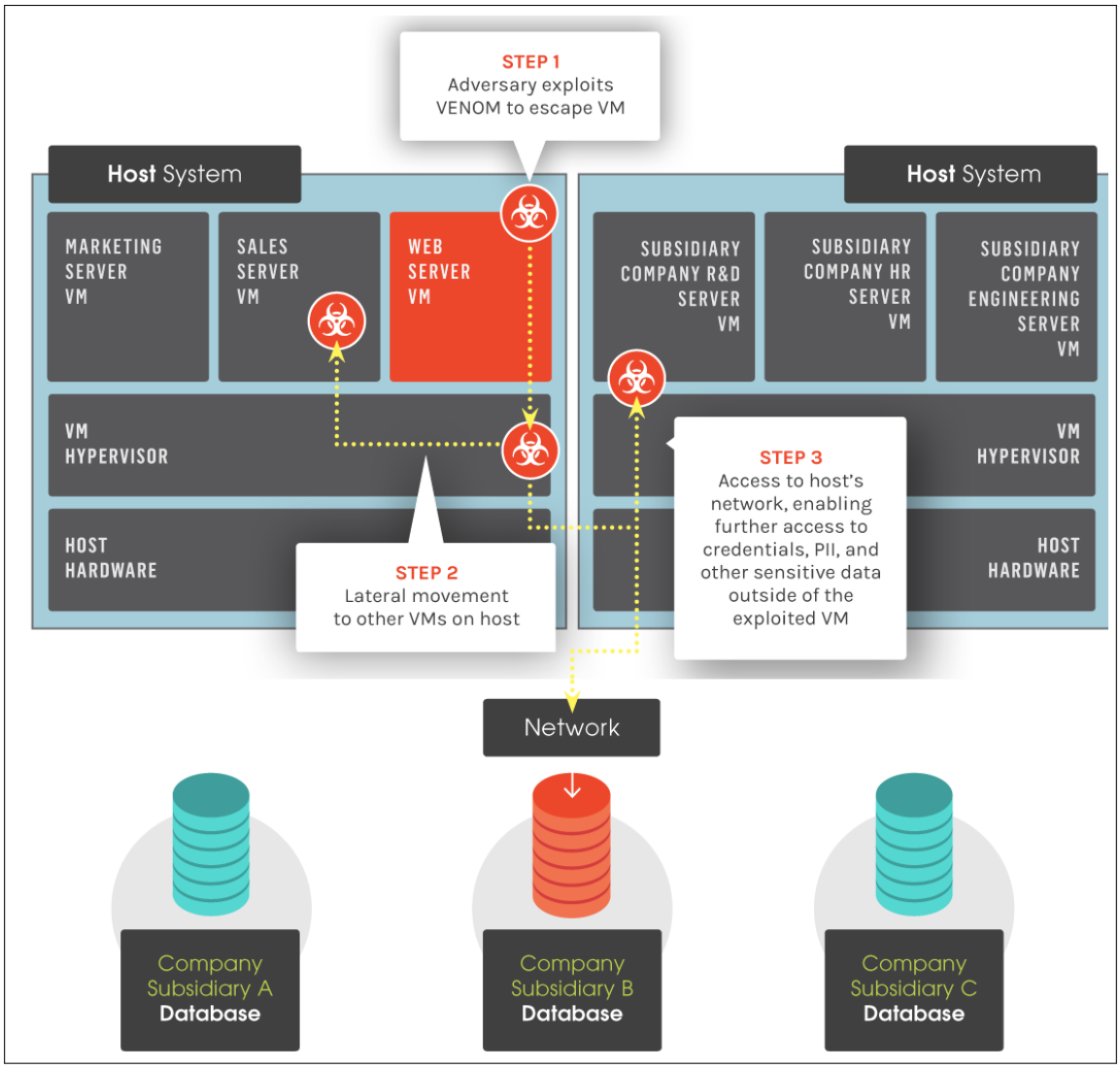


Figure 13. Venom Vulnerability - VM Escape [20]

2.7 Related Research

The increased use of cloud computing and server consolidation has made inter-VM traffic security more relevant. Research on inter-VM traffic includes using packet inspection, comparing virtual platforms, and performance testing.

2.7.1 Inter-VM Visibility in a Cloud Environment.

Benzidane et al. used a packet inspection approach to perform research on inter-VM traffic [17]. The focus was to manage inter-VM traffic, analyze it, and prevent non-compliant packets from being passed. The key component and centerpiece of inter-VM traffic security for packet inspection is the master and slave agents. Credentials and slave agent synchronization is maintained by the master agent. Slave agents place a frame tag within the IP packet which contains data origin authentication, integrity, and sending VM information. Figure 14 shows the construction of a frame tag within an IP packet. The receiving VM agent performs the packet inspection process to determine if the packet is compliant to the rule set allocated by the master agent. The complete process is outline in Figure 15. The proposed research has not been implemented in a wide-scale environment and future works are scheduled to use a dynamic approach for generating packet inspection rules [17].

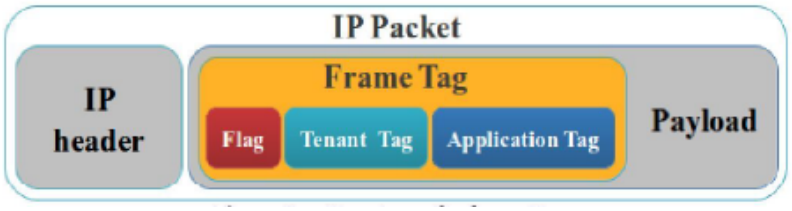


Figure 14. Frame Tag [17]

2.7.2 Communication-Aware Inter-VM Scheduling.

Guan et al. investigated the manner in which the Xen hypervisor or VMM scheduler handles the latency of inter-VM traffic [23]. The baseline VMM scheduler manages the physical resources and is agnostic to the communication performance between VMs. Communication-Aware Inter-VM Scheduling (CIVSched) was developed to work in conjunction with the VMM scheduler to decrease the latency of communica-

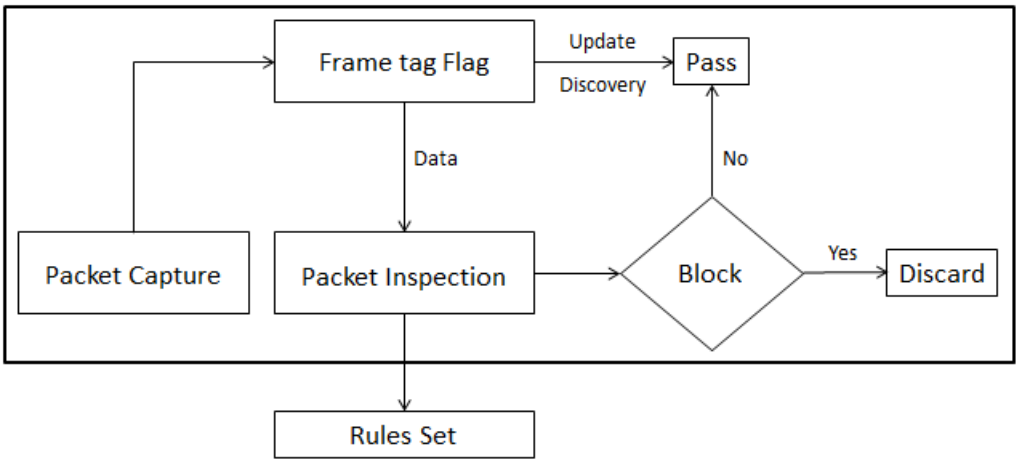


Figure 15. Packet Inspection Process [17]

tion between the hypervisor and VM. The five modules highlighted in Figure 16 make up the components of CIVSched. The CivMonitor is a core component that extracts data from network traffic and coordinates with the CivScheduler and PidScheduler for processing. The AutoCover module discovers co-located VMs and maintains a mapping table for faster lookup. The run queue within the hypervisor is then reordered by the CivScheduler for inter-VM traffic. The results of this technique lowered the response time between two inter-VMs but is limited to a virtual host with a single CPU. Future work is scheduled for multi-CPU systems [23].

2.8 Summary

This chapter introduces the main concepts of virtualization, covering the hypervisor, VMs, virtual networks, and virtual traffic. The Kusnetzky Group Model of Virtualization offers a reference to better understand the complexity of virtualization. Virtual privilege mechanisms use protective rings and domains to offer protection between the VM OS and hardware access. The different techniques of full virtualization, paravirtualization, and HAV handle the difficult tasks of virtual hardware management. Hyperjacking, VM escape, and VM hopping are security threats that

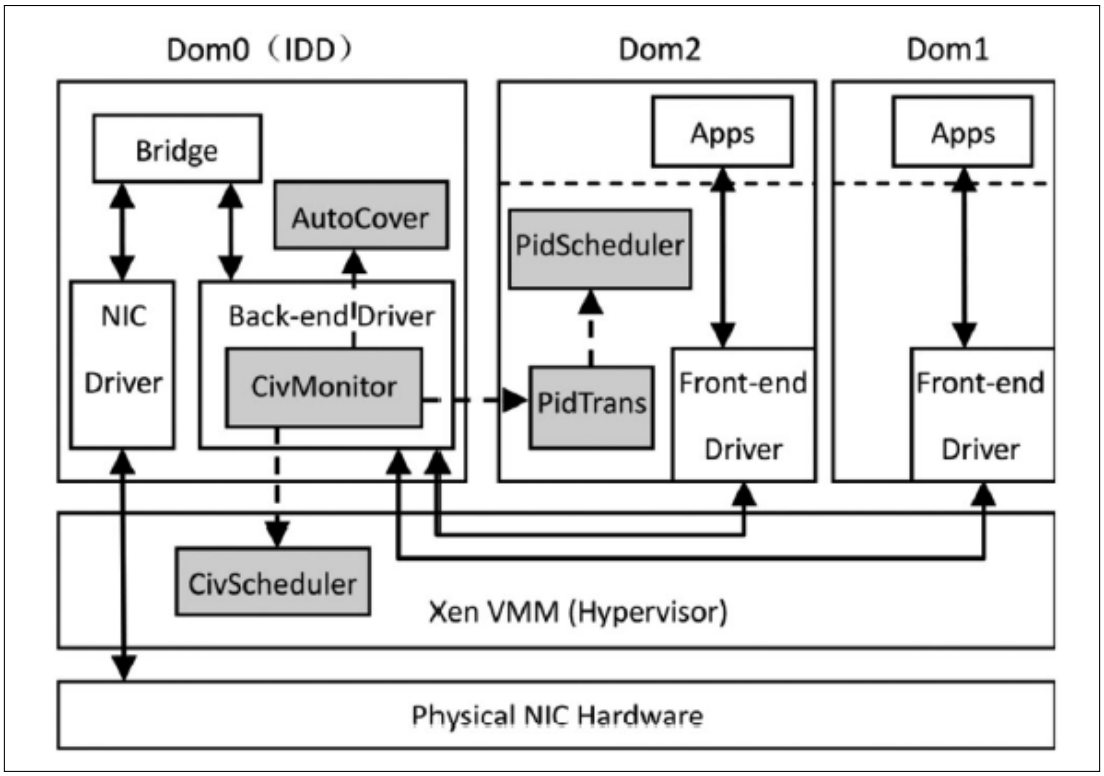


Figure 16. CIVSched Architecture [23]

violate the concept of isolation. Related research for inter-VM traffic includes using agents for visibility in a cloud environment and methods to improve latency between the VMs and hypervisor.

III. Methodology

This chapter presents the methodology used for evaluating the visibility and performance of inter-VM traffic within a virtualized environment. Section 3.1 defines the specific problem to be address during this research. Section 3.2 describes the goals and hypothesis for the research. Section 3.3 renders an overview of inter-VM traffic and basic understanding of the communication path. Section 3.4 details the factors related to the experiment, and Section 3.5 outlines the process used to conduct the experiment.

3.1 Problem Definition

This research evaluates and compares server virtualization solutions that leverage manageability and security for the DoD networks. Server virtualization has changed the way network traffic is passed and monitored. In a traditional network, traffic is forwarded through physical switches and managed by network administrators. In a virtualized environment, network traffic is passed through a virtual switch installed on a virtual host server managed by server administrators. The movement of networked components from physical to virtual environments is an added challenge for network administration and security. Virtual traffic passed within the virtual host is processed by the hypervisor but does not offer visibility to an IDS or network management system. The network administrator no longer maintains complete control of network traffic and must rely on system administrators to expose inter-VM traffic to the physical network. Technology provides several avenues to capture inter-VM traffic but implementation of a secure solution requires additional research. This research focuses on maintaining security for both physical and virtual networks while also managing virtual network traffic.

3.2 Goals and Hypothesis

The objective of this research is to evaluate the different methods that provide visibility of inter-VM traffic. The goal is to expose inter-VM network traffic to external applications for security and administration. Visibility and the ability to manage traffic is a large determining factor for the security level within the virtual network. There are many vendors that furnish virtualization frameworks; this research evaluates the performance of VMWare ESXi 5.5 and Citrix XenServer 6.5 in passing transport-layer traffic through multiple virtual switch configurations. VMWare ESXi is a proprietary product that uses the vSphere hypervisor, and Citrix XenServer is an open source product that uses the Xen Project hypervisor.

The first research goal is to construct a network environment that uses DISA specifications to assess optimum visibility into inter-VM traffic on each hypervisor and virtual switch. The second goal is to evaluate performance of inter-VM and external traffic within each virtual network configuration. The overall goal is to assist in the DoD decision-making process by determining which virtual framework provides higher visibility of inter-VM traffic and offer performance levels for each virtual network.

The hypothesis for this research is that visibility into inter-VM traffic is dependent upon the type of hypervisor and virtual switch used. The expectation is that each hypervisor has multiple methods available and may give different results of visibility for inter-VM traffic. Since distributed switches involve larger networks, it is hypothesized that there are more options available than standard switches when capturing inter-VM traffic.

3.3 Approach

This section outlines a high-level approach for using different hypervisors, virtual switches, and protocols to obtain the visibility of inter-VM traffic. This section describes the basic configuration used to fulfill the research goals.

Administration of a virtual network requires a thorough understanding of inter-VM traffic and how packets are passed internally. Figure 17 displays the basic inter-VM traffic pattern within a single host. VM(X) initiates traffic by sending packets to the hypervisor. The hypervisor then evaluates the packet header for destination information and determines if the traffic should be sent internally or to the physical network. If the destination is internal, traffic is passed to the destination VM(Y) using a VMWare vSphere Standard vSwitch (vSS) or XenServer Standard Open vSwitch (OVS) which bypasses the physical network. At this point, all traffic is only visible to the source VM(X) and destination VM(Y), virtual switch, and hypervisor. The lack of inter-VM traffic visibility causes both administration and security issues.

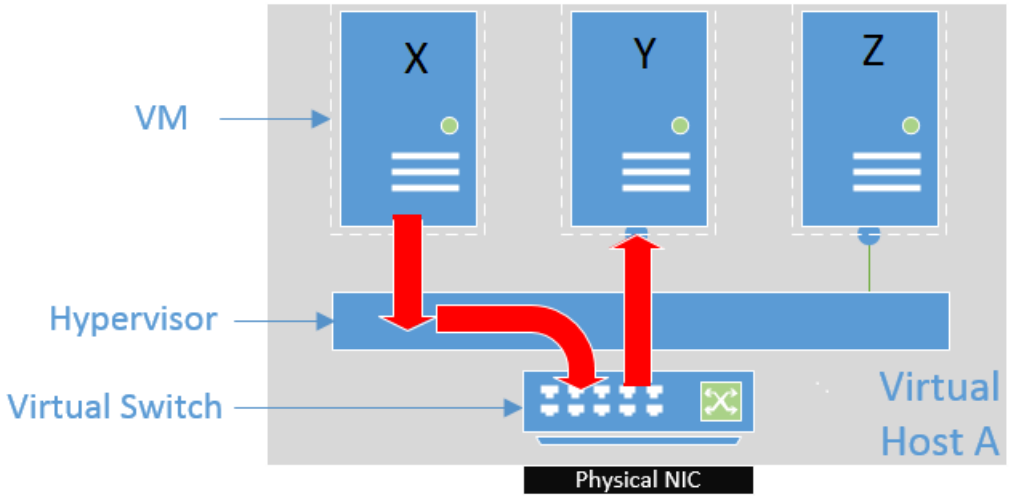


Figure 17. Inter-VM Traffic

As shown in Figure 18, transferring network connections from a standard vSwitch to a distributed vSwitch gives administrators the ability to oversee both physical

and virtual network traffic. Both VMWare vSphere Distributed Switch (vDS) and XenServer Distributed Virtual Switch Controller (DVSC) give network administrators more control and visibility of network traffic by passing virtual traffic to network management tools. Traffic passed by VM(X) is sent to the distributed switch for processing before being sent to VM(Y). The distributed switch manages the connection and provides additional features for traffic collection such as Switch Port Analyzer (SPAN), Remote SPAN (RSPAN), Port Mirroring, sFlow, and NetFlow.

