TAMPERE UNIVERSITY OF TECHNOLOGY

JARNO RUOTSALAINEN

HARDENING AND ARCHITECTURE OF AN INDUSTRIAL CONTROL SYSTEM IN A VIRTUALIZED ENVIRONMENT

Master of Science thesis

Examiners: Prof. Hannu Koivisto,
University Teacher Mikko Salmenperä
Examiners and topic approved by the
Faculty Council of the Faculty of
Engineering Sciences
on 1st November 2017

ABSTRACT

JARNO RUOTSALAINEN: Hardening and architecture of an industrial control system in a virtualized environment
Tampere University of Technology
Master of Science thesis, 62 pages
January 2018
Master's Degree Programme in Automation Technology
Major: Automation Software Engineering
Examiners: Prof. Hannu Koivisto, University Teacher Mikko Salmenperä
Keywords: virtualization, security, hardening, industrial control systems, automation

Virtualization is widely used in traditional ICT in order to share hardware resources between separate software applications while also creating isolation. This makes it possible to more efficiently utilize hardware resources as isolation doesn't require running software on separate hardware servers. Virtualization offers features like fault tolerance and the ability to create easily managed test environments. Such features are also desirable in designing and maintaining automation systems. Industrial control systems and their requirements differ significantly from traditional ICT, however. Security and reliability are of critical concern in ICS, and the effects of introducing new technology need to be thoroughly considered. Many practices that may be well-established and trusted in ICT can't be used directly in ICS, if at all. Industrial automation uses highly specialized solutions, and security measures can hinder or prevent system performance.

This thesis presents the main challenges and solutions related to using virtualization in industrial automation, with a focus on security and hardening. The virtualization platform used is VMware's vSphere 6.5, and thus the practical recommendations are aimed at VMware products. Much of the general design and security principles are also applicable in environments using different virtualization software. Automation systems are complex, and maintaining virtualization adds its own operational workload. Available scripting languages and programming interfaces are researched to find ways to decrease this workload by automating some of the maintenance tasks.

Automation systems are very heterogeneous and the integration of virtualization needs a lot of additional case specific consideration and practical work. Still, many of the established ICT solutions addressing virtualization security and hardening problems are found suitable for use in the ICS domain with some special considerations. Using the available VMware APIs and scripting solutions, practical tools automating security checks and hardening of virtual environments was developed.

TIIVISTELMÄ

JARNO RUOTSALAINEN: Teollisen ohjausjärjestelmän koventaminen ja arkkitehtuuri virtualisoidussa ympäristössä
Tampereen teknillinen yliopisto
Diplomityö, 62 sivua
Tammikuu 2018
Automaatiotekniikan koulutusohjelma
Pääaine: Automaation ohjelmistotekniikka
Tarkastajat: Prof. Hannu Koivisto, yliopisto-opettaja Mikko Salmenperä
Avainsanat: virtualisointi, tietoturva, koventaminen, teollisuuden ohjausjärjestelmät, automaatio

Virtualisointia käytetään laajalti perinteisen tieto- ja viestintäteknologian maailmassa rautaresurssien jakamiseksi ohjelmistojen kesken ja ohjelmistojen toisistaan eristämiseksi. Tämä mahdollistaa resurssien tehokkaamman käytön, koska ohjelmistojen eristäminen ei vaadi erillisiä fyysisiä palvelimia. Virtualisointi tarjoaa virheensieto-ominaisuuksia ja mahdollisuuden luoda joustavia testiympäristöjä. Nämä ominaisuudet ovat haluttuja myös automaatiojärjestelmien suunnittelussa ja ylläpidossa. Teollisten ohjausjärjestelmien vaatimukset kuitenkin eroavat merkittävästi perinteisestä tietotekniikasta. Tietoturva ja luotettavuus ovat kriittisen tärkeitä, ja uusien teknologioiden vaikutukset järjestelmään tulee tuntea tarkasti. Monet perinteisen tietotekniikan alalla vakiintuneet käytännöt eivät ole käyttökelpoisia automaatiossa. Teollisuusautomaatiojärjestelmät käyttävät hyvin erikoistuneita ratkaisuja, ja turvallisuusratkaisut voivat vaikuttaa järjestelmän suorituskykyyn.

Tämä diplomityö esittelee virtualisoinnin teollisuusautomaatiossa hyödyntämiseen liittyviä keskeisiä haasteita ja ratkaisuja. Käytetty virtualisointialusta on VMwaren vSphere 6.5, ja käytännön suositukset on suunnattu VMware-tuotteille. Suuri osa yleisistä suunnittelu- ja tietoturvaperiaatteista soveltuu myös muille virtualisointiohjelmistoille. Automaatiojärjestelmät ovat monimutkaisia, ja virtualisoinnin ylläpitäminen lisää oman työtaakkansa. Saataville olevia skriptauskieliä ja ohjelmointirajapintoja tutkitaan päämääränä löytää tapoja pienentää tätä työtaakkaa automatisoimalla ylläpitotoimintoja.

Automaatiojärjestelmät ovat hyvin heterogeenisiä ja virtualisoinnin integroiminen tarvitsee paljon tapauskohtaista suunnittelua ja käytännön työtä. Silti monet perinteisen tietotekniikan alalla vakiintuneet käytännöt osoittautuivat käyttökelpoisiksi myös automaatiossa erityisvarauksin. Käyttämällä olemassa olevia skriptauskieliä ja ohjelmointirajapintoja kehitettiin virtuaaliympäristön tietoturvatarkastuksia ja kovennusta automatisoivia työkaluja.

PREFACE

Writing this thesis allowed me to learn a lot about virtualization and related security practices. This acquired knowledge has already been of some practical use in my current job, and will no doubt be useful in the future. This thesis was written during a turbulent time in my personal life, and I'd like to thank everyone involved for their patience and understanding. A special thank you to Markku Tyynelä for guidance and offering me the chance to do this thesis. Thank you to Hannu Koivisto, Mikko Salmenperä and Jari Seppälä for their invaluable expertise and feedback.

Tampere, 20.1.2018

Jarno Ruotsalainen

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

API	Application programming interface
CLI	Command-line interface
COTS	Commercial-off-the-shelf
DoS	Denial of service
DMZ	Demilitarized zone
GUI	Graphical user interface
ICS	Industrial control system
ICT	Information and communication technology
IETF	Internet Engineering Task Force
IoT	Internet of things
IT	Information technology
LAN	Local area network
NFV	Network function virtualization
NPB	Network packet broker
NTP	Network Time Protocol
OOB	Out-of-band
OS	Operating System
PCS	Process control system
PLC	Programmable logic controller
PS	Microsoft PowerShell
RAM	Random-access memory
SCADA	Supervisory control and data acquisition
SDN	Software-defined networking
SDK	Software development kit
SNMP	Simple Network Management Protocol
TUT	Tampere University of Technology
VLAN	Virtual local area network
VM	Virtual machine
VMM	Virtual machine monitor
VNF	Virtualized network function
VXLAN	Virtual Extensible LAN
WAN	Wide area network

1. INTRODUCTION

Industrial automation has strict requirements for reliability, security and availability. When it comes to software, it is preferable to isolate programs in order to minimize the damage one software error or vulnerability can inflict on the overall system performance. Traditionally this has been accomplished by running software on separate hardware, which is not very efficient considering the hardware and maintenance costs and the physical space requirements.

Virtualization is a technology used in splitting or merging computing resources into new logical partitions. Virtualization makes possible the running of multiple isolated virtual machines on one hardware system. A virtual machine behaves like an isolated server with its own dedicated virtual CPU, RAM, disk space, and networking resources. Virtualization is widely used in ICT environments and it has recently started seeing increasing use in industrial environments. Industrial automation has traditionally been slow to adopt technologies used in conventional IT systems because of the stricter performance requirements and long life cycles of systems.