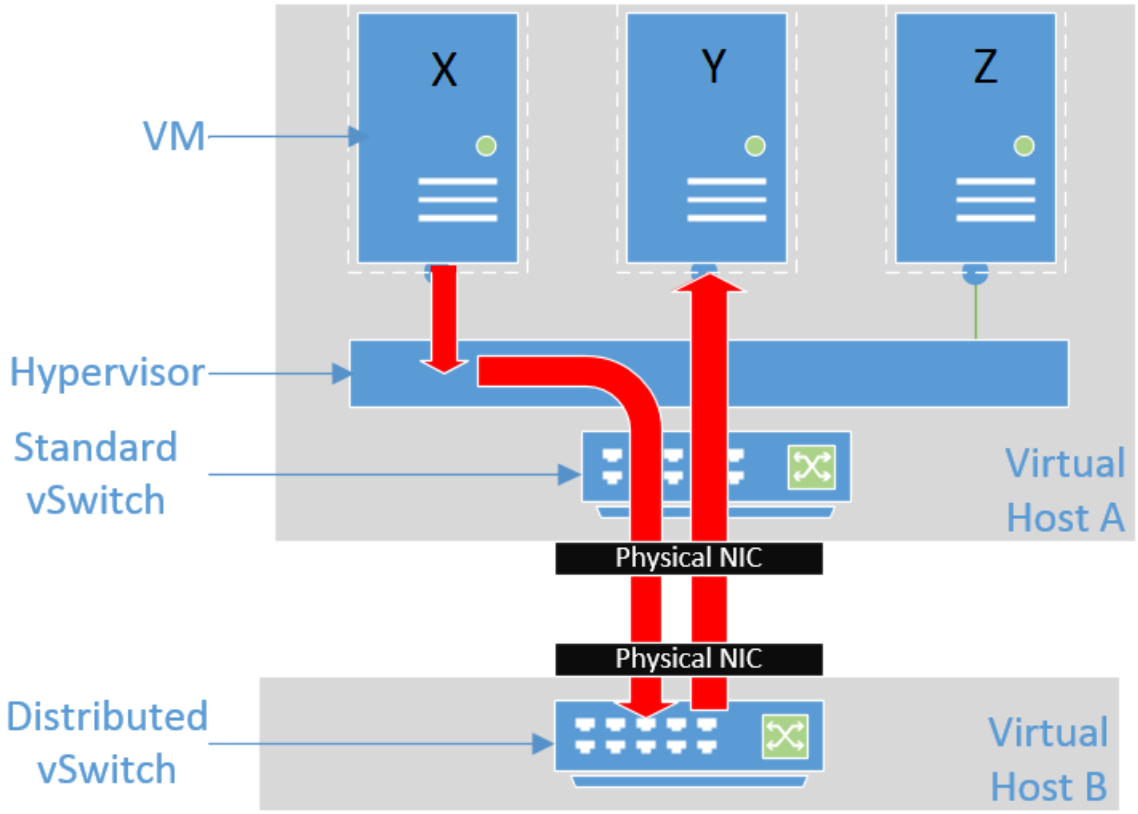


Figure 18. Distributed Virtual Network

3.4 Experiment Factors

There are four experiments conducted using a variety of components and configurations. The key configuration factors are standardized-configured hardware, hyper-

visor with management tools, virtual switch type, and protocol used to send traffic. As shown in Figure 19, the System Under Test (SUT) represents the hardware components of the virtual host and the Component Under Test (CUT) which are the components of interest for the experiment (e.g., hypervisor and virtual switch). As traffic enters the SUT, processes are performed within the CUT which provide traffic visibility and performance outcomes for analysis. The experiment factors are listed in Table 1.

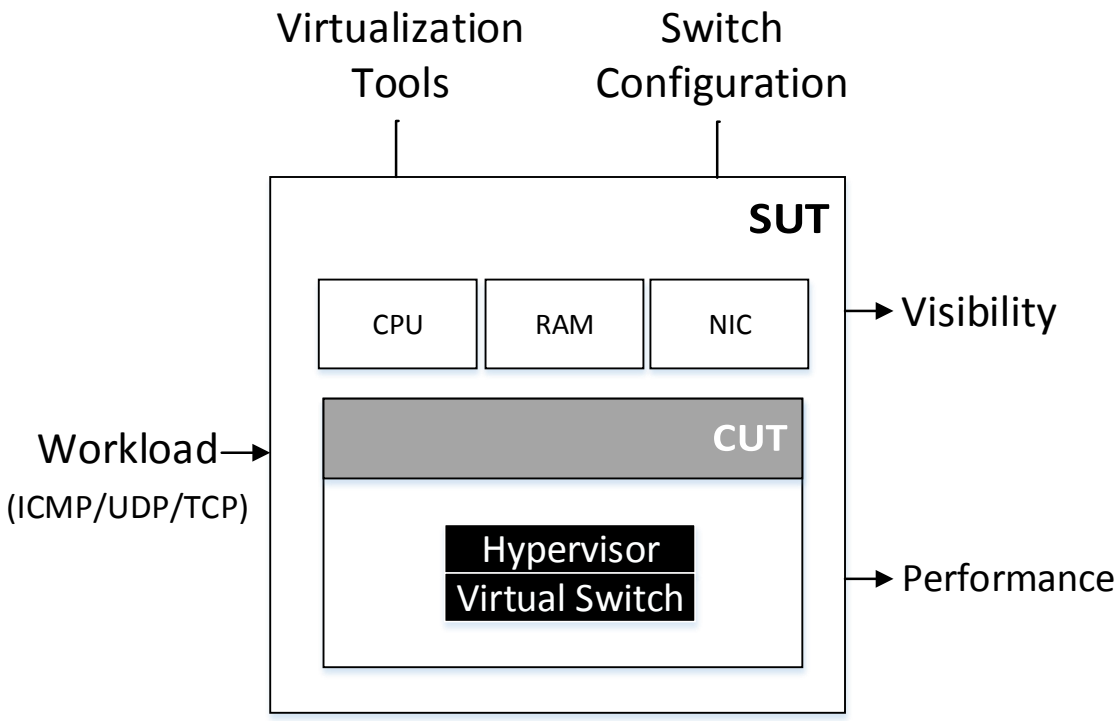


Figure 19. System Under Test

Table 1. Experiment Factors

Factor	**Level**
Hypervisor	ESXi, Xen Project
vSwitch Type	Standard, Distributed
Protocol	ICMP, TCP, UDP
Evaluation Technique	Initial, Packet Capture, Promiscuous, Port Mirroring

The basic metric for comparing visibility of inter-VM traffic is if the traffic is visible to external sources. Performance comparison is based on a collection of duration times of TCP and UDP connections between VMs, measured in microseconds (µs). Comparing the ability to view inter-VM traffic and performance of virtual networking components is based on using two products, VMWare ESXi and Citrix XenServer. The CUT for each experiment is listed below.

1. ESXi Standard Network (vSS)
2. Xen Project Standard Network (OVS)
3. ESXi Distributed Network (vDS)
4. Xen Project Distributed Network (DVSC)

3.4.1 Physical Infrastructure.

The experiment consist of three physical hosts connected to a single switch. Two physical hosts support the production and management traffic, while the third is used for visibility of network traffic. To obtain accurate results, the physical hosts must meet the specifications required to support the virtualization framework. VMWare ESXi and Citrix XenServer maintain different minimum requirements for their virtual platforms. Table 2 compares the host minimum requirements for each vendor.

Table 2. Host Minimum Requirements

Product	**VMWare ESXi 5.5**	**XenServer 6.5**
CPU	Dual Core 64-bit x86	Single Core 64-bit x86
NIC	1GigE	100Mbit/s
Memory	4GB	2GB
Disk	10GB	16GB

Physical resources available for the experiment include three servers, a laptop, and a manageable switch. The specifications for each physical component is listed in Tables 3-5.

Table 3. Physical Host Specifications

Type	Specification
Model	Dell Poweredge R415
CPU	AMD Opteron(tm) 2600MHz - Dual Core
NIC	Broadcom NeXtreme II BCM5716 - 1GigE
RAM	64GB - DDR3 1600 MT/s
Disk	450GB

Table 4. Management Laptop Specifications

Type	Specification
Model	Dell Precision 4500
CPU	Intel i7 M640 2.8GHz
NIC	Intel 82577LM - 1GigE
RAM	8GB
Disk	465GB

Table 5. Physical Switch Specifications

Type	Specification
Model	Cisco Nexus 3048TP
Switching	Layer 2 and 3
Ethernet	48 fixed 10/100/1000-Mbps ports
Management	Two 10/100/1000-Mbps ports
Console	One RS-232 serial port

3.4.2 Physical Switch Configuration.

Several tasks are required to configure the physical switch for operations. The switch configuration tasks include setting up a Virtual Local Area Network (VLAN), VLAN Gateway, trunk or access ports, and SPAN monitoring. PuTTy telnet client [24] provided access with direct connection to the console port using an RS232 cable,

or Secure Shell (SSH) once the gateway is configured. Appendix A provides step-by-step commands for implementing each task. Below is a list of simple commands for verifying the physical switch configuration.

- `show vlan name` <`VLAN Name`> (VLAN status and ports)
- `show interface vlan` <`VLAN #`> (VLAN gateway status)
- `show interface brief` (Physical switch interfaces, mode: trunk or access)
- `show monitor session` <`Session #`> (Only if SPAN is required)

3.4.3 Hypervisor with Management Tools.

The hypervisor and management tools work closely together and are core components for any virtualized platform. Each platform has its own version of hypervisor and management tools to support the physical resources of the virtual environment. The management tools render a detailed view of the network while assisting in testing the hypervisor.

As discussed in Section 3.2, VMWare includes the vSphere hypervisor and Citrix uses the Xen Project hypervisor. Both are Type 1 bare-metal hypervisors that provide the virtualization environment for their respective physical host. A major difference between the two hypervisors is the virtualization techniques discussed in Section 2.5. vSphere applies full virtualization while XenCenter employs paravirtualization. Table 6 compares some of the key characteristics of each hypervisor.

The ability to manage the hypervisor within a virtualized environment is done either through a Command Line Interface (CLI) or Graphical User Interface (GUI). The options to manage VMWare using CLI include PowerCLI for Powershell, vSphere Management Assistant (vMA) for perl, or the built-in ESXi command shell. Citrix developed the Xen Application Programming Interface (XAPI) toolstack that provides

Table 6. Hypervisor Specification [25]

Product	vSphere	Xen Project
Max Host	512	500(Win)/650(Linux)
Max CPU per Host	320 (logical)	160 (logical)
Max RAM per Host	6TB	1TB
Max RAM per VM	1TB	192GB
OVF Support	Yes	Yes
Guest OS Support	Comprehensive	Good

"xe" for CLI management. The GUI management tools used for this experiment are VMWare vCenter Server and Citrix XenCenter Server. vCenter and XenCenter both offer a centralized location to view and manage their perspective virtual environments. vCenter is installed using an OVA and is a separate deployed VM. XenCenter is an executable that is installed on the management laptop. Both provide similar functionality as it pertains to management.

Table 7. Management Server Specification [25]

Product	vCenter	XenCenter
Virtual and Physical	Limited	No
Cross Vendor Mgmt	Limited	No (native)
Browser Based Mgmt	Yes	No
Security	ESXi Firewall	Basic (Netscaler)
P2V	Stand-alone only	No

3.4.4 Virtual Machine Configuration.

The VM baseline emulates systems placed on a DoD network. Guidelines for creating and implementing the OS images are found on the DISA Information Assurance Support Environment (IASE) website [26]. The IASE maintains the DoD Secure Host Baseline Repository (SHB) as a framework to simplify deploying systems that are compliant with Information Assurance requirements [26]. Table 8 outlines the basic configuration for VMs created using the SHB images.

Table 8. Virtul Machine Specification

Name	Windows OS (Enterprise)	CPU's	RAM	Disk Size
W2K8SP1-DC	Server 2008 R2 - SP1	2	16GB	40GB
W2K8SP1-DHCP	Server 2008 R2 - SP1	2	16GB	40GB
WIN7SP1-11	WIN7 - SP1	1	8GB	60GB
WIN7SP1-12	WIN7 - SP1	1	8GB	60GB
WIN7SP1-13	WIN7 - SP1	1	8GB	60GB
WIN7SP1-14	WIN7 - SP1	1	8GB	60GB
WIN7SP1-MGT	WIN7 - SP1	1	16GB	60GB
WIN7SP1-SPAN	WIN7 - SP1	1	4GB	60GB

Once created and configured, an OVF is generated from each image to ensure all systems are identical for both virtual infrastructures. Each hypervisor required additional tools for proper integration with the VM. To integrate the VM, it must have VMWare tools or XenServer tools installed depending on the hypervisor. These tools provide the required virtualization device drivers and a management agent.

3.4.5 VMWare ESXi Standard and Distributed Switch.

The VMWare ESXi virtual network is built upon a vSS and has been around since the early implementation of ESX. The vSS is still popular today due to the simplicity it offers. The vSS is responsible for routing traffic internally between VMs on a single host and for communications to external networks. Small computing environments typically utilize vSS as their networking infrastructure. VMWare recommends separating virtual networks by creating a different vSS and connecting each to its own physical NIC. As shown in Figure 20, the initial vSS consist of a VMkernel network (vSwitch0) and virtual NIC (vmk0) for host management. VM Network (vSwitch1) is added for VM traffic. vSwitch0 uses vmnic0 physical adapter, and vSwitch1 connects to vmnic1 physical adapter for connectivity to the physical network.

A VMWare ESXi vDS is a vSwitch that is co-located with a vCenter Server on a virtual host. The vDS is used predominately within an enterprise-level network and

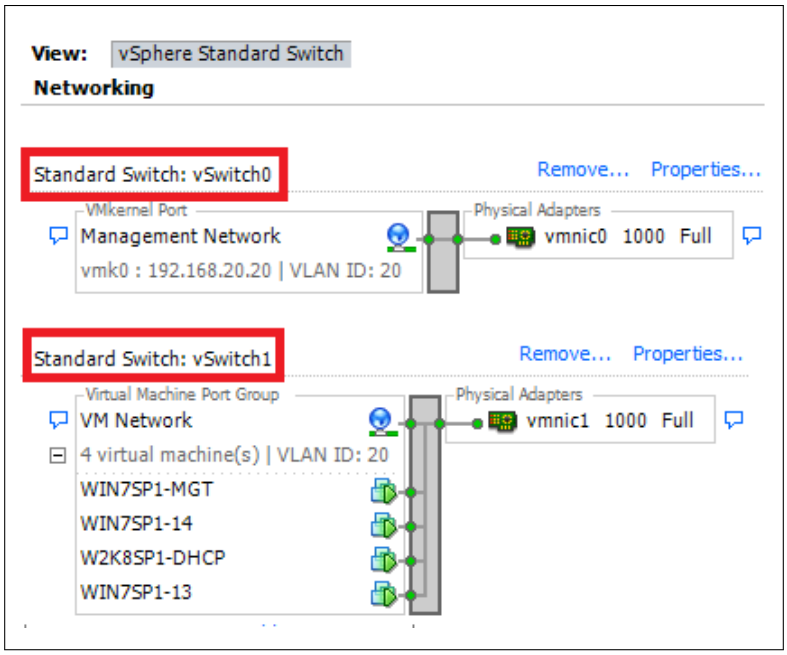


Figure 20. ESXi Standard vSwitch

provides the additional capability of NIC teaming, Link Layer Discovery Protocol (LLDP), VLAN-based SPAN (VSPAN), Link Aggregation Control Protocol (LACP), Private VLAN, NetFlow, and Port Mirroring. Administration of the vDS can be performed through the VMWare vSphere client but the web client must be used to access full functionality available. A vDS manages network traffic between any physical switch, vSS, or VM that is connected to the distributed switch. As shown in Figure 21, two distributed switches called *MGT_DSwitch* and *VM_DSwitch* are created to separate the management traffic from the production traffic. Physical adapter management is accomplished through the DSwitch-DVUplink port groups. DVUplink port groups consolidate physical adapters and offer additional capabilities (e.g., NIC teaming or fail over). The separate *Monitor* port group is created on the production network for port mirroring functionality.

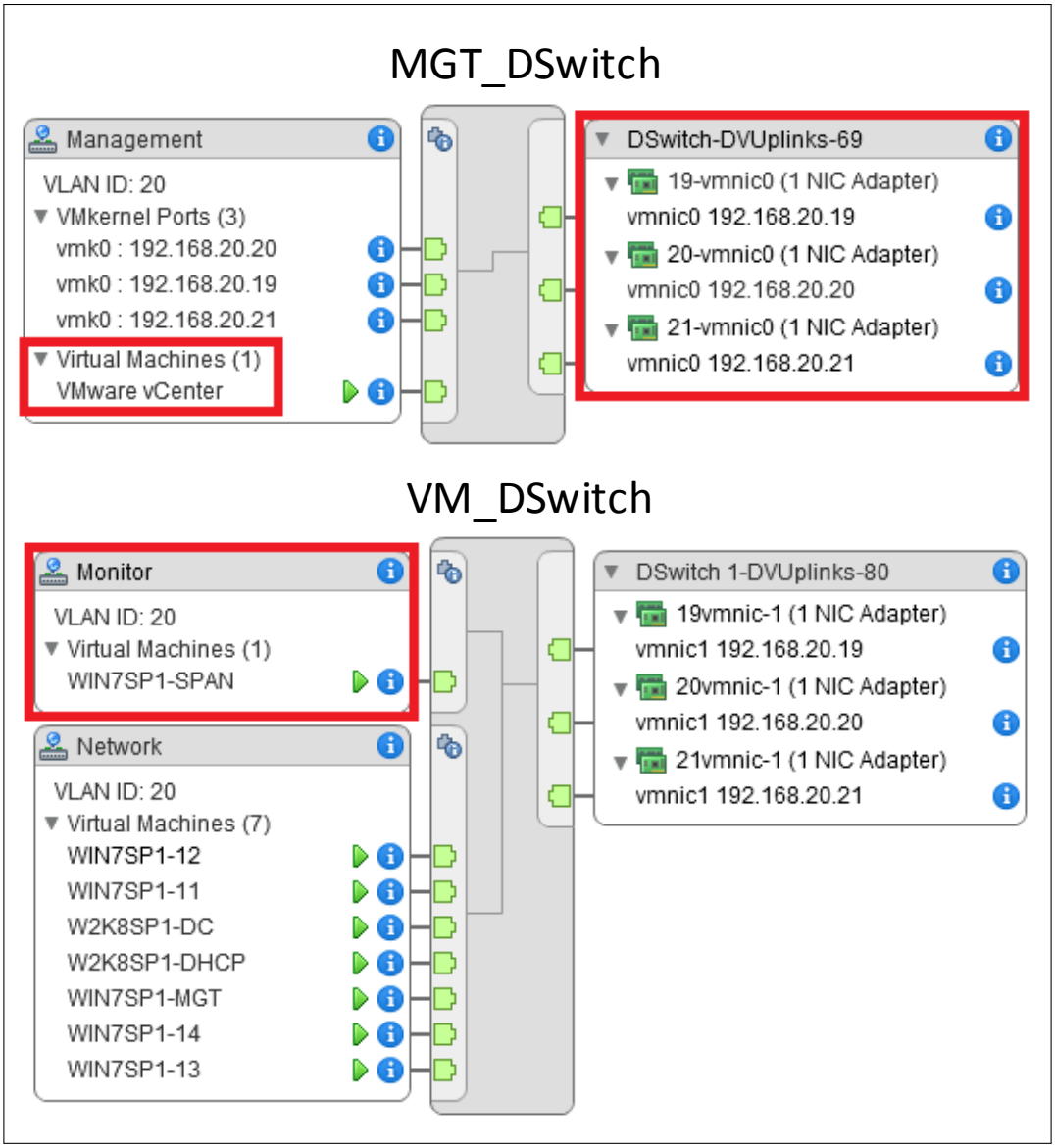


Figure 21. ESXi Distributed vSwitch Configuration

3.4.6 Citrix XenServer Standard and Distributed Switch.

The Citrix XenServer virtual network has two network stack options for standard virtual switching (OVS and Linux bridge) with OVS being the default. Determining the current vSwitch is accomplished by typing "`cat /etc/xensource/network.conf`" using the CLI. OVS is a Linux-based platform that gives production-level switching to the virtual environment and supports advanced features discussed in Section 3.3.