This thesis presents best practices pertaining to hardening, as well as security in general, and architecture of virtual systems used as a part of industrial control systems. Hardening refers to stripping a system of all unnecessary features and reconfiguring default settings to minimize the system's vulnerable attack surface. There is plenty of information on how to utilize virtualization and achieve passable security in traditional ICT environments. ICS, however, presents stricter security and performance requirements, and much of the ICT practices aren't applicable. Designing the architecture includes the choice of physical and virtual components and their placement and relations in the environment. This includes components related to computing resources, storage, networking, management systems, monitoring and so forth. All of these choices are a factor in the overall system security. In order to form an understanding of the problems related to ICS virtualization, the requirements and limitations of traditional industrial automation systems are discussed, and virtualization technology in general is explained. After this a review of the benefits and challenges of combining the two is presented. The primary goal of the thesis is to find out how to reap the benefits and address the challenges in an efficient manner.

A secondary goal is to research automating the management of the virtual environment. Virtualization presents worth-while benefits and makes some tasks easier, but utilizing it comes with additional operational workload related to managing and monitoring the virtualization platforms, virtual machines, virtual networking and so forth. Some of the tasks are very repetitive in nature, and automating them would lighten the operational workload considerably. This thesis' practical focus is on VMware products. Their official APIs, SDKs, and related scripting tools are studied to find possibilites for practical operational automation.

Chapter 2 explains the basics of industrial control systems and their typical special requirements. Chapter 3 discusses virtualization in general. Hardware virtualization and the concepts of virtual machines, hypervisors, and network virtualization are introduced. Chapter 4 addresses the special considerations needed in ICS virtualization.

Chapter 5 introduces central software tools considered and used in this thesis. VMware's virtualization platform vSphere's most relevant components are introduced. Programming interfaces, development kits, and scripting languages possibly beneficial in automating the virtual environments operations are presented.

Chapter 6 focuses on industrial virtual environment design, and in chapter 7, the topic of industrial virtual environment security is discussed more in depth. Former research on the subject is reviewed and security best practices on different aspects of virtualization are presented.

Chapter 8 presents practical proof-of-concept software tools developed to automate the security checks and hardening of virtual machines.

Chapter 9 articulates the conclusions and possible future work.

2. INDUSTRIAL CONTROL SYSTEMS

Industrial control systems (ICS) are comprised of hardware and software components designed to control, manage and monitor production machinery and related devices in an industrial environment. ICSs range from simple monitoring systems with few actual control duties to systems performing advanced data analysis combined with extensive automated and autonomic control functions. Modern ICSs are strongly coupled with automation and the term ICS is used mostly synonymously with the term *industrial automation and control systems* (IACS) preferred by the International Society of Automation. These control systems are used to make certain work tasks safer, more efficient or possible in the first place. ICS as a term encompasses various different (and somewhat overlapping) technologies and terms, such as *supervisory control and data acquisition* (SCADA), *process control system* (PCS) and *distributed control system* (DCS).

Figure 2.1 depicts a classic pyramid model of an industrial control system. This model consists of five levels with lower levels relating closer to the system and higher levels relating to enterprise level tasks. Generally gathered information is communicated upwards in the pyramid, while planning and control instructions based on the information flow down. The real-time requirements are strictest on the field level and become more lenient when moving up the pyramid. The time scale of actions performed by each level ranges from the field level's millisecond and even microsecond range of actuator and signal operations to business plans spanning several months on the management level.

While there are various slight adaptations from the classic ISA-95 model[30] and the naming convention varies according to source material, the main responsibilities of the hierarchy levels are quite consistently agreed upon as follows:

Field level Physical devices interacting with the actual process such as actuators and sensors.

Control level This level consists of different kinds of logical controller systems that send control signals to the field level devices. These systems typically reside in a single facility to locally achieve reliable and responsive real-time control.