Also, the Linux bridge cannot be managed by XenServer. The standard-level OVS is instantiated on a host and is configured as a stand-alone vSwitch.

The initial OVS created during installation consists of a Network0 (Management) and Network1 (Production). Each separate virtual network created is connected to its own physical NIC. Network0 uses NIC0 physical adapter, and Network1 connects to NIC1 physical adapter. VLANs are enabled by configuring separate networks on each physical adapter allowing communications with a physical switch trunk port. Figure 22 displays the XenCenter standard network configuration for both the management and production networks. The virtual host management network configuration is found within the *IP Address Configuration* block. Production traffic is configured using the VLAN network (19-VLAN20-1) added to Network1.

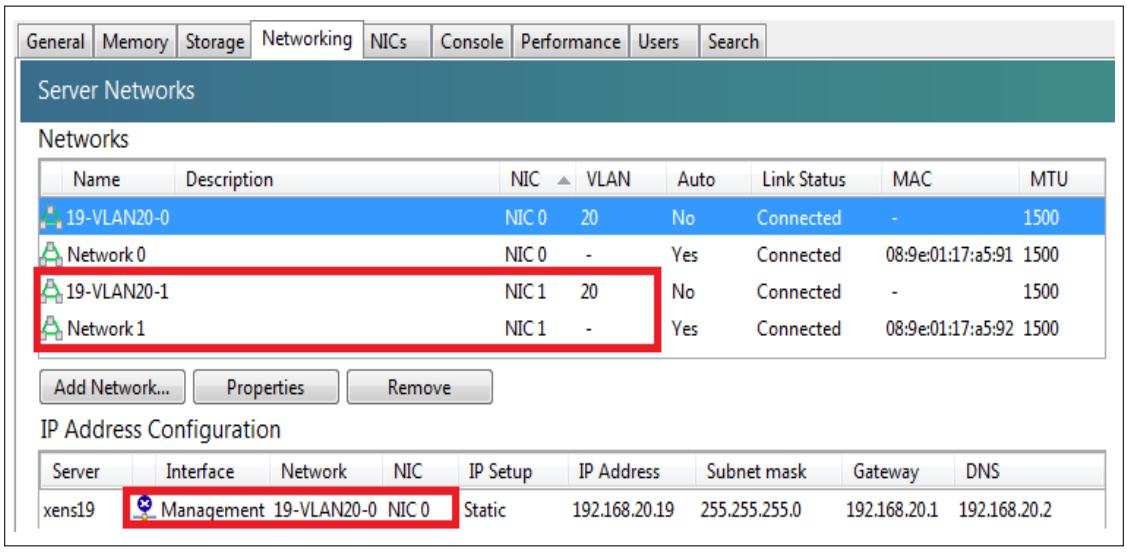


Figure 22. XenCenter Standard Configuration

The distributed-level OVS extends the security and management functionality, such as traffic filtering and Netflow. OVS is designed to manage multiple physical servers similar to VMWare vDS [27]. The OVS for a distributed environment is similar to standard in the ability to manage through the console. The DVSC is used to manage the XenServer distributed network which brings visibility, security, and

control to the XenServer virtual network. The controller provides a centralized server to manage the behavior of multiple individual vSwitches such that they appear as a single vSwitch[28]. Figure 23 shows how DVSC consolidates physical interfaces on separate virtual hosts to create a single virtual network. Consolidation allows VM access to the single network from multiple virtual host.

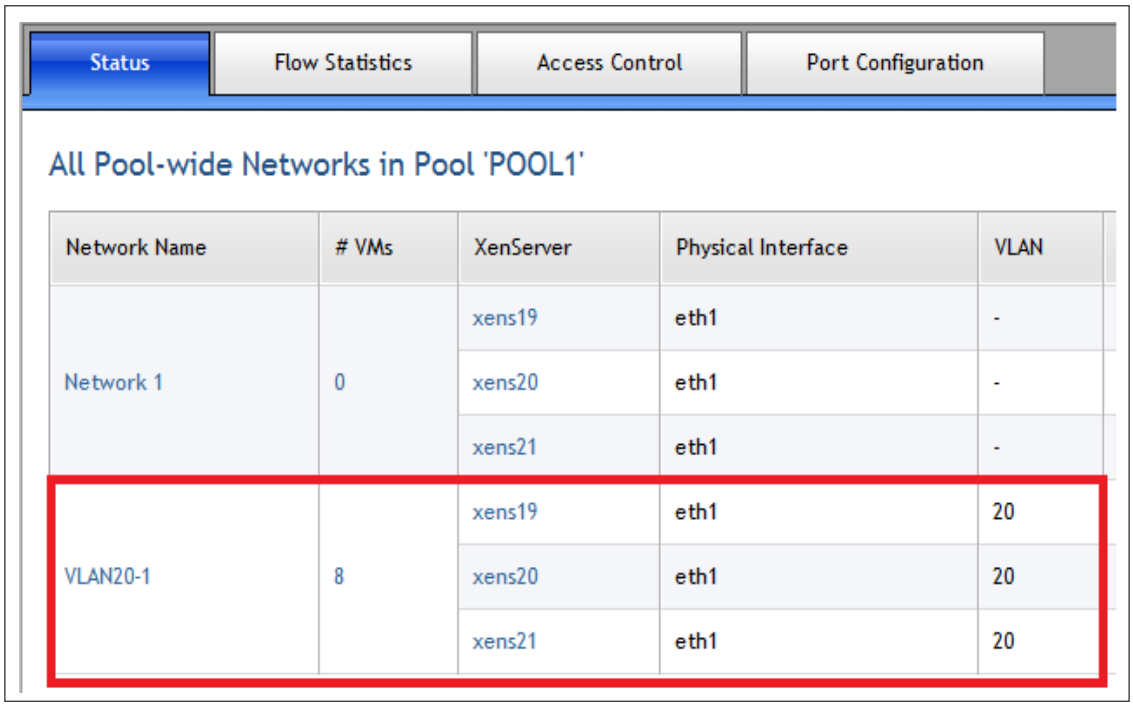


Figure 23. DVSC Network Consolidation

3.5 Experiment Details

This section details the implementation of the experiments. The research virtual environment is developed to resemble a scaled-down model of a DoD domain. Multiple host configurations are designed to create an environment that equally evaluates the factors discussed in Section 3.4. Section 3.5.1 provides an overview of traffic generation and Section 3.5.2 discusses the packet capture methods used. Section

3.5.3 outlines the performance data collection process. Sections 3.5.4 and 3.5.5 discuss standard and distributed configurations respectively.

3.5.1 Traffic Generation.

Generating traffic for the experiments is accomplished at different levels using custom source code executed on the management and test VMs. The objective is to minimize the processing done on the test VMs. Batch files, Powershell scripts, and Python code are placed on each system to create and capture ICMP, TCP, and UDP traffic for evaluation. Traffic is also captured using either a SPAN port on the physical switch or through port mirroring within the hypervisor management tools.

Initiation of all testing begins with the management VM (WIN7SP1-MGT). As annotated in Appendix B, batch files located on WIN7SP1-MGT trigger the start of testing. The batch files contain variables that are passed to the Powershell scripts. The variable data consist of the vendor, protocol used, internal or external, the last octets of test VMs, and number of iterations to perform. The Powershell source code shown in Appendix C performs the majority of tasks during testing by interacting with the domain controller and sending required data to the VMs for processing.

The purpose of the VM batch file code listed in Appendix D is to perform the function sent by the management VM. The types of functions include starting and stopping the WinDump [29] program, running the ICMP test, or sending request to the Python code. The Python source code provided in Appendix E, carries out the instructions to start or stop the protocol servers and initiate protocol testing. The complete process outlined in Figure 24 generates a packet capture file to be viewed using the Wireshark Network Protocol Analyzer [30] for data collection and analysis.

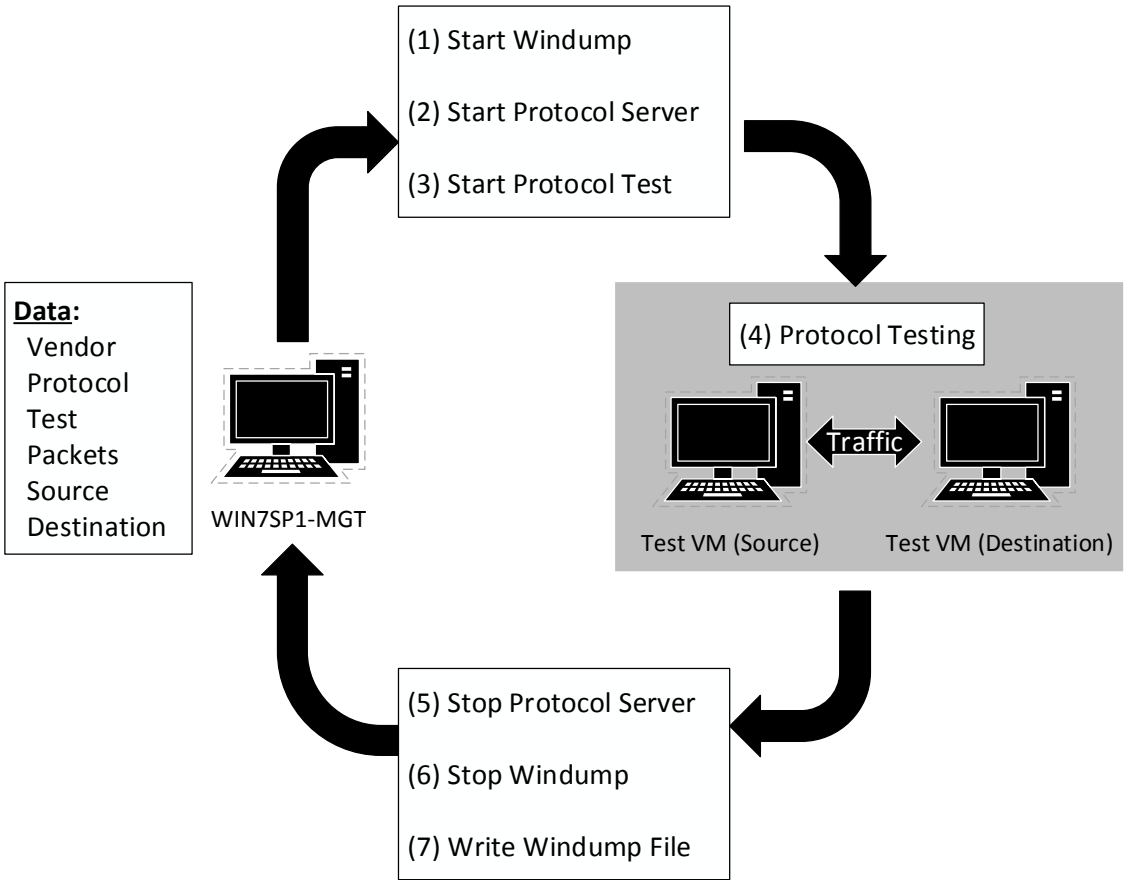


Figure 24. Protocol Testing Process

3.5.2 Traffic Capture.

VMWare developed an enhanced packet capture and analysis tool called *pktcap-uw*. ESXi 5.5 virtual hosts installs the tool by default and replaces the *tcpdump-uw* tool used in previous versions. The *tcpdump-uw* tool only captures packets at the vmkernel interface level while *pktcap-uw* captures at all levels within the hypervisor [31]. The *pktcap-uw* commands must be run using the built-in ESXi CLI. An example of *pktcap-uw* commands used within the experiment are listed below. To access the built-in ESXi command shell, SSH must be enabled on the virtual host.

1. **Display Virtual Switch Info:**

   - net-stats -l

2. **Check for pktcap-uw Processes:**

   - lsof — grep pktcap-uw — awk 'print $1' — sort -u

3. **Stop pktcap-uw Processes:**

   - kill $(lsof — grep pktcap-uw — awk 'print $1' — sort -u)

4. **Start pktcap-uw Capture:**

   - pktcap-uw –switchport 50331669 -o /tmp/WIN7SP1-14.pcap &
     - -switchport (PortNum from command #1)
     - -o (Output file location and name)

XenServer uses the *TCPDUMP* program [32] that allows for packet capture and provides the ability to view inter-VM traffic on a single host. Physical interfaces, virtual bridges, and VIF components, as shown in Figure 25, can be captured using *tcpdump* [33]. The information needed to perform a *tcpdump* capture include the VM name, dom-id, and device number. This data is obtained from a host using the server console or CLI. Below are steps used to perform a *tcpdump* capture on VM named WIN7SP-11.

1. **Get dom-id:**

   - xe vm-list name-label=WIN7SP1-11 params=dom-id

2. **Get device:**

   - xe vif-list vm-name-label=WIN7SP1-11

3. **Run tcpdump:**

   - tcpdump -i vif22.0 -s 0 -w /WIN7SP1-11.pcap

– -i (Interface)

– -s (Include packet header and entire contents of the data payload)

– -w (Write location for pcap file)

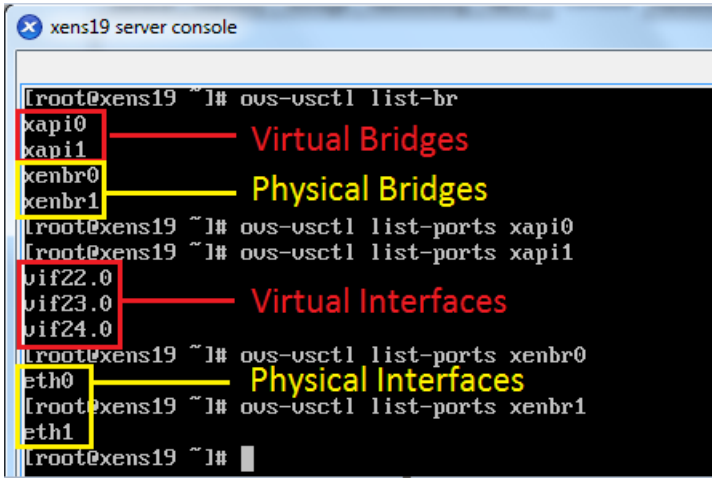


Figure 25. OVS Interfaces

The XenServer OVS also supports packet capture using port mirroring. SPAN must be setup to send all traffic on the physical switch to the network interface of a packet capture host and in turn make the traffic available to the virtual switch. The VM interfaces reside within a XAPI bridge. The XAPI bridge configures and controls VMs on a Xen-enabled host [34]. Packet forwarding is accomplished by setting up port mirroring as shown in the list below [35]. The commands provide a list of bridges and ports available, port mirroring setup, and how to stop port mirroring.

1. **List OVS Bridges:**

   - ovs-vsctl list-br

2. **List OVS Ports:**

   - ovs-vsctl list-ports xapi1

3. **Start Port Mirroring:**

- ovs-vsctl – set Bridge xapi1 mirrors=@m \(Identify bridge used)
- – –id=@v9 get ports vif22.0 \(Identify port used)
- – –id=@m create Mirror name=MyMirror select-all=true output-port=@v9 (Send all traffic to port in previous line)

4. **Stop Port Mirroring:**

- ovs-vsctl clear bridge xapi1 mirrors

3.5.3 Performance Data Collection.

Section 3.5.1 provides the basic guideline for generating traffic for evaluating virtual switch performance. The basic task used WinDump [29] on each VM to collect a packet capture. The packet capture included either TCP or UDP communications between internal and external hosts. Using Wireshark to review captured files, data is gathered from the statistical conversations tool and custom filters. Figure 26 shows the custom filters used to parse captured traffic. Four different VMs are used to send each traffic type to ensure uniformity between host. For example, internal traffic is captured from both WIN7SP1-11 to WIN7SP1-12 and WIN7SP1-13 to WIN7SP1-14 on the same host in which they are contained.

Enabled	Label	Filter Expression
☑	tcp11-13	(tcp) && ((ip.src==192.168.20.11 || ip.src==192.168.20.13) && (ip.dst==192.168.20.11 || ip.dst==192.168.20.13))
☑	tcp11-12	(tcp) && ((ip.src==192.168.20.12 || ip.src==192.168.20.11) && (ip.dst==192.168.20.12 || ip.dst==192.168.20.11))
☑	tcp13-14	(tcp) && ((ip.src==192.168.20.13 || ip.src==192.168.20.14) && (ip.dst==192.168.20.13 || ip.dst==192.168.20.14))
☑	udp11-13	(udp) && ((ip.src==192.168.20.11 || ip.src==192.168.20.13) && (ip.dst==192.168.20.11 || ip.dst==192.168.20.13))
☑	udp13-12	(udp) && ((ip.src==192.168.20.12 || ip.src==192.168.20.13) && (ip.dst==192.168.20.12 || ip.dst==192.168.20.13))
☑	udp11-12	(udp) && ((ip.src==192.168.20.12 || ip.src==192.168.20.11) && (ip.dst==192.168.20.12 || ip.dst==192.168.20.11))
☑	udp13-14	(udp) && ((ip.src==192.168.20.13 || ip.src==192.168.20.14) && (ip.dst==192.168.20.13 || ip.dst==192.168.20.14))
☑	icmp11-13	(icmp) && ((ip.src==192.168.20.11 || ip.src==192.168.20.13) && (ip.dst==192.168.20.11 || ip.dst==192.168.20.13))
☑	icmp13-12	(icmp) && ((ip.src==192.168.20.12 || ip.src==192.168.20.13) && (ip.dst==192.168.20.12 || ip.dst==192.168.20.13))
☑	icmp11-12	(icmp) && ((ip.src==192.168.20.12 || ip.src==192.168.20.11) && (ip.dst==192.168.20.12 || ip.dst==192.168.20.11))
☑	icmp13-14	(icmp) && ((ip.src==192.168.20.13 || ip.src==192.168.20.14) && (ip.dst==192.168.20.13 || ip.dst==192.168.20.14))

Figure 26. Wireshark Filters

3.5.4 Standard Configuration.

The standard configuraton experiment consists of two objectives. The first is to build a baseline network as outlined in Section 3.4 and generate network traffic for evaluation. The second is to determine the extent of visibility for inter-VM traffic to external sources of both ESXi and XenServer hosts. Inter-VM traffic is contained within the vSwitch when the destination is on the same host as the source. Scripting network traffic using a stream of ICMP messages and TCP or UDP segments ensures that the source and destination reflects traffic as internal or external. Configuring SPAN on the physical switch enables port mirroring to view all data being passed to the physical network. Port mirroring copies each packet and forwards it to a network monitoring port that has a management system connected. Figure 27 shows the topology for the standard vSwitch configuration on two production hosts and a management host for packet capture. Each production host is responsible for managing their respective production and management vSwitches. The management host captures port mirroring traffic and determines if inter-VM traffic is visible to external devices. The management host is not considered part of the domain infrastructure.

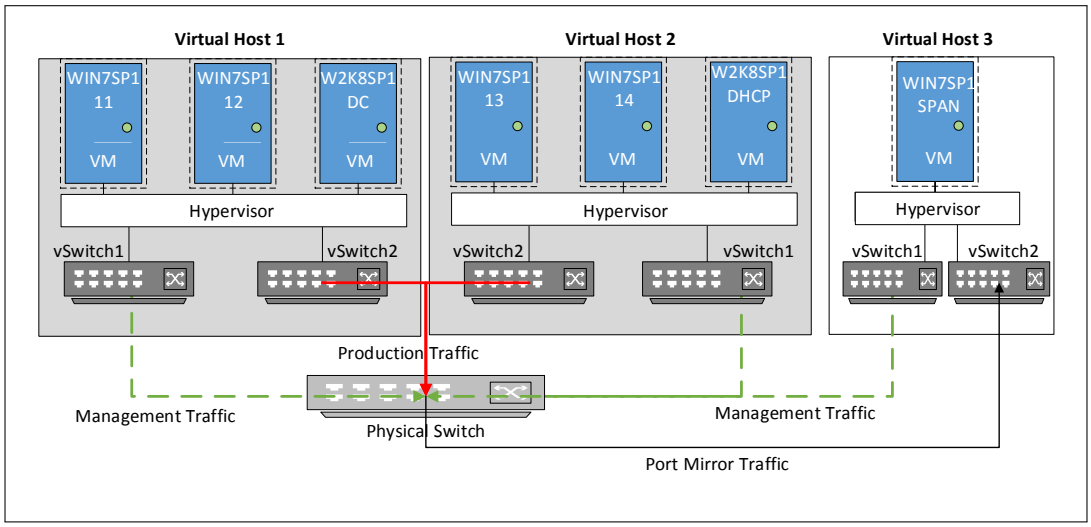


Figure 27. Standard vSwitch Infrastucture

3.5.5 Distributed Configuration.

The goal for the distributed configuration experiment is to determine the amount of visibility of inter-VM traffic in a distributed environment. Section 3.4.6 gives details on the differences of standard and distributed vSwitches. The distributed configuration moves vSwitch management from each individual host to a centralized host for managing vSwitches on multiple hosts. Figure 28 shows the topology of three virtual host in a distributed configuration. The distributed vSwitch manages the production traffic for all hosts and is co-located with ESXi vCenter or XenServer DVSC management server for configuration. All virtual host are considered part of the domain infrastructure.

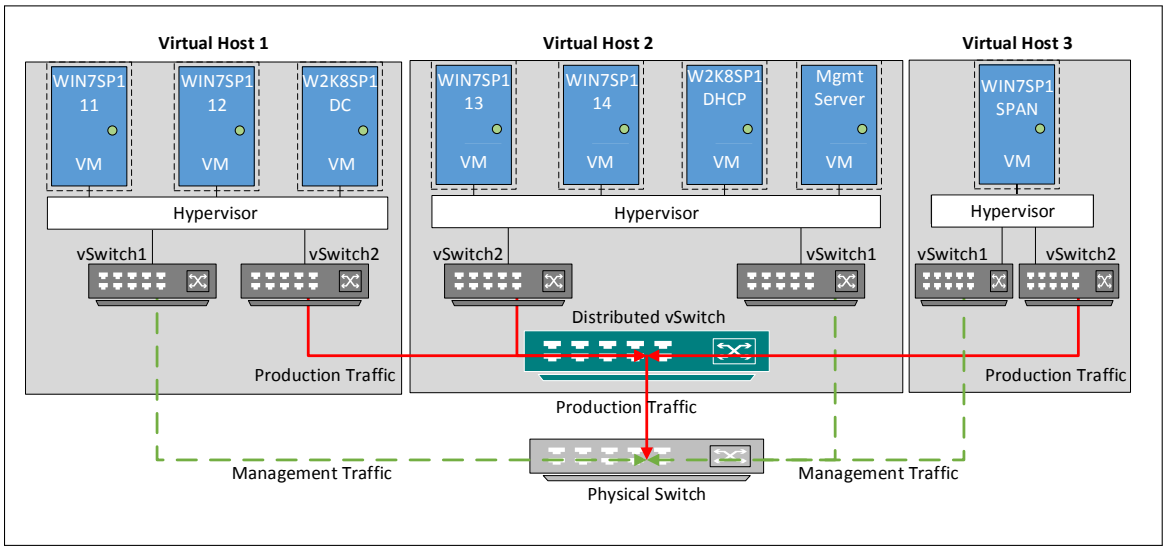


Figure 28. Distributed vSwitch Infrastructure

Distributed vSwitches also open many different avenues for capturing the inter-VM traffic, such as virtual port mirroring. As in the standard configuration, port mirroring is also used in a distributed environment but configured through the virtual management tool instead of the physical network. An example of port mirroring in a distributed virtual network is using VMWare ESXi vDS and vSphere web client. The port mirroring options available through the vSphere web client are listed below.

1. **Distributed Port Mirroring** - Mirror packets from a number of distributed ports to other distributed ports on the same host. If the source and the destination are on different hosts, this session type does not function.

2. **Remote Mirroring Source** - Mirror packets from a number of distributed ports to specific uplink ports on the corresponding host.

3. **Remote Mirroring Destination** - Mirror packets from a number of VLANs to distributed ports.

4. **Encapsulated Remote Mirroring (Layer3) Source** - Mirror packets from a number of distributed ports to remote agents' IP addresses. The virtual machines traffic is mirrored to a remote physical destination through an IP tunnel.

5. **Distributed Port Mirroring (legacy)** - Mirror packets from a number of distributed ports to a number of distributed ports and/or uplink ports on the corresponding host.

3.6 Summary

The problem with virtualization is the network administrator's visibility of inter-VM traffic on the physical network. This chapter discusses the approach taken to gain a basic understanding of the inter-VM traffic visibility and performance. The experiment factors covers both physical hardware configuration and virtual management specifications required for the research. Furthmore, experiment details provide information on the traffic generation, packet capture, and network diagrams used for overall assessment.

IV. Results and Analysis

This chapter outlines the results and analysis of visibility and performance for inter-VM traffic. Using the scripts discussed in Section 3.5.1, network traffic is generated and collected for both visibility and duration performance. A collection of 10,000 connections are captured for each TCP and UDP protocol. Each TCP connection consists of nine packets with a total size of 548 bytes, and each UDP connection consists of two packets with total size of 115 bytes. Figure 29 displays a single TCP and UDP connection viewed within Wireshark. Section 4.1 provides results for inter-VM traffic visibility and Section 4.2 outlines the performance results for VM traffic and vendor.

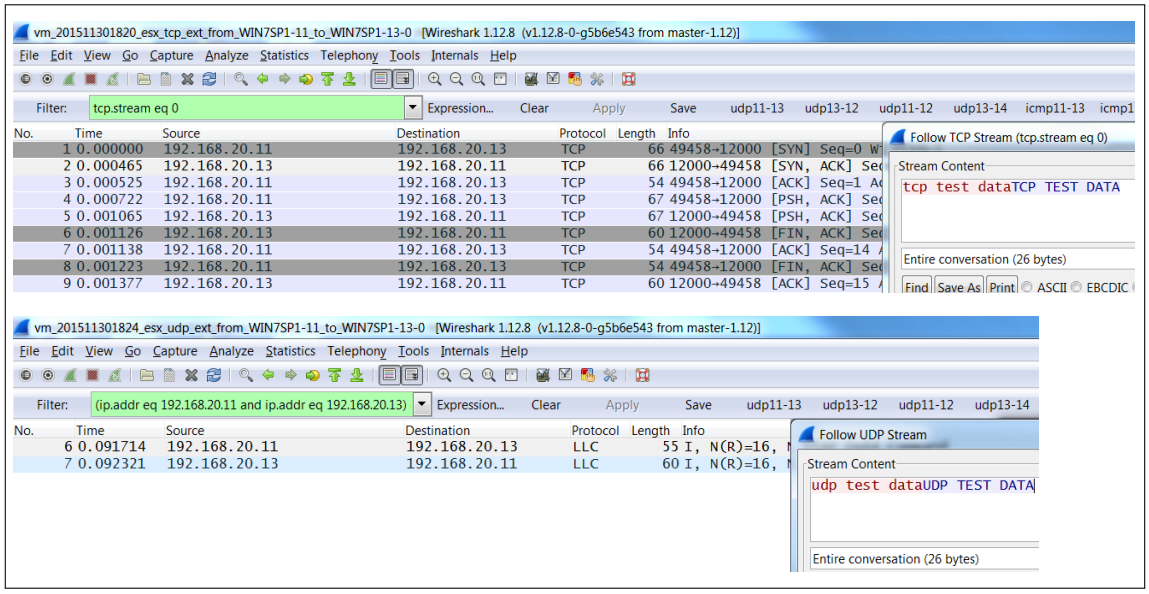


Figure 29. TCP and UDP Conversations

4.1 Visibility Results and Analysis

Complete visibility of network traffic enhances security by having the ability to detect possible attacks within the whole network. Inter-VM visibility evaluation for

ESXi and XenServer standard networks are covered in Sections 4.1.1 and 4.1.2 while distributed networks are covered in Sections 4.1.3 and 4.1.4.

4.1.1 VMWare ESXi Standard Network.

Three evaluations for visibility are accomplished using an ESXi standard network. The evaluations of the vSS include initial install, *pktcap-uw* program, and promiscuous mode. Connectivity between VMs is verified using the *ping* application.

The first evaluation is to determine the real-time visibility of inter-VM traffic within a host using the initial installation of ESXi. A review of the Wireshark packet capture provided the outcome of no inter-VM traffic being exposed to the external network. Only traffic with an external destination is visible and provides monitoring capability to an administrator or IDS.

Due to the lack of visibility on initial installation, *pktcap-uw* is used for a second evaluation of inter-VM traffic visibility. A limitation to the *pktcap-uw* tool is the packet capture (pcap) file location. The pcap file is maintained on the virtual host and must be downloaded for analysis. WinSCP [36] is a file transfer program that provides a secure method for transferring the pcap files. Packet captures using *pktcap-uw* cannot be analyzed in real-time and requires additional administrative tasks to accomplish the analysis. Results also reveal no inter-VM traffic being exposed to the external network but a complete capture of network traffic is obtained within the pcap files for analysis. By itself, *pktcap-uw* does not alleviate any vulnerability and only visibility of external traffic is able to be monitored in real-time.

The final vSS evaluation conducted uses promiscuous mode. By default, a VM can only receive traffic directly sent to it. Setting a portgroup or virtual switch to promiscuous mode causes the VMs to detect all traffic passed on the virtual switch. Figure 30 shows the location for setting a portgroup to promiscuous mode.

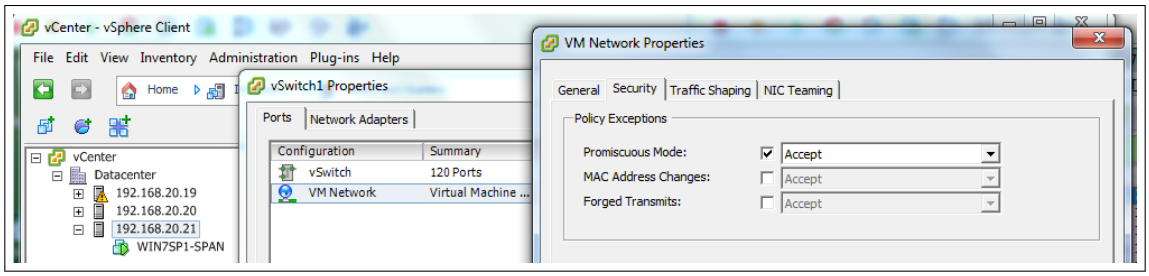


Figure 30. Promiscuous Mode Setting

A virtual switch using promiscuous mode broadcasts traffic to all VMs but does not pass inter-VM traffic to the external network. Promiscuous mode can benefit an enterprise IDS by allowing the administrator to install a separate VM to perform IDS functions on each local host. The IDS VM can collect data in real-time and communicate with the enterprise IDS for security.

The overall results reveal that no inter-VM traffic is exposed to the external network using a vSS. Table 9 shows the results based upon the methods and protocols evaluated. The ability to monitor or manage real-time inter-VM traffic on a vSS network requires additional administrative tasks or the deployment of other resources, such as local IDS.

Table 9. ESXi Standard Inter-VM Visibility Results

Evaluation	ICMP	TCP	UDP
Initial	No	No	No
PKTCAP-UW	No	No	No
Promiscuous	No	No	No

4.1.2 Citrix XenServer Standard Network.

Three evaluations for visibility are accomplished using a XenServer OVS network. The evaluations include initial install, tcdump utility, and port mirroring. Connectivity between VMs is verified using the *ping* application.

The first evaluation is to determine the visibility of inter-VM traffic within a host using the initial installation of XenServer. A review of the Wireshark packet capture provided the same outcome as ESXi standard network. The results reveal that no internal traffic is being exposed to the external network and is open to vulnerabilities.

The next option for inter-VM traffic is using the *tcpdump* program. Tcpdump requires the use of SSH and WinSCP to transfer the pcap file to a system for analysis. SSH is enabled by default and is a vulnerability if not managed properly. The pcap file delivered a complete view of inter-VM traffic but real-time analysis cannot be performed due to traffic not visible to the physical network.

With the proper amount of physical interfaces, the use of port mirroring to forward traffic on a standard configuration is possible but not within the scope of this research. The physical interface must be designated for *port mirroring only* since it disables the interface for full operations.

The overall results reveal that no inter-VM traffic is exposed to the external network using a standard OVS. Table 10 shows that all OVS options evaluated cannot pass inter-VM traffic to the physical network. The ability to monitor or manage the inter-VM traffic on the current standard OVS network requires additional administrative tasks or deployment of other resources. Similar to ESXi standard, the options available for capturing inter-VM traffic require a packet capture program on local host and transfer to external source or installing a local IDS VM on each virtual host.

Table 10. Xen Standard Inter-VM Visibility Results

Evaluation	**ICMP**	**TCP**	**UDP**
Initial	No	No	No
Tcpdump	No	No	No
Port Mirroring	No	No	No

4.1.3 VMWare ESXi Distributed Network.

The initial evaluation of the vDS begins with the same criteria as the vSS. Evaluation of an initial installation, *pktcap-uw*, and promiscuous mode provide a comparison. The vDS evaluation results demonstrate a similar outcome as vSS and did not show any difference between the two types of virtual switches. No real-time inter-VM traffic was visible to the external network using the evaluation methods.

Additional vDS testing for inter-VM traffic includes port mirroring. Setting up port mirroring requires some infrastructure changes. The first item is to remove the SPAN configuration from the physical switch. The next change is to add the packet capture VM (WIN7SP1-SPAN) to the domain. **Encapsulated Remote Mirroring (Layer3) Source** option is selected within the web client for port mirroring. Figure 31 shows the configuration used for this evaluation.

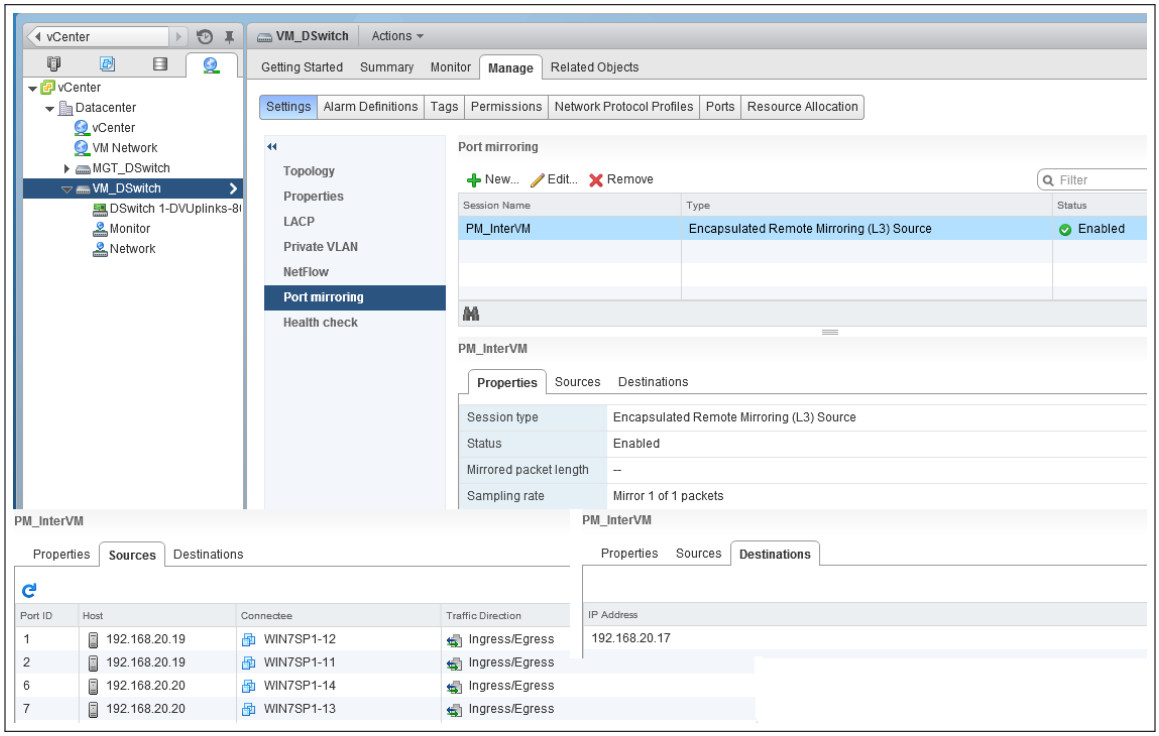


Figure 31. ESXi Port Mirroring Configuration

The evaluation of port mirroring provides the desired results. Port mirroring exposes real-time inter-VM traffic to the external network and a complete view of network traffic for packet capture and security.