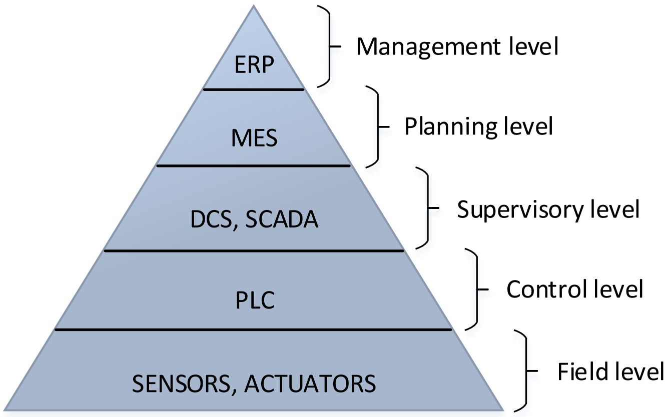


Figure 2.1 *Hierarchy model of industrial control systems.*

Supervisory level This level does the supervisory control and data acquisition over the lower levels. Also handles more advanced multivariable control based on analyzis of the gathered data. SCADA systems typically oversee more geometrically spread out systems of several local facilities or distribution networks.

Planning level MES stands for manufacturing execution system. It's the factory floor level planning system responsible for managing and monitoring production.

Management level Enterprise resource planning (ERP) is the name for a system managing a company's business processes such as product and production design, inventory handling, finances, and marketing.

While the pyramid model still does its job presenting theoretical hierarchy levels of an automation system, it's been foreseen to somewhat dissolve with the coming of Industry 4.0 and the increased connectivity and capability of lower level components. From a security and management perspective it is still beneficial to categorize and group system components according to their core functions and capabilities to make real-time requirements easier to achieve and to confine possible security anomalies. The levels of the pyramid aren't likely to completely vanish or merge in the near future, but the communication between levels is predicted to become more flexible.[38]

Distributed control system is a bit of an ambiguous term, but in this thesis it is used to refer to a "control level" system that consists of several separated controller subsystems with their own responsibilities rather than a system with centralized control. DCSs control industrial processes by using feedback loops and automated controllers to continuously ensure that the desired set point conditions remain in effect; DCSs are process-oriented and closely connected to lower level controllers interacting with sensors and actuators.

A common example of a “control level” controller is the programmable logic controller (PLC). A PLC is a specialized computer adapted to controlling industrial processes and equipment with an emphasis on high reliability. They are used extensively in controlling various types of industrial processes as parts of SCADA and DCS systems and as the main components in simpler control setups.

SCADA is a more data-oriented higher level ICS component that centrally gathers process and control data from the lower levels as well as sends control instructions based on analysis of the collected data. Traditionally, DCS has been defined as the closed-loop control system inside a single facility or small geometric area with its own supervisory features, while SCADA is the open-loop control system centrally overseeing the functions of several geographically spread out DCS field locations. While DCS can be considered to be driven by the state of the process, SCADA is more event-driven. [23] A power company, for example, could use DCS systems in its power generation and a SCADA system handling the distribution. The primary *human-machine interface* (HMI) for a control system is usually linked to SCADA or a comparable supervisory level subsystem. The HMI presents current and historical information about the actual process and the control systems, and is used to configure the set points and other parameters of the control.

In some models, SCADA is presented as part of DCS, and the two are seen as having started fusing together. With slower networks of the past local control was a necessity. SCADA system communications used to be slower and less reliable, with DCS systems focusing on smaller networks of high bandwidth and reliability. [35] Advancements in network technology and the consequent faster and more reliable long range connections have made it possible for SCADA to take on a more direct control role, and the increased processing power of lower level components allows for more distributed data analysis functions. This overlapping, merging and ambiguity of the terms can lead to some confusion. As one book by experts in the field put it: “ the most significant difference is in the local cultural usage of the terms related to the process.”[23]

The “planning level” MES manages production-related data regarding personnel, machinery, materials, costs, energy, quality assurance, inventory management, product data, recipes and other resources related to the production process. At this level, a highly detailed production plan is established and closely monitored to ensure that schedules and other production goals are met. This includes managing everything from personnel holidays and work shifts to scheduling maintenance tasks, both preventive and responsive, including handling replacement part inventory. [30]

Enterprise resource planning is a high-level business administration system. ERP systems can be used to manage and automate numerous business functions such as finances, marketing, sales, public and customer relations, order and inventory management, supply chains, and human resources. ERP lays out a rough plan for production to be refined on the MES level.