Table 11 lists the outcome for the ESXi distributed switch visibility of inter-VM traffic. The basic functionality of vDS is similar to vSS but the advanced options provide greater functionality for administration and security.

Table 11. ESXi Distributed Inter-VM Visibility Results

Evaluation	ICMP	TCP	UDP
Initial	No	No	No
PKTCAP-UW	No	No	No
Promiscuous	No	No	No
Port Mirroring	Yes	Yes	Yes

4.1.4 Citrix XenServer Distributed Network.

A complete traffic visibility evaluation was not performed on a Citrix XenServer distributed network using the DVSC. Initial install and *tcpdump* evaluations using SPAN on the physical switch resulted in external visibility only. Advanced options, such as port mirroring, are unavailable due to the inability to obtain a license for the version of XenServer 6.5. The only version available for download is DVSC version 6.2 which the functionality is being deprecated and no longer available in future releases [37]. Also, the DVSC version 6.2 only supports RSPAN for port analysis while the Cisco Nexus physical switch only supports SPAN and Encapsulated Remote SPAN.

Table 12. Xen Distributed Inter-VM Visibility Results

Evaluation	ICMP	TCP	UDP
Initial	No	No	No
Tcpdump	No	No	No

4.2 Performance Results and Analysis

Latency and the comparative cost within different virtualization implementations is key to the overall performance of the network. Latency in terms of this evaluation is defined as the duration time for each transport-layer conversation. Since TCP is a connection-oriented protocol, duration time for each session starts with a three-way handshake and finishes with a shutdown acknowledgment from the destination VM. Duration time for UDP entails only a data packet sent by the source VM and a response by the destination VM. Studies show both TCP and UDP communications may see degradation in a cloud-based or small virtualization environment due to the virtual host performing both computing and networking responsibilities [38]. The latency cost of implementing a virtual environment relies on the vendor, type of virtual switch, and type of traffic used (e.g., internal or external).

4.2.1 Performance Data.

The Minitab Statistical Analysis Tool [39] is used to analyze data captured during evaluation. Minitab offers the Anderson-Darling Normality statistic [40] which measures how well a set of data is distributed. A smaller p-value defines how well the data is distributed. The experiment results reflect a p-value of less than 0.005 and interprets the data as well distributed with a 95% confidence interval. Appendix F provides detailed Minitab summary reports used for calculating overall traffic comparison. Tables 13 and 14 offer a data summary to reflect the 95% confidence interval, minimum, maximum, and mean duration time for each virtual switch, including internal and external traffic.

Table 13. TCP Traffic Summary in Microseconds

Switch	Traffic	Min	Mean	Max	95% CI
ESX Standard	Internal	0.7	0.855	1.3	0.8539 - 0.8562
ESX Standard	External	0.8	1.034	1.4	1.0327 - 1.0352
ESX Distributed	Internal	0.7	0.913	1.8	0.9111 - 0.9140
ESX Distributed	External	1.0	1.253	1.8	1.2515 - 1.2546
XEN Standard	Internal	0.8	1.020	1.1	1.0187 - 1.0214
XEN Standard	External	1.0	1.185	1.5	1.1837 - 1.1872
XEN Distributed	Internal	1.1	1.441	2.0	1.4390 - 1.4433
XEN Distributed	External	1.2	2.029	2.8	2.0256 - 2.0316

Table 14. UDP Traffic Summary in Microseconds

Switch	Traffic	Min	Mean	Max	95% CI
ESX Standard	Internal	0.3	0.393	0.9	0.3922 - 0.3934
ESX Standard	External	0.3	0.491	0.9	0.4902 - 0.4922
ESX Distributed	Internal	0.3	0.404	1.0	0.4031 - 0.4044
ESX Distributed	External	0.4	0.618	1.0	0.6168 - 0.6192
XEN Standard	Internal	0.3	0.481	0.8	0.4800 - 0.4820
XEN Standard	External	0.4	0.563	1.0	0.5614 - 0.5639
XEN Distributed	Internal	0.6	0.839	1.3	0.8375 - 0.8412
XEN Distributed	External	0.8	1.351	1.9	1.3489 - 1.3538

4.2.2 Virtual Switch Comparison.

With TCP being widely used in today's network environment, it is important to know the effect of virtualization on different types of network traffic. The results are determined by calculating the average duration time of traffic between VMs.

Figure 32 shows the internal and external TCP traffic duration means on each virtual switch for comparison. As shown in the visibility test, distributed switches offer more functionality in administration and security but come with a latency cost. The cost of using a distributed switch for internal TCP traffic increased by 7% on ESXi and 34% on XenServer. The cost increased 19% and 53% respectively for external TCP traffic. The TCP traffic latency cost is relatively substantial and should be consider before deploying a distributed switch infrastructure.

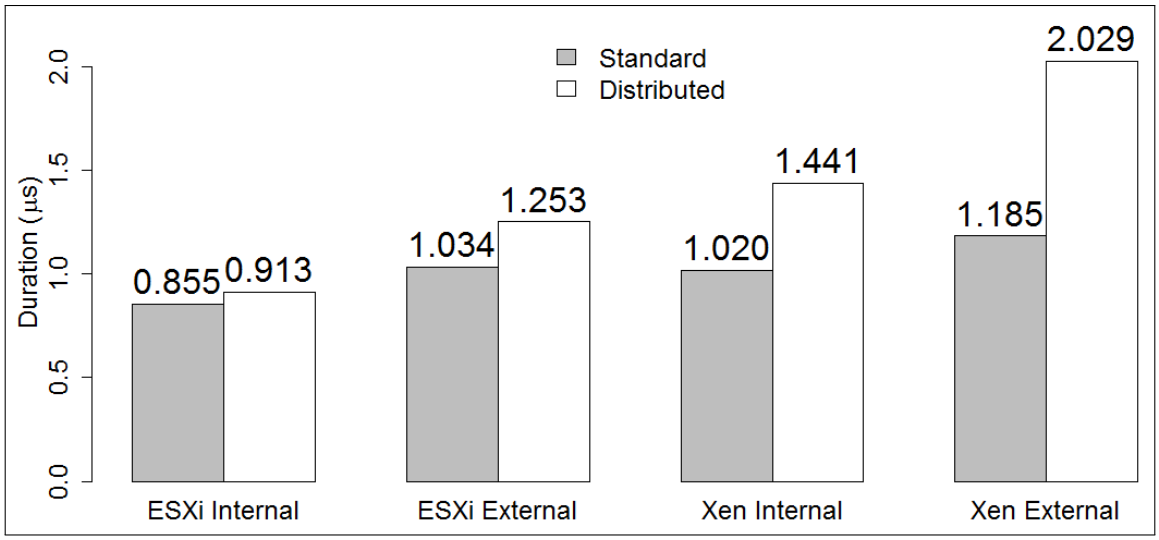


Figure 32. Virtual Switch TCP Traffic Means

Figure 33 shows the UDP traffic duration means for internal and external traffic on each virtual switch. Similar to TCP, the data reflects an average performance for sending UDP traffic. The cost of using a distributed switch for internal UDP traffic increased by 3% on ESXi and 54% on XenServer. The cost increased 23% and 82% respectively for external UDP traffic.

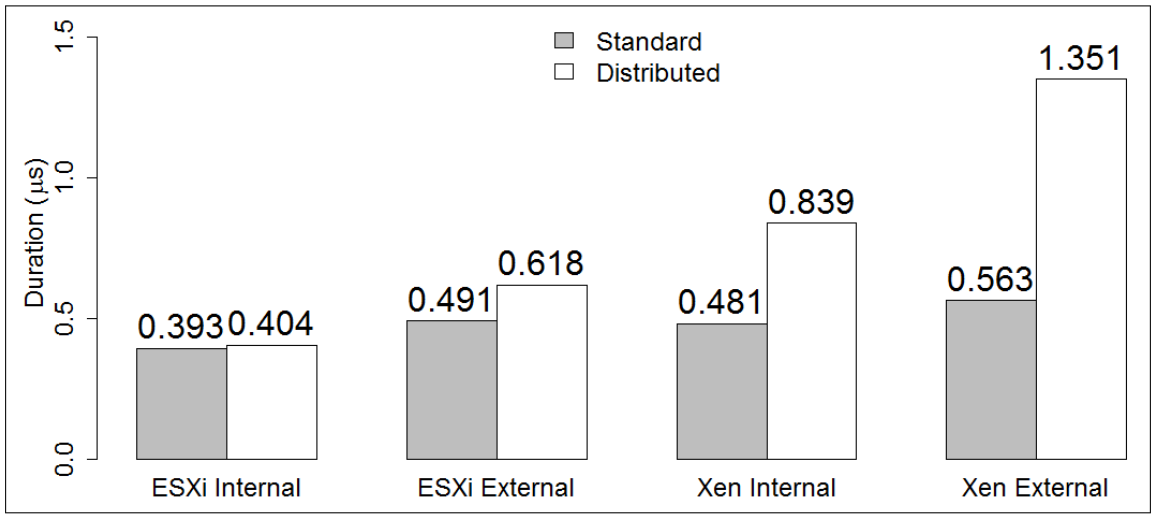


Figure 33. Virtual Switch UDP Traffic Means

4.2.3 Vendor Comparison.

Comparison of vendors requires knowledge of virtualization technology and specifications of different providers. Further details are found in Section 2.5 that describes the different technologies considered and Table 6 details the hypervisor specifications between ESXi and XenServer.

Vendor comparison provides a performance evaluation for passing inter-VM traffic within a virtual environment. Figure 34 provides results in a side-by-side view of switch type and traffic by vendor. Based upon the results, ESXi communicated TCP traffic 18% faster on vSS than Xen OVS and 45% on vDS over Xen DVSC. UDP traffic displayed similar results, as shown in Figure 35, the ESXi vSS is 20% faster than Xen OVS and ESXi vDS is 70% faster than Xen DVSC. The overall results shows that a VMWare ESXi platform has a lower duration time than Citrix XenServer platform when processing the generated VM traffic.

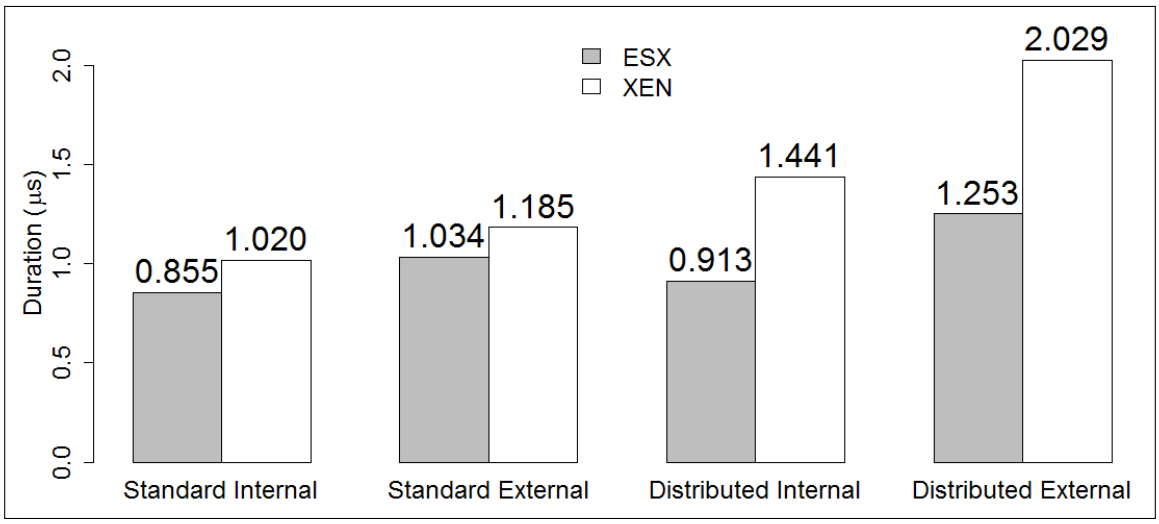


Figure 34. Vendor TCP Traffic Means

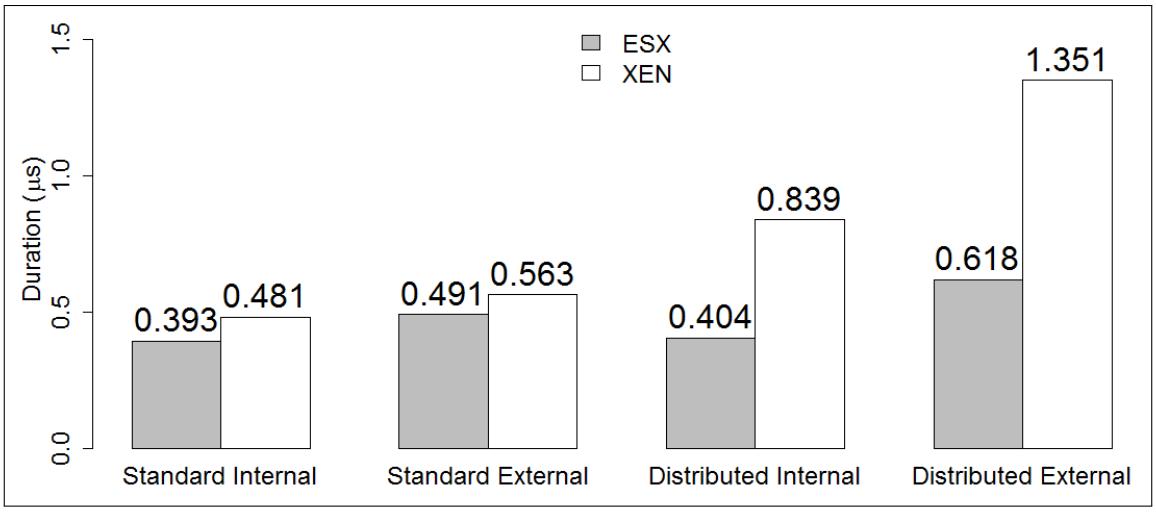


Figure 35. Vendor UDP Traffic Means

4.3 Summary

This chapter presents and analyzes the data collected for inter-VM traffic visibility and performance. The results for visibility show initial configuration for neither standard or distributed virtual switches offer visibility of inter-VM traffic. It is determined that standard switches require additional resources and extensive administration. The results also show inter-VM traffic is more accessible to distributed switches when using the advanced functionality (e.g., port mirroring and NetFlow).

Additionally, traffic visibility comes with a latency cost when implementing a distributed switch over a standard switch. The cost of TCP and UDP traffic increased for both ESXi and XenServer when using a distributed switch. The results also indicate that VMWare ESXi vSS and vDS outperforms Citrix XenServer OVS and DVSC when transmitting both TCP and UDP traffic.

V. Conclusions and Recommendations

This chapter summarizes the research performed in this study. Section 5.1 presents the conclusions reached during experimentation. Section 5.2 discusses the impact and contributions of this research. Section 5.3 presents potential future work.

5.1 Research Conclusion

The main research goal of determining inter-VM traffic visibility is successful for three out four network switches. Visibility testing on the XenServer DVSC was minimal due to physical switch constraints and inability to obtain DVSC license for current version. Inter-VM traffic is not visible on initial installation of a virtual switch and requires additional configuration depending on the type of switch. Methods used for obtaining traffic visibility are based upon the vendor and type of switch implemented. A summary of results for a standard switch can be found in Section 5.1.1 and distributed switch in Section 5.1.2.

The secondary goal of performance provided a latency cost for using a distributed switch over standard switch and a comparison of VMWare and Citrix virtual platforms. The results provide additional information that can be used to determine if the advanced options of a distributed switch outweigh the latency cost. The vendor comparison revealed that a proprietary solution using full virtualization outperformed an open source solution using paravirtualization. The summarization of performance results are covered in Section 5.1.3.

5.1.1 Standard Switch.

A standard virtual switch offers no visibility of inter-VM traffic to the external network. The use of a packet capture program, such as *pktcap-uw* for ESXi and

tcpdump for XenServer, are required to gain inter-VM traffic visibility. Use of a packet capture program for administration also requires a method to move the pcap file from a virtual host to a different location where the file is analyzed. The other option is to install an additional VM on the same virtual host that performs as an IDS or other management type system. To obtain complete visibility with this option, promiscuous mode must be enabled on the virtual switch. Port mirroring is available on the XenServer OVS but not tested.

5.1.2 Distributed Switch.

The distributed switch has the same ability to gain inter-VM traffic visibility as a standard switch but offers more advanced functionality. Distributed switches offer advanced functionality such as port mirroring and Netflow. A successful port mirroring session was accomplish using ESXi vDS but not XenServer DVSC. The version of DVSC required could not be obtained for the research.

5.1.3 Performance.

The overall performance shows a standard switch is faster than distributed switch when processing networking traffic. The latency cost difference for internal TCP traffic was 7% on ESXi and 34% on XenServer and external was 19% and 53% respectively. Internal UDP traffic was more significant showing a difference of only 3% on ESXi but 54% on XenServer. The cost also increased 23% and 82% respectively for external UDP traffic.

Overall vendor results show that VMWare ESXi is faster than Citrix XenServer. The difference in latency cost for both TCP and UDP traffic on the XenServer OVS and DVSC was higher than an ESXi vSS and vDS.

5.2 Research Contributions

This research contributes to the DoD focus as it moves into the fast growing technology of virtualization. Additionally, the research brings up critical considerations that must be taken when introducing virtualization into a networking environment.

A key consideration is how the level of network traffic visibility correlates with the ability to secure the network. Multiple options for gaining traffic visibility are identified in this research and provides an administrator with a additional information for monitoring and securing the network.

Performance is an additional consideration for virtualization. The performance difference between a standard and distributed virtual switch is a determining factor on which to implement. Research shows that the standard switch is faster but requires more administration for security.