As was the case with SCADA and DCS, even the more official specifications of ERP and MES systems leave room for interpretation on the responsibilities and are somewhat vague when it comes to some specifics. Many of the concepts and technologies related to ICS are interlinked and the boundaries between them fluid. This is due to the control systems being utilized and advanced in various significantly differing fields of industry. There is still, however, on-going effort to shape more explicit standards and terminology. [28]

As ICSs control physical systems, they have different performance requirements compared to purely software systems. Whereas a failure in many software systems might lead to inconvenience or delays, performance-critical physical systems not meeting their requirements can lead to damage to the system, the environment, or personnel [55, 42]. ICSs are also designed to run continuously for extended periods and failures may lead to system stops and significant financial losses. This is why ICS requirements are in general strict when it comes to reliability, availability and real-time performance.

In industrial automation, life cycles of systems tend to be long. A several decade long life cycle for a control system is not unheard of. This is due to the high initial design and implementation costs combined with the relatively low operating costs of automation. Once the system is finally ready and running, it's profitable to utilize it as long as possible. Maintaining a system for decades, especially one related to rapidly advancing technology, can prove troublesome. Support for third-party software can cease and replacements for old hardware become unavailable. Replacing system components with newer ones can also cause compatibility issues.

2.1 Industrial control system security

The key objectives of information security are summed up in the so-called CIA triad: confidentiality, integrity, and availability. Confidentiality refers to limitations on access to information, integrity to assurance of accuracy and trustworthiness of the information, and availability to reliable access to the information. The CIA model can't be used to fully represent the security requirements in either complex ICT or ICS systems, but it can be used to point out some fundamental differences in how security is traditionally approached in the two domains. Security strategy for a general purpose IT environment is likely to have confidentiality as the highest priority. ICS systems, however, have traditionally prioritized the CIA objectives differently, placing more importance on availability. [31] The differing priorities are presented in Table 2.1.

***Table* 2.1** *Comparison of CIA priorities of ICT and ICS*

Priority	ICT	ICS
Higher	Confidentiality	Availability
↕	Integrity	Integrity
Lower	Availability	Confidentiality

Confidentiality is generally less important in ICS as much of the data isn't useful outside of the system context and current time window. Reliable availability, on the other hand, is often critical in continuously running systems with millisecond range response requirements of some subsystems.

A key concern of ICS security is that the systems control physical processes that are commonly required to run continuously. This may be due to straight- forward productivity reasons for a company or the system being part of critical infrastructure. ICSs are used to control systems related to large- scale infrastructure such as power, water, transportation, and communications.[25] In large communities, downtime of these kinds of systems can impact millions of people. A 100% availability is something to very actively aim for when dealing with such systems. The security implications of this are not only that a breach can't be allowed to stop the systems from functioning properly, but also that any security procedures (e.g. updates, reactions, recovery) shouldn't halt them either. Depending on its specific field and implementation, failures in an ICS's security can lead to several types of consequences, such as:

- loss of, or other harmful effects on, production
- harm to employees
- damage to the environment
- problems with infrastructure
- damage to software and hardware equipment
- theft of critical information
- damage to the company's image

This diversity of repercussions presents various possible motives for cyber attacks. The potential to do direct harm to the company, steal critical information, or when applicable, cause societal impact by disturbing critical infrastructure make ICSs tempting targets. Harmful effects on the infrastructure are possible even if it isn't the attacker's (or independently spreading malicious software's or user error's) actual intention.[36, 43] ICSs are also a prime target for recently thriving ransomware. Ransomware is a type of attack that disrupts a system unless or until a ransom is paid. This may be achieved by encrypting system files or otherwise restricting access to system functionality, or by threatening to release obtained critical information.

The security critical nature of ICSs necessitates special attention on connections to the outside world. For long, isolation and obscurity were the central information security safeguards for ICSs. The systems were kept separate from other networks and used proprietary software and networking solutions and were therefore quite safe from outside attacks. The Internet-capable business management and ICS zones were technically and often physically separated. With the advent of IoT and Industry 4.0, ICS systems have moved towards increased connectivity and utilizing generic commercial ICT solutions instead of the highly proprietary ad hoc systems. The growing use of commercial-off-the- shelf (COTS) products in ICS systems is a controversial issue. COTS components can bring cost savings and better usability, but they might not meet the special requirements of ICS systems and potentially carry more well- known and exploitable vulnerabilities.[42, 25] Integration is essential in modern ICS and TCP/IP is pervasive throughout automation systems. This means that from a security point of view, modern automation should be seen as a complex software product with all the benefits and vulnerabilites found in modern Internet environment.[21]

The increased unification of ICS with ICT has further revealed — and made it necessary to react to — security weaknesses of traditional ICSs. In 2010 , the "Stuxnet" computer worm designed to target PLCs made these weaknesses very tangible and can be considered a major wake-up call in the field. Another, more recent, major incident was the 2015 attack on the SCADA systems of three Ukrainian electricity distribution companies. The attack is considered the first successful attack on a power grid and resulted in approximately 225,000 customers losing power for several hours.[5] More connectivity requires better security. While ICT has had to deal with security threats for decades more, ICT specific security practices can't often be directly utilized in the ICS domain because of compatibility issues or potential effects on system reliability. This has lead to specialized ICS security solutions being increasingly developed by ICS research, standardization organizations, and companies.[10, 44] An example of such work was the Finnish 2014-2016 KYBER-TEO undertaking[2] aimed at advancing and implementing cyber security in industrial automation. The project set out to increase awareness on security issues and solutions in the ICS field. This was accomplished by forming and expanding collaborative efforts between security experts working for various enterprises and in research. Even if ICS security has gained more mainstream recognition after the major incidents and the amount of related attention and research has increased, there was major work in the field even before Stuxnet. For example, the mentioned KYBER-TEO was preceded by and builds upon such large-scale Finnish ICS security projects as TITAN, TEO-TT, COREQ-VE, and COREQ-ACT, with the TITAN project having started in 2008.

In addition to configuring the technical system for security, the responsibilities and roles of personnel in keeping the company secure need to be officially and clearly established. While this section may seem disconnected in a technical thesis, it is worth briefly noting that great technical security measures can be promptly undone by human error or malicious intent. As *Handbook of SCADA/Control Systems Security* [35] expressed it: “It may be trite and pedantic to say this — but security begins and ends with people.” This issue becomes even clearer when one starts to consider things such as out-sourcing where the training and work effort can’t be controlled or monitored directly by the organisation. The German ICT security agency Federal Office for Information Security ranked social engineering and phishing as the foremost threats to ICS security in 2016 [33]. Social engineering is the act of manipulating personnel into giving access to critical systems or information by exploiting human traits such as helpfulness, fear or respect for authority. Phishing is trying to scam personnel into providing crucial information usually with fake e-mails.

Not all social threats require an outside attacker. Inside threats consist of personnel with access to critical systems or inside knowledge pertaining to those systems. The access may be authorized or unauthorized, the potential damages intentional or unintentional. In any case, proper security measures have to be implemented to acknowledge and eliminate these threats or make the threat levels manageable. An example of a realized inside threat is the Maroochy Water Breach case of a disgruntled worker of an Australian SCADA equipment company. In 2000, after not getting hired for another job, the worker decided to exact revenge by accessing equipment he helped install in his previous job. He drove around with stolen radio equipment issuing commands to sewage control systems, causing 800,000 liters of sewage to spill out into the environment. [1] While technically already an outsider, the worker used inside information and skills acquired while working at the SCADA company to perform the attacks.

3. VIRTUALIZATION

Even when confined to the field of computing, *virtualization* is an ambiguous term covering various different methods related to creating and managing virtual versions of something. At its core it's a technology for splitting or merging computer resources into new logical arrangements. A simple example of virtualization would be partitioning a physical hard drive into many virtual drives. In the context of this thesis we're mostly interested in complete *virtual machines* (VM), i.e. hardware virtualization, and network virtualization, which will both be introduced in their designated sections of this chapter. Figure 3.1 shows a comparison between a traditional hardware computer setup, hardware virtualization, and so-called *containers*. In hardware virtualization (VM (virtual machine) setup in the figure) a software called the hypervisor creates a virtualization layer on top of which the virtual machines reside. Depending on the type, the hypervisor may or may not require an underlying operating system. Containers are a a more light-weight operating system level virtualization technology used mostly for isolating single applications in an environment with the necessary dependencies. Docker is a popular container solution. In this example, the underlying OS runs the Docker engine that is a software for creating and managing Docker containers.