5.3 Recommendation for Future Work

There are many paths that can be taken as future work. Future work could include integrating the research methods with a host-based security system or further testing the advanced features of virtual networking. Virtualization also offers the means to easily transfer resources from one host to another, such as vMotion within VMWare ESXi. Capturing the traffic from resource movements and analyzing for possible threats or performance degradation. Another path is to analyze the visibility of traffic from a host with VMs using SAN or NAS. A more in depth study of additional options within an OVS standard with multiple NIC may offer further capabilities in a smaller network. Obtain the DVSC version that is compatible with XenServer 6.5 for enterprise-level comparison. Software Defined Networking (SDN) is another avenue that can be incorporated using the research conducted. SDNs include resources that control network traffic that could have an effect on visibility of inter-VM traffic.

Appendix A. Physical Switch Configuration

1. VLAN

   - configure terminal
   - vlan 20
   - name VM
   - no shutdown
   - exit
   - interface ethernet 1/25 (repeat for each required interface)
   - switchport access vlan 20
   - show vlan name 20

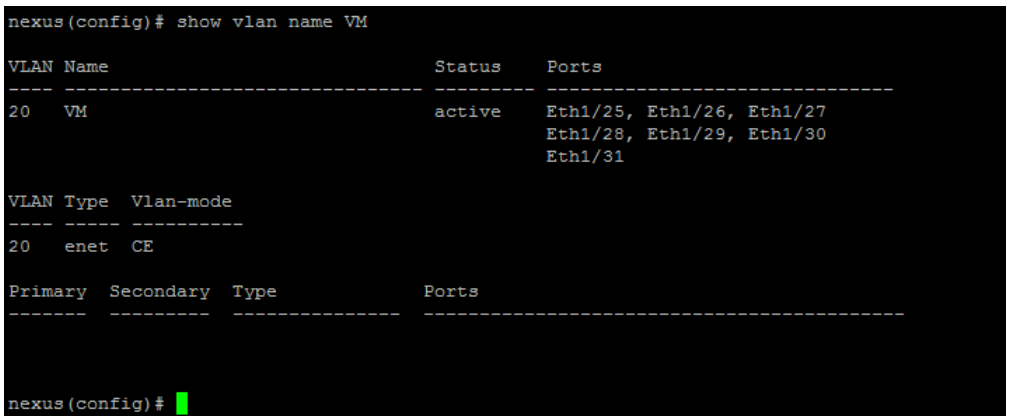


Figure 36. VLAN

2. VLAN Gateway

   - configure terminal
   - interface vlan 20
   - ip address 192.168.20.1 255.255.255.0
   - show interfaces vlan 20

```
nexus(config)# show interface vlan 20
Vlan20 is up, line protocol is up, autostate enabled
  Hardware is EtherSVI, address is  58f3.9c4b.8b7c
  Internet Address is 192.168.20.1/24
  MTU 1500 bytes, BW 1000000 Kbit, DLY 10 usec
  Last clearing of "show interface" counters never

nexus(config)#
```

Figure 37. Gateway

3. Trunk Port

    - configure terminal
    - interface ethernet 1/25 (repeat for each required interface)
    - switchport trunk allowed vlan 20
    - switchport mode trunk
    - spanning-tree port type edge

4. Access Port

    - configure terminal
    - interface ethernet 1/31
    - switchport mode access
    - switchport access vlan 20

```
nexus# show interface brief

--------------------------------------------------------------------------------
Ethernet      VLAN   Type Mode   Status  Reason                   Speed     Port
Interface                                                                  Ch #
--------------------------------------------------------------------------------
Eth1/25       1       eth  trunk  up      none                    1000(D) --
Eth1/26       1       eth  trunk  up      none                    1000(D) --
Eth1/27       1       eth  trunk  up      none                    1000(D) --
Eth1/28       1       eth  trunk  up      none                    1000(D) --
Eth1/29       1       eth  trunk  up      none                    1000(D) --
Eth1/30       1       eth  trunk  up      none                    1000(D) --
Eth1/31       20      eth  access up      none                    1000(D) --
```

Figure 38. Switch Interfaces

5. Switch Port Analyzer

- configure terminal
- monitor session 1
- exit
- interface ethernet 1/27 (destination interface)
- switchport monitor
- exit
- monitor session 1
- destination interface ethernet 1/27
- source interface ethernet 1/25 both
- source interface ethernet 1/26 both
- source vlan 20
- exit
- no monitor session 1 shut
- show monitor session 1

```
nexus# sh monitor session 1
   session 1
---------------
type              : local
state             : up
acl-name          : acl-name not specified
source intf       :
    rx            : Eth1/25       Eth1/26
    tx            : Eth1/25       Eth1/26
    both          : Eth1/25       Eth1/26
source VLANs      :
filter VLANs      : filter not specified
    rx            : 20
destination ports : Eth1/27

Legend: f = forwarding enabled, l = learning enabled

nexus#
```

Figure 39. Switch Port Analyzer Monitor

Appendix B. Management Test Batch Code

1. C:\PS_Scripts\InterVMTest.bat

```
@echo off
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen icmp int 11 12 250"
timeout 298
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen tcp int 11 12 250"
timeout 298
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen udp int 11 12 250"
timeout 298
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen icmp ext 11 13 250"
timeout 298
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen tcp ext 11 13 250"
timeout 298
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen udp ext 11 13 250"
timeout 298
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen icmp int 13 14 250"
timeout 298
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen tcp int 13 14 250"
timeout 298
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen udp int 13 14 250"
timeout 298
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen icmp ext 13 12 250"
timeout 298
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen tcp ext 13 12 250"
timeout 298
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen udp ext 13 12 250"
timeout 298
echo.
```

```
echo Packet Capture Completed
echo.
timeout 10
```

2. C:\PS_Scripts\PingAll.bat

```
@echo off
echo System Ping Test has started.
echo.
ping 192.168.20.1
timeout 3
echo.
ping 192.168.20.2
timeout 3
echo.
ping 192.168.20.3
timeout 3
echo.
ping 192.168.20.11
timeout 3
echo.
ping 192.168.20.12
timeout 3
echo.
ping 192.168.20.13
timeout 3
echo.
ping 192.168.20.14
timeout 3
echo.
echo Ping Test is complete
echo.
timeout 3
```

3. C:\PS_Scripts\SystemReset-Part1.bat

```
@echo off
echo Checking for Windump processes.
echo.
powershell.exe -command "& C:\PS_Scripts\5StopWinDump.ps1"
echo.
echo System reset Part 1 has started.
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen icmp int 11 12 5"
timeout 90
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen icmp int 12 11 5"
timeout 90
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen icmp int 13 14 5"
timeout 90
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen icmp int 14 13 5"
timeout 90
echo.
echo System reset Part 1 is complete when all boxes are closed.
echo.
echo Run Part 2
echo.
pause
```

4. C:\PS_Scripts\SystemReset-Part2.bat

```
@echo off
echo System reset Part 2 has started.
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen icmp int 11 12 5"
timeout 60
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen icmp int 12 11 5"
timeout 60
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen icmp int 13 14 5"
timeout 60
echo.
```

```
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen icmp int 14 13 5"
timeout 60
echo.
echo System reset Part 2 is complete when all boxes are closed.
echo.
echo Run Part 3
echo.
pause
```

5. C:\PS_Scripts\SystemReset-Part3.bat

```
@echo off
echo System reset Part 3 has started.
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen icmp int 11 12 3"
timeout 10
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen tcp int 11 12 3"
timeout 10
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen udp int 11 12 3"
timeout 10
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen icmp ext 11 13 3"
timeout 10
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen tcp ext 11 13 3"
timeout 10
echo.
powershell.exe -command "& C:\PS_Scripts\1StartTest.ps1 xen udp ext 11 13 3"
timeout 10
echo.
echo System Reset is complete
echo.
timeout 10
```

Appendix C. Management Test Powershell Code

1. C:\PS_Scripts\1StartTest.ps1

```
$vend = $args[0]
$proto = $args[1]
$test = $args[2]
$src = $args[3]
$dst = $args[4]
$pkt = $args[5]
$tsrc = "WIN7SP1-$src"
$tdst = "WIN7SP1-$dst"

$startwindump = {
        start-process powershell -argument "C:\PS_Scripts
            ↪ \2StartWinDump.ps1 $vend $proto $test $tsrc
            ↪  $tdst"
        &$startserver
}
$startserver = {
        if ($proto -eq 'tcp' -or $proto -eq 'udp') {
                start-process powershell -argument "C:\
                    ↪ PS_Scripts\3StartServer.ps1 $proto
                    ↪ $tdst"
                &$starttest
        }
        else {
                &$starttest
```

```
        }
}
$starttest = {
        start-process powershell -argument "C:\PS_Scripts
            ↪ \4ProtocolTesting.ps1 $proto $tsrc $tdst
            ↪ $pkt $test"
}
Write-Host
Write-Host "Vendor:" $vend " Protocol:" $proto " Test
    ↪ Type:" $test
Write-Host "Source:" $tsrc " Destination:" $tdst
Write-Host
&$startwindump
```

2. C:\PS_Scripts\2StartWinDump.ps1

```
$vend = $args[0]
$proto = $args[1]
$test = $args[2]
$src = $args[3]
$dst = $args[4]
$tdate = Get-Date -format yyyyMMddHHmm
Invoke-Command  ComputerName $src -ScriptBlock { C:\
    ↪ Intervm\Batch\StartWinDump.bat $using:vend $using:
    ↪ proto $using:test $using:src $using:dst $using:
    ↪ tdate }
```

3. C:\PS_Scripts\3StartServer.ps1

```
$proto = $args[0]
$dst = $args[1]
Invoke-Command ComputerName $dst -ScriptBlock { C:\
    ↪ Intervm\Batch\StartTCPUDPServer.bat $using:proto }
```

4. C:\PS_Scripts\4ProtocolTesting.ps1

```
$proto = $args[0]
$src = $args[1]
$dst = $args[2]
$pkt = $args[3]
$test = $args[4]
$prototesting = {
        switch ($proto) {
                icmp { &$testicmp; break }
                tcp  { $stpr = 5; &$testudptcp ;break }
                udp  { $stpr = 4; &$testudptcp ;break }
                }
        }
$testicmp = {
Invoke-Command ComputerName $src -ScriptBlock { C:\
    ↪ Intervm\Batch\TestICMP.bat $using:dst $using:pkt }
&$finish
}
$testudptcp = {
```

```
        Invoke-Command ComputerName $src -ScriptBlock
            ↪ { C:\Intervm\Batch\TestTCPUDP.bat $using:
            ↪ dst $using:proto $using:pkt }
        Invoke-Command ComputerName $dst -ScriptBlock
            ↪ { C:\Intervm\Batch\StopServer.bat $using:
            ↪ stpr }
        &$finish
}
$finish = {
        Invoke-Command ComputerName $src -ScriptBlock
            ↪ { C:\Intervm\Batch\StopWinDump.bat }
        New-PSDrive -Name X -Root \\$src\c$\Intervm\
            ↪ Capture -PSProvider FileSystem
        cd X:
        Move-Item -Path ./vm* -Destination C:\Capture\
        cd C:\PS_Scripts
}
&$prototesting
```

5. C:\PS_Scripts\5StopWinDump.ps1

```
$computers = Get-ADComputer -Filter 'Name -like "WIN7SP1
    ↪ -1*"' | Foreach-Object { $_.Name }
Invoke-Command ComputerName $computers -ScriptBlock {
    ↪ C:\Intervm\Batch\StopWinDump.bat }
```

6. C:\PS_Scripts\6StopServer.ps1

```
$computers = Get-ADComputer -Filter 'Name -like "WIN7SP1
    ↪ -1*"' | Foreach-Object { $_.Name }
Invoke-Command ComputerName $computers -ScriptBlock {
    ↪ C:\Intervm\Batch\StopServer.bat }
```

Appendix D. Virtual Machine Batch Source Code

1. C:\Intervm\Batch\StartTCPUDPServer.bat

```
@echo off
echo %1 Server started on %computername%
python c:\Intervm\%1Server.py
```

2. C:\Intervm\Batch\StartWinDump.bat

```
@echo off
echo %computername% WinDump Collection started on %1 for %2 protocol and %3
    ↪ traffic
Start C:\Intervm\WinDump.exe -i 1 -q -w C:\Intervm\Capture\vm_%6_%1_%2_%3_from_
    ↪ %4_to_%5- -n -C 30 -W 10 -U -s 0
```

3. C:\Intervm\Batch\StopServer.bat

```
@echo off
for /f "tokens=%1" %%a in ('netstat -aon ^| find "0.0.0.0:12000" ') do taskkill
    ↪  /F /PID %%a
```

4. C:\Intervm\Batch\StopWinDump.bat

```
@echo off
taskkill /f /im "WinDump.exe"
```

5. C:\Intervm\Batch\TestICMP.bat

```
@echo off
echo ICMP Testing Started
echo From %computername% to %1
ping -n %2 -l 1460 %1
```

6. C:\Intervm\Batch\TestTCPUDP.bat

```
@echo off
echo From %computername% to %1
python C:\Intervm\ProtocolTest.py %1 12000 %2 %3
```

Appendix E. Virtual Machine Python Source Code

1. C:\Intervm\ProtocolTest.py [41]

```
# Modified code found at http://johnallen.us/?p=236
# Author: John Allen
# Date: Feb 10, 2010
# Last: Dec 19, 2015

import sys
import socket
import time

host = sys.argv[1]
port = int(sys.argv[2])
proto = sys.argv[3]
test = int(sys.argv[4])

while 1 :
  if proto == "udp":
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
  else:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  s.settimeout(5)
  try:
    if proto == "udp":
      s.sendto("udp test data", (host, port))
      modifiedMessage, serverAddress = s.recvfrom(2048)
      s.shutdown(2)
      print "Successful UDP connection: host " + host + " port " + str(port)
      test = test - 1
    else:
      s.connect((host, port))
      sentence = "tcp test data"
      s.send(sentence)
      modifiedSentence = s.recv(1024)
      s.shutdown(2)
      print "Successful TCP connection: host " + host + " port " + str(port)
      test = test - 1
  except Exception, e:
    try:
```

```
      errno, errtxt = e
    except ValueError:
      print "Successful UDP connection: host " + host + " port " + str(port)
    else:
      if errno == 107:
        print "Success connecting to " + host + " on UDP port: " + str(port)
      else:
        print "Cannot connect to " + host + " on port: " + str(port)
        print e
  if test == 0 :
    sys.exit(0)
  s.close
  time.sleep(1)
```

2. C:\Intervm\TCPServer.py [42]

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
while 1:
     connectionSocket, addr = serverSocket.accept()
     sentence = connectionSocket.recv(1024)
     capitalizedSentence = sentence.upper()
     connectionSocket.send(capitalizedSentence)
     connectionSocket.close()
```

3. C:\Intervm\UDPServer.py [42]

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
while 1:
     message, clientAddress = serverSocket.recvfrom(2048)
     modifiedMessage = message.upper()
     serverSocket.sendto(modifiedMessage, clientAddress)
```

Appendix F. Minitab - Data Summary Reports

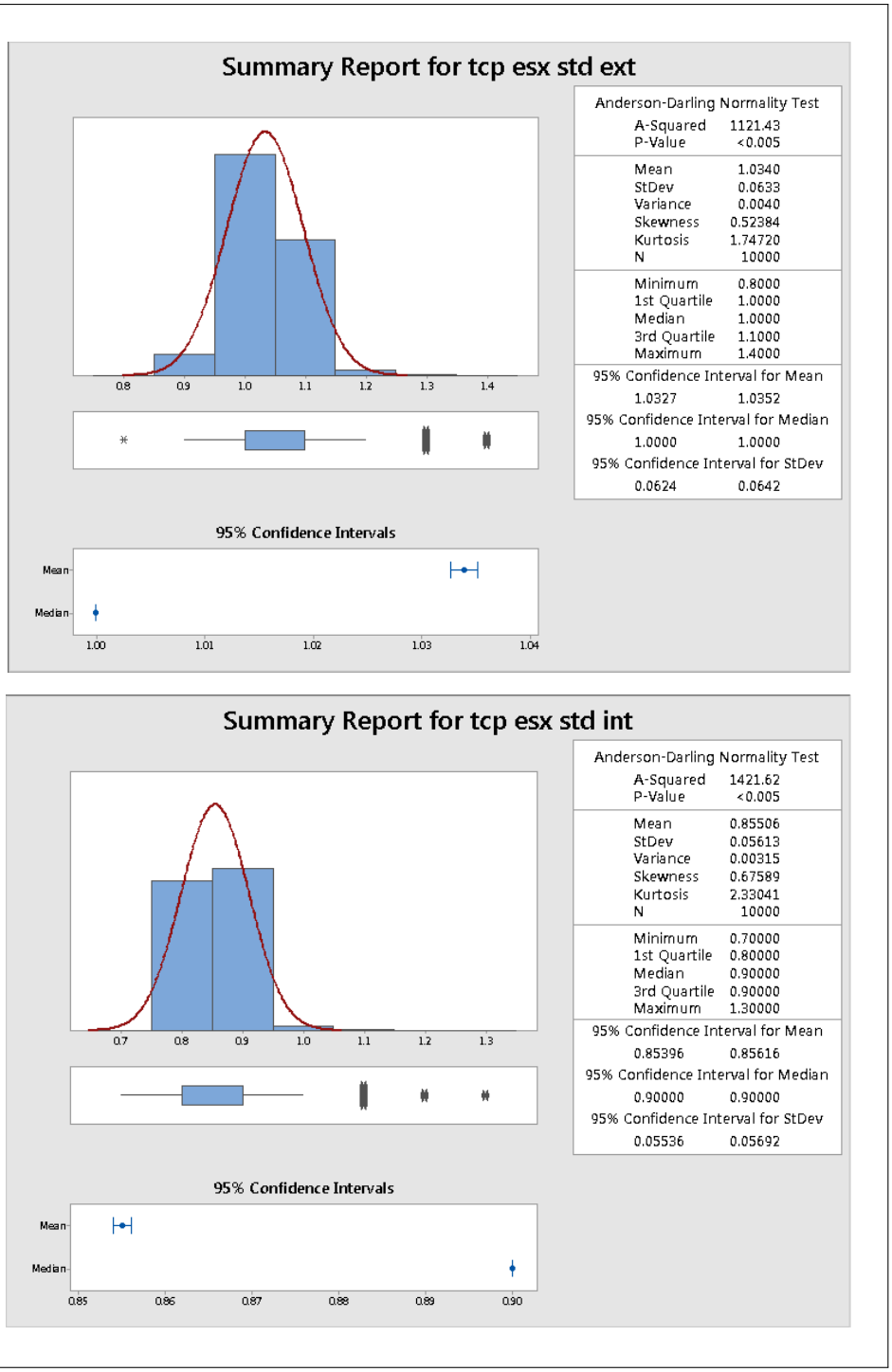


Figure 40. ESXi Standard TCP Summary Report

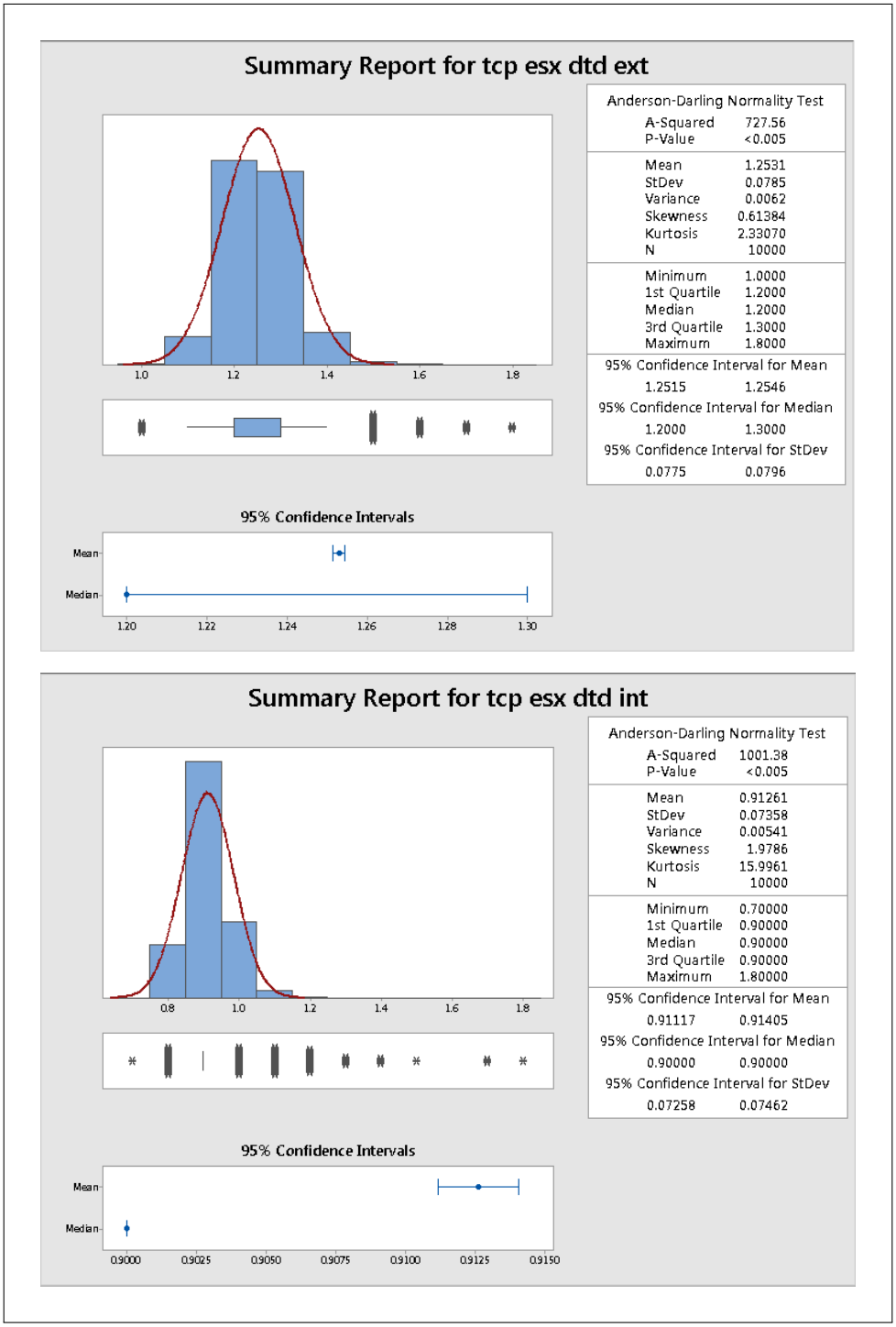


Figure 41. ESXi Distributed TCP Summary Report

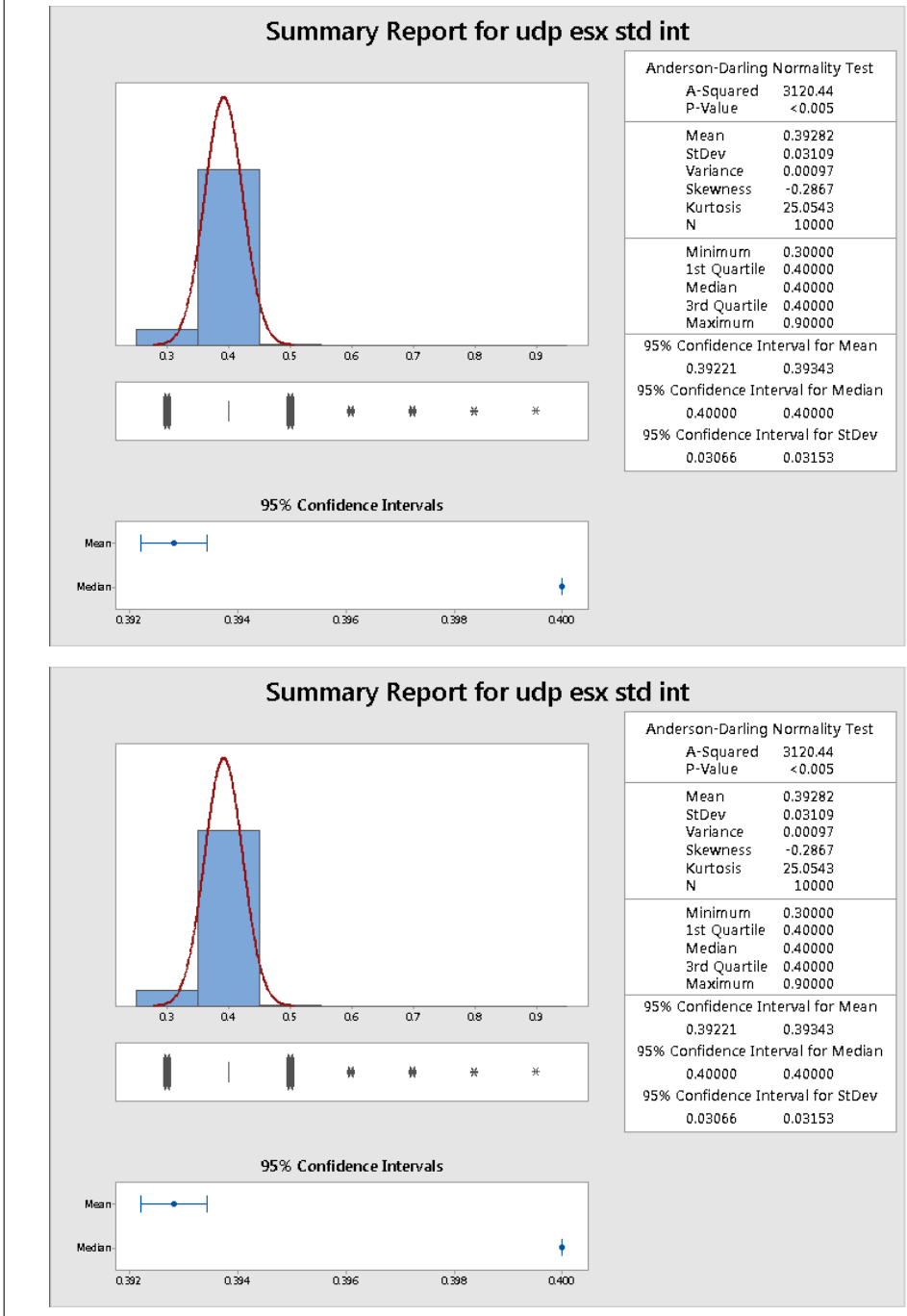


Figure 42. ESXi Standard UDP Summary Report

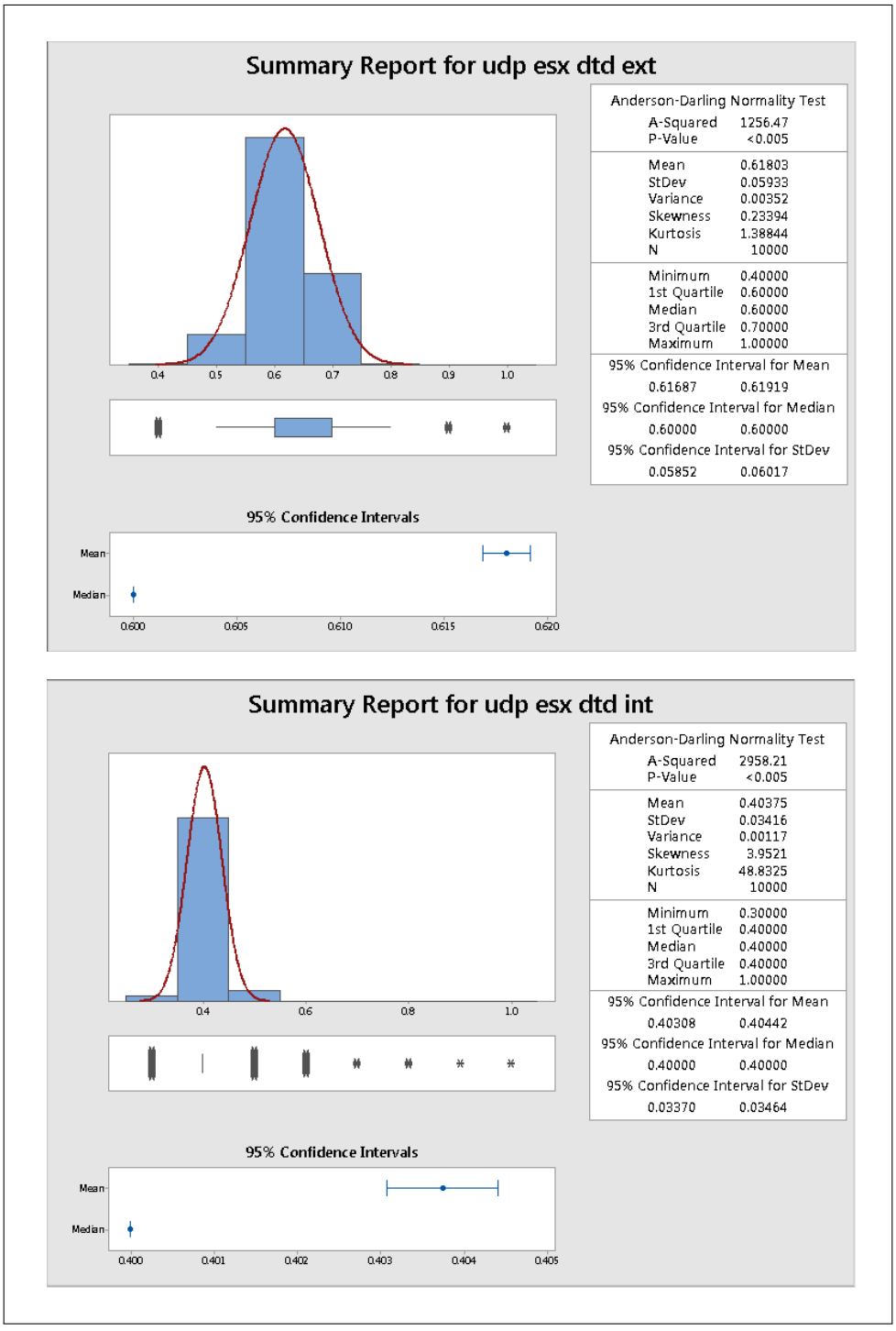


Figure 43. ESXi Distributed UDP Summary Report

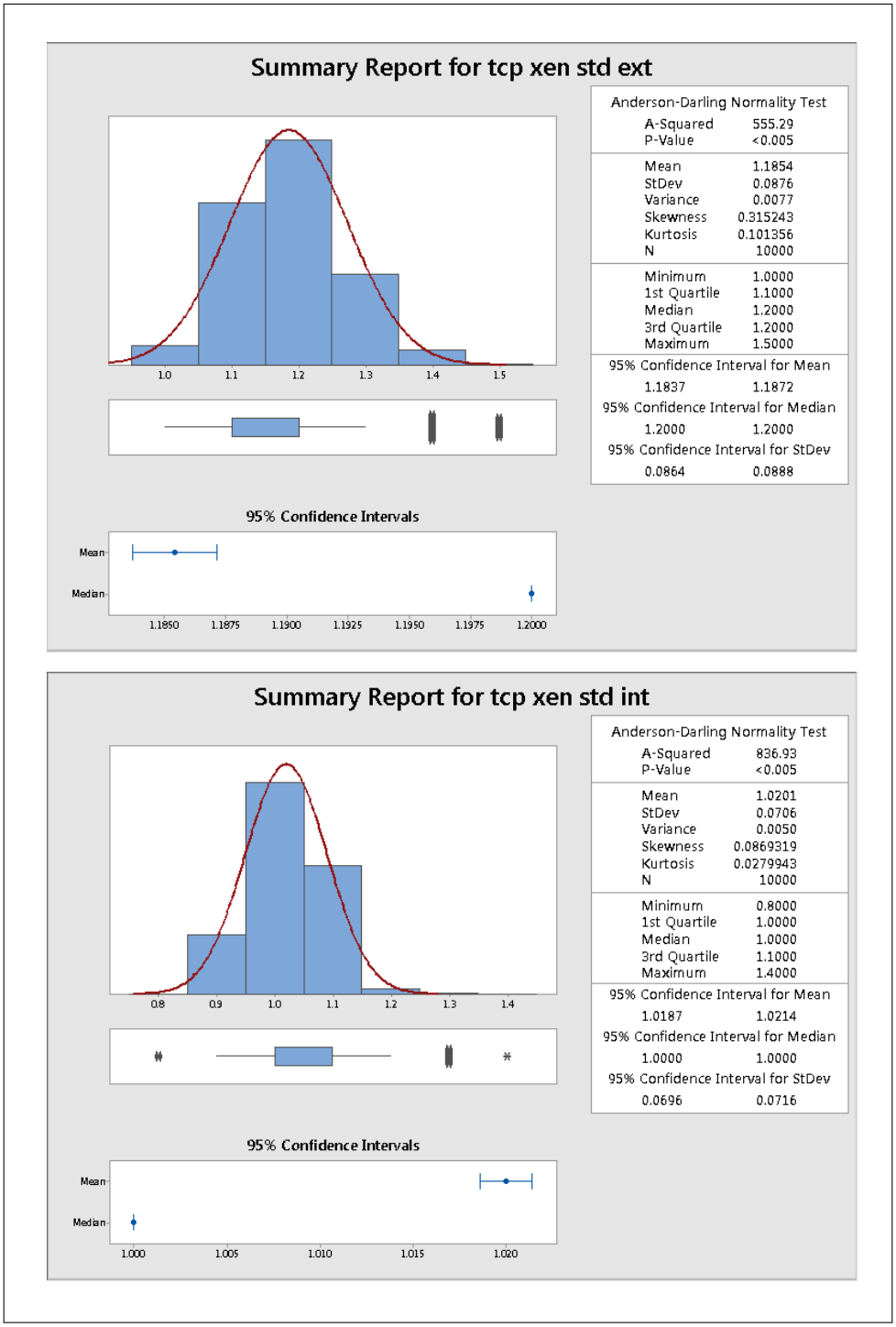
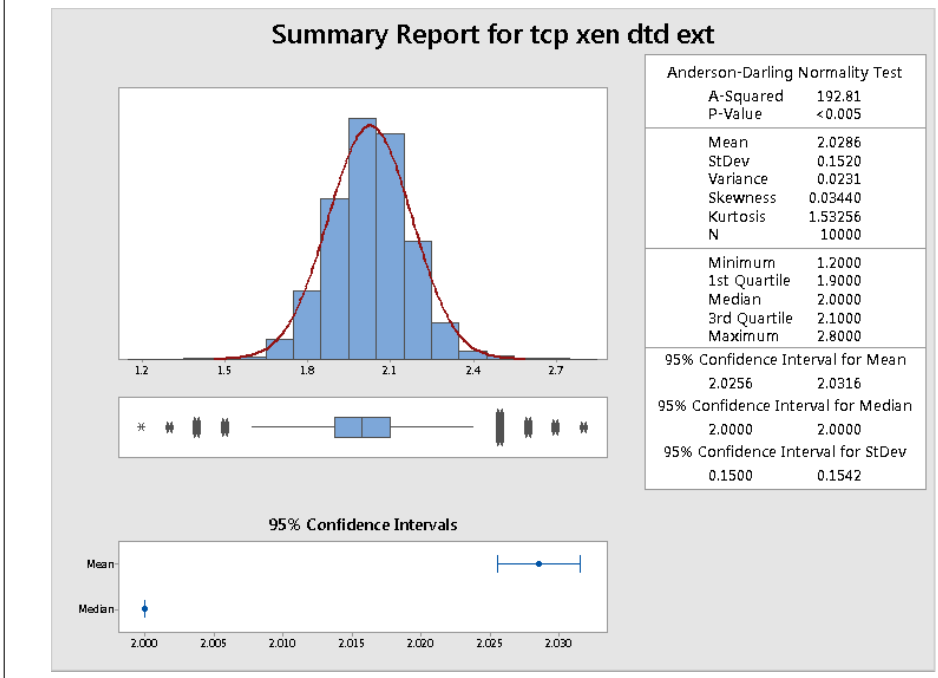


Figure 44. Xen Standard TCP Summary Report



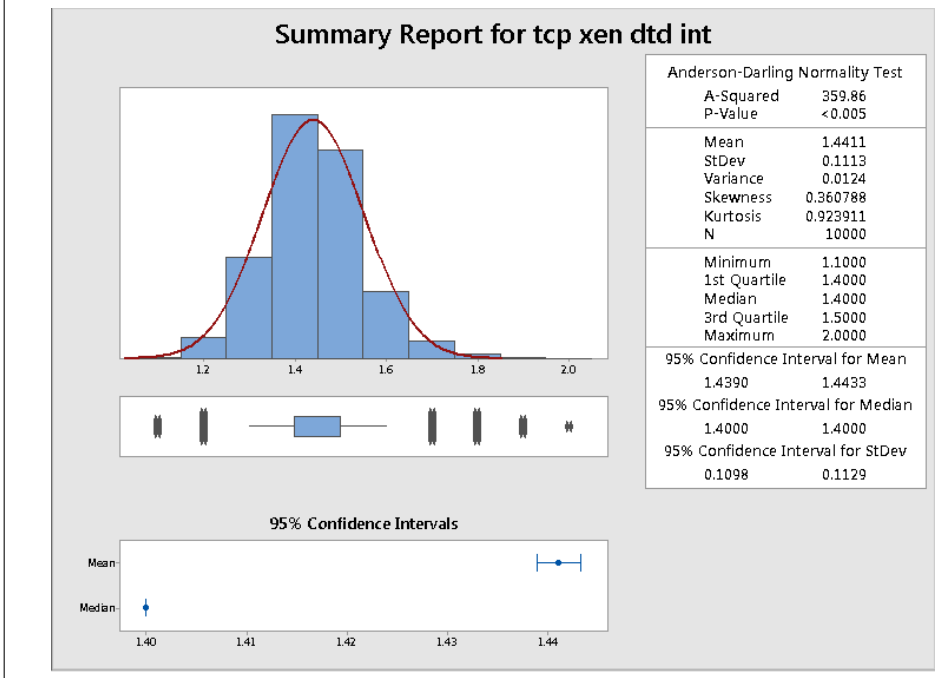


Figure 45. Xen Distributed TCP Summary Report

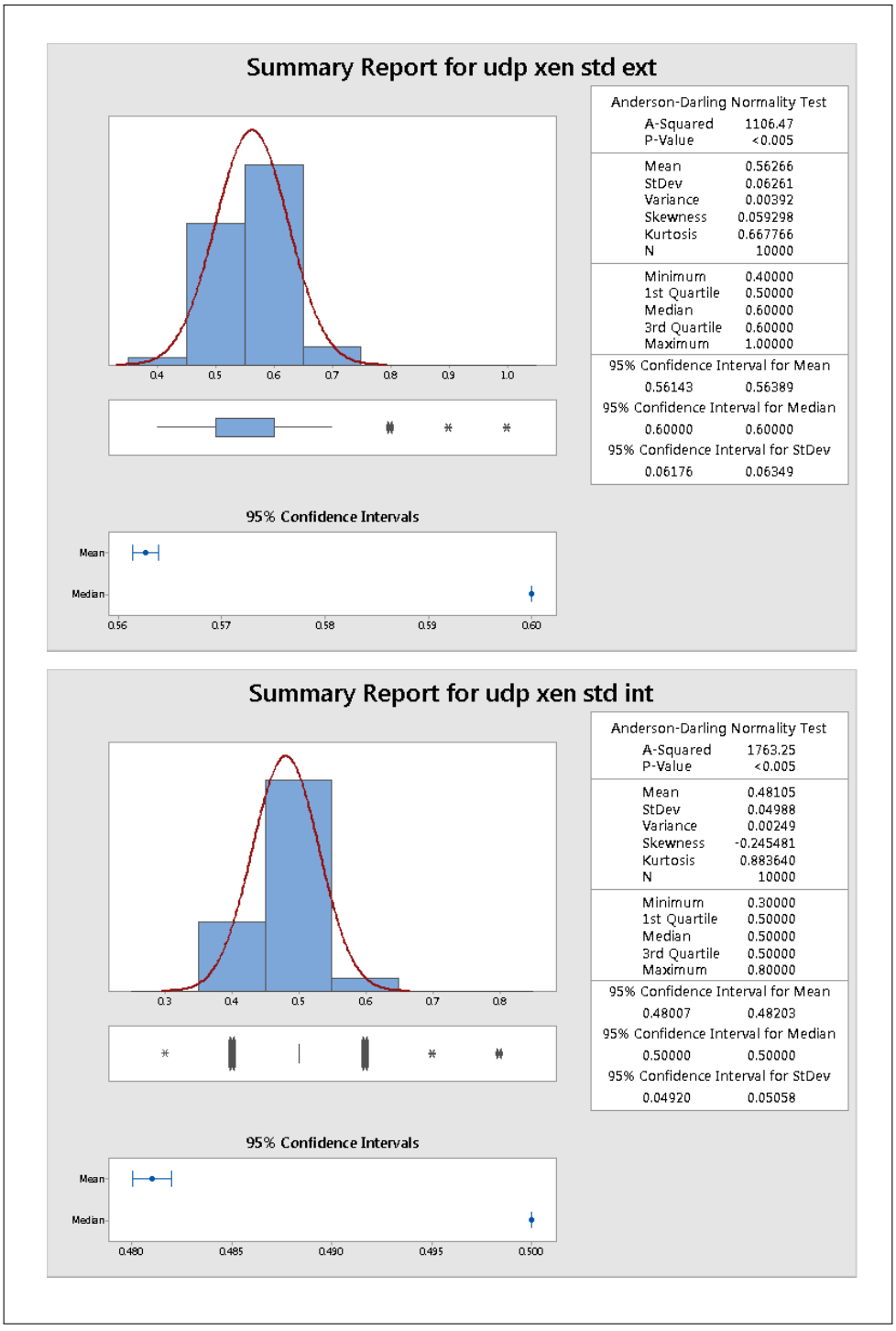


Figure 46. Xen Standard UDP Summary Report

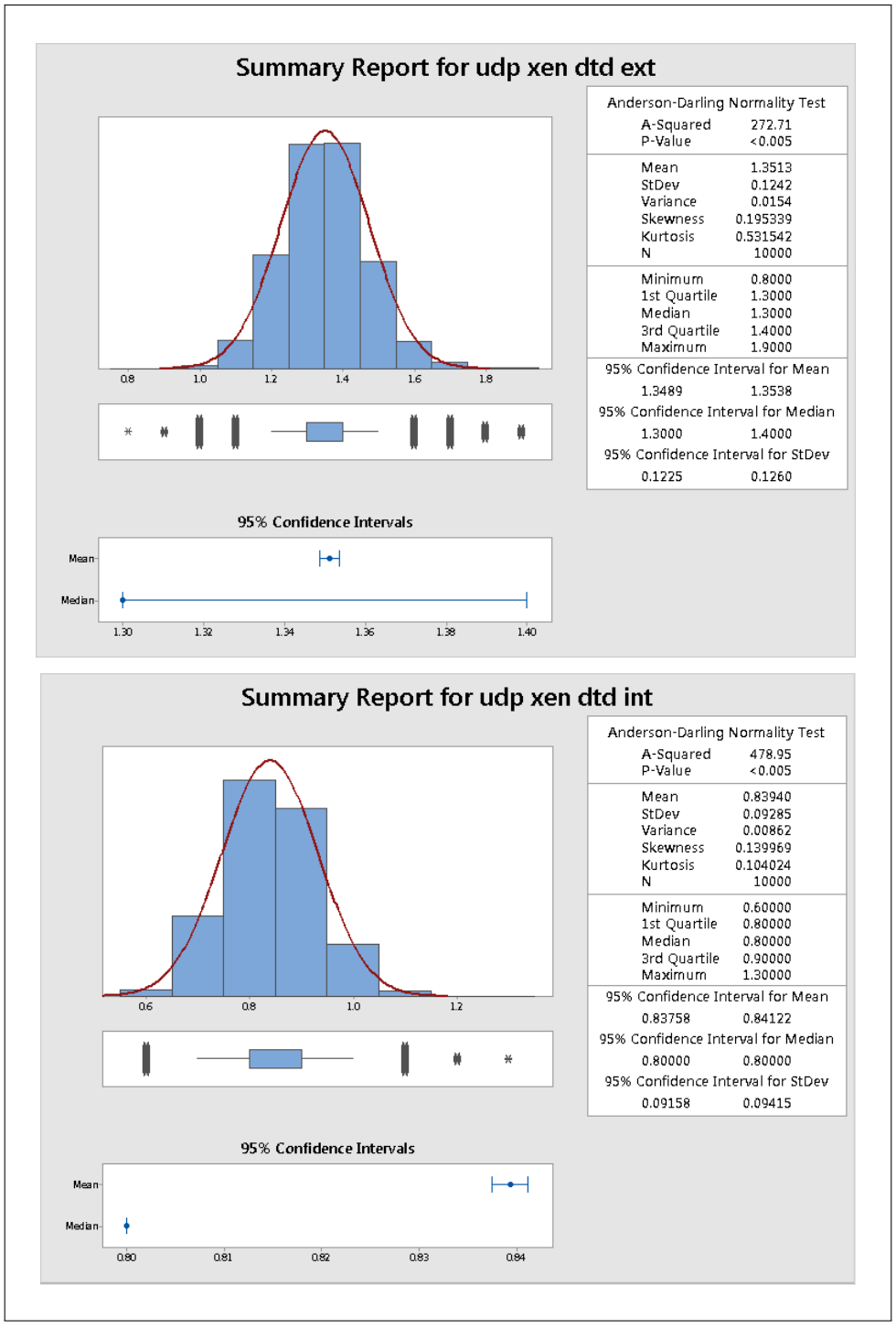


Figure 47. Xen Distributed UDP Summary Report

Bibliography

1. J. Brodkin. With long history of virtualization behind it, IBM looks to the future. Last Accessed: Jan 25, 2016, 2009. *http://www.networkworld.com/article/2254433/virtualization/with-long-history-of-virtualization-behind-it–ibm-looks-to-the-future.html*.

2. M.B. Walker. DISA zeroing in on virtualization. Last Accessed: Jan 25, 2016, 2015. *http://www.fiercegovernmentit.com/story/disa-zeroing-virtualization/2015-11-02*.

3. VMWare Inc. Understanding Full Virtualization, Paravirtualization, and Hardware Assist. Last Accessed: Jan 25, 2016, 2007. *https://www.vmware.com/files/pdf/VMware_paravirtualization.pdf*.

4. M. Portnoy. *Virtualization Essentials*. Sybex, May 2012.

5. R. Pavlicek. Securing Your Cloud With the Xen Hypervisor. Last Accessed: Jan 25, 2016, 2013. *http://www.slideshare.net/buildacloud/securing-your-cloud-with-xen-by-russell-pavlicek*.

6. Distributed Management Task Force Inc. Open Virtualization Format White Paper. Last Accessed: Jan 25, 2016, 2014. *http://www.dmtf.org/sites/default/files/standards/documents/DSP2017_2.0.0.pdf*.

7. Distributed Management Task Force Inc. Virtual Networking Management White Paper. Last Accessed: Jan 25, 2016, 2012. *http://dmtf.org/sites/default/files/standards/documents/DSP2025_1.0.0.pdf*.

8. VMWare Inc. VMWare vSphere Basics. Last Accessed: Jan 25, 2016, 2011. *https://pubs.vmware.com/vsphere-50/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-50-basics-guide.pdf*.

9. D. Kusnetzky. *Virtualization: A Manager's Guide*. O'Reilly Media, June 2011.

10. K. Chandrasekaran. *Essentials of Cloud Computing*. CRC Press, 2015.

11. M. Fawzi. Virtualization and Protection Rings Welcome to Ring -1 Part i. Last Accessed: Jan 25, 2016, 2009. *https://fawzi.wordpress.com/2009/05/24/virtualization-and-protection-rings-welcome-to-ring-1-part-i/*.

12. X. Li, H. Sun, and Q. Wen. An approach for secure-communication between XEN virtual machines. In *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*, volume 01, pages 283–286, Oct 2012.

13. XenProject. Domain. Last Accessed: Jan 25, 2016, 2011. *http://wiki.xen.org/wiki/Domain*.

14. Citrix Inc. XenServer Tech Info. Last Accessed: Jan 25, 2016, 2015. *https://www.citrix.com/products/xenserver/tech-info.html*.

15. R. Kumar. An Importance of Using Virtualization Technology in Cloud Computing. 1(2), Feb 2015.

16. D. Marinescu. *Cloud Computing*. Elsevier Inc., May 2013.

17. K. Benzidane, S. Khoudali, F. Leila, and A. Sekkaki. Toward inter-VM Visibility in a Cloud Environment using Packet Inspection. In *Telecommunications (ICT), 2013 20th International Conference on*, pages 1–5, May 2013.

18. J. Rutkowska. Introducing Blue Pill. Last Accessed: Jan 25, 2016, 2006. *http://theinvisiblethings.blogspot.com/2006/06/introducing-blue-pill.html*.

19. M. Rouse. Virtual Machine Escape. Last Accessed: Jan 25, 2016, 2009. *http://whatis.techtarget.com/definition/virtual-machine-escape*.

20. D. Goodin. Extremely serious virtual machine bug threatens cloud providers everywhere. Last Accessed: Jan 25, 2016, 2015. *http://arstechnica.com/security/2015/05/extremely-serious-virtual-machine-bug-threatens-cloud-providers-everywhere/*.

21. R. Naraine. US-CERT warns of guest-to-host VM escape vulnerability. Last Accessed: Jan 25, 2016, 2012. *http://www.zdnet.com/article/us-cert-warns-of-guest-to-host-vm-escape-vulnerability/*.

22. Y. Shoaib and O. Das. Pouring Cloud Virtualization Security Inside Out. Last Accessed: Jan 25, 2016, 2014. *http://arxiv.org/abs/1411.3771*.

23. B. Guan, J. Wu, Y. Wang, and S.U. Khan. CIVSched: A Communication-Aware Inter-VM Scheduling Technique for Decreased Network Latency between Co-Located VMs. *Cloud Computing, IEEE Transactions on*, 2(3):320–332, July 2014.

24. S. Tatham. PuTTY SSH and Telnet Client. Last Accessed: Jan 25, 2016, 2015. *http://www.putty.org/*.

25. R. Macek. WhatMatrix - Virtualization Comparison. Last Accessed: Jan 25, 2016, 2015. *https://www.whatmatrix.com/comparison/Virtualization*.

26. DISA Field Security Operations. DoD Secure Host Baseline Repository. Last Accessed: Jan 25, 2016, 2015. *https://disa.deps.mil/ext/cop/iase/dod-images/Pages/index.aspx*.

27. Open vSwitch. Production Quality, Multilayer Open Virtual Switch. Last Accessed: Feb 2, 2016, 2014. *http://openvswitch.org/*.

28. Citrix Inc. Citrix XenServer 6.2.0 vSwitch Controller User Guide. Last Accessed: Jan 25, 2016, 2013. *http://docs.citrix.com/content/dam/en-us/xenserver/xenserver-62/dvs_controller.pdf*.

29. Riverbed Technology. WinDump, tcpdump for Windows using WinPcap. Last Accessed: Jan 25, 2016, 2006. *https://www.winpcap.org/windump/*.

30. Wireshark Foundation. Wireshark Network Protocol Analyzer. Last Accessed: Jan 25, 2016, 2015. *https://www.wireshark.org/*.

31. VMWare Inc. Using the pktcap-uw tool in ESXi 5.5 and later (2051814). Last Accessed: Jan 25, 2016, 2015. *http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&extern*

32. L. MartinGarcia. TCPDUMP and LIBPCAP. Last Accessed: Jan 25, 2016, 2015. *http://www.tcpdump.org/*.

33. Citrix Inc. How to Capture a Network Trace from a XenServer Physical Interface, Virtual Bridge, and VM Virtual Interface. Last Accessed: Jan 25, 2016, 2014. *http://support.citrix.com/article/CTX120869*.

34. Linux Foundation. What is XAPI? Last Accessed: Jan 25, 2016, 2013. *http://www.xenproject.org/developers/teams/xapi.html*.

35. Open vSwitch. Open vSwitch Manual ovs-vsctl(8). Last Accessed: Feb 2, 2016, 2016. *http://openvswitch.org/support/dist-docs/ovs-vsctl.8.txt*.

36. M. Prikryl. WinSCP SFTP, SCP and FTP client for Windows. Last Accessed: Jan 25, 2016, 2014. *https://winscp.net/eng/docs/introduction*.

37. Citrix Inc. Deprecated Feature: Distributed Virtual Switch Controller for XenServer 6.2.0. Last Accessed: Jan 25, 2016, 2013. *http://support.citrix.com/article/CTX137336*.

38. R. Shea, F. Wang, H Wang, and J. Liu. A deep investigation into network performance in virtual machine based cloud environments. In *INFOCOM, 2014 Proceedings IEEE*, pages 1285–1293, April 2014.

39. Minitab Inc. Minitab statistical analysis. Last Accessed: Feb 9, 2016, 2016. *https://www.minitab.com/en-us/*.

40. Minitab Inc. The Anderson-Darling Statistic. Last Accessed: Feb 10, 2016, 2016. *http://support.minitab.com/en-us/minitab/17/topic-library/basic-statistics-and-graphs/introductory-concepts/data-concepts/anderson-darling/*.

41. J. Allen. Python Connectivity Check Script. Last Accessed: Jan 25, 2016, 2010. *http://johnallen.us/?p=236*.

42. J.F. Kurose and K.W. Ross. *Computer Networking - A Top-Down Approach*. Addison-Wesley, 6th edition, 2013.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
24–03–2016	Master's Thesis	Aug 2014 — Mar 2016

4. TITLE AND SUBTITLE

Exposing Inter-Virtual Machine Networking Traffic to External Applications

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Byrd, Charles, E., CW3, USA

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-16-M-006

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Virtualization has become a powerful and fast growing technology. The Department of Defense is focused on taking advantage of virtualized hardware, software, and networks. Virtual environments create administrative and security challenges in having visibility of inter-virtual machine (VM) traffic. This thesis attempts to gain visibility and evaluate performance of inter-VM traffic. Separate virtual networks using VMWare ESXi and Citrix XenServer that comprise of three virtual host containing a computing domain of eight VMs. Configuration of all components are identical on each network for a consistent comparison. Transport-layer traffic is generated to test each network using batch files, Powershell scripts, and Python scripts. The results show standard virtual networks require additional resources and more hands-on administration for real-time traffic visibility than a distributed switch. Traffic visibility within a standard network is limited to using programs such as pktcap-uw, windump, or tcpdump. However, distributed networks offer advanced options, such as port mirroring, that deliver higher visibility but come at a higher latency cost.

15. SUBJECT TERMS

Inter-VM, vSwitch, ESXi, XenServer

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Barry E. Mullins, AFIT/ENG
U	U	U	U	107	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, x7979; barry.mullins@afit.edu