Virtualization has been around for decades, originally known as *time sharing* in the 1960s. The need for multiple programmers being able to use “large fast computers” simultaneously was recognized, and the concept of “multi-programming” was born. The idea of multi-user computers lead to the development of an operating system component supervisor that was capable of provisioning hardware resources between multiple processes. The technology advanced and its utilization increased through the years, until virtualization really took off around the end of the millennium. This was largely because of prominent actors in the field such as the Linux Foundation, Microsoft, Sun, and VMware releasing more feature-rich and easily adaptable virtualization platforms. [11]

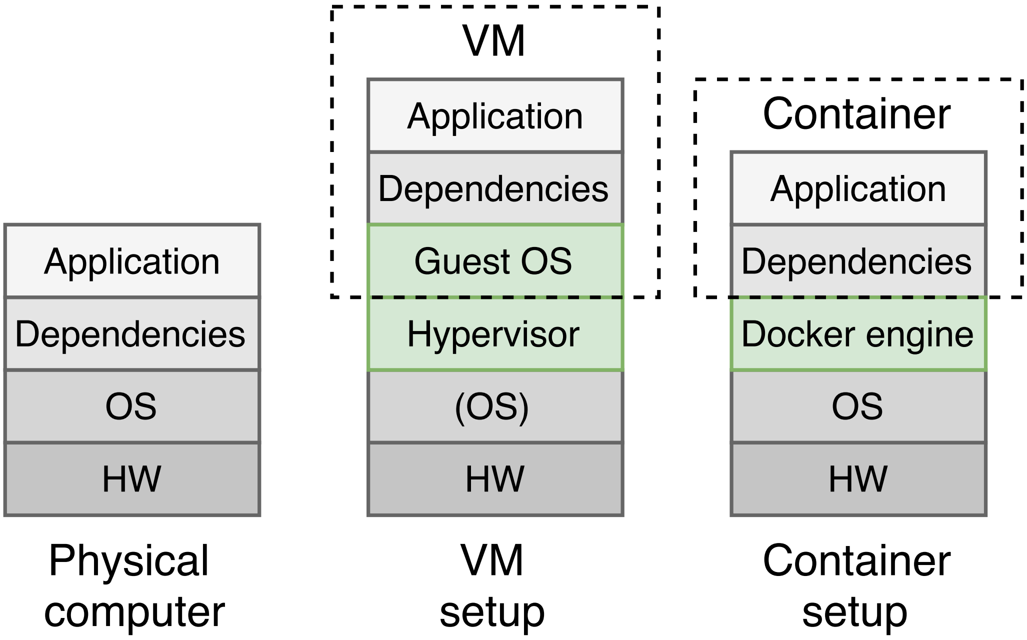


Figure 3.1 *Comparison of virtualization techniques and traditional computing.*

3.1 Hardware virtualization

Hardware virtualization is the earliest form of virtualization. It is used to create a level of abstraction between software and the underlying hardware. Pieces of software are presented with virtual representations of the hardware components as a means to access their designated hardware partitions. The hardware is fully virtualized and this makes possible the virtualization of complete computers as so-called virtual machines that are able to run unmodified operating systems.

The basic principle of hardware virtualization is presented in Figure 3.2. The virtualization layer is installed directly onto the physical hardware or on top of a host operating system. In virtualization terminology the actual hardware computer is called the *host* and the virtual machines it hosts are called *guests*. The virtualization layer handles the creation and management of the VMs and associates them with virtual hardware to act as an interface to the underlying physical computer. The guest operating systems are unmodified and perform similar to them running on a physical server.

3.1.1 Hypervisor

Hardware virtualization utilizes so called *hypervisor* software to emulate computer hardware for other software. Hypervisors are also called *virtual machine monitors* (VMM).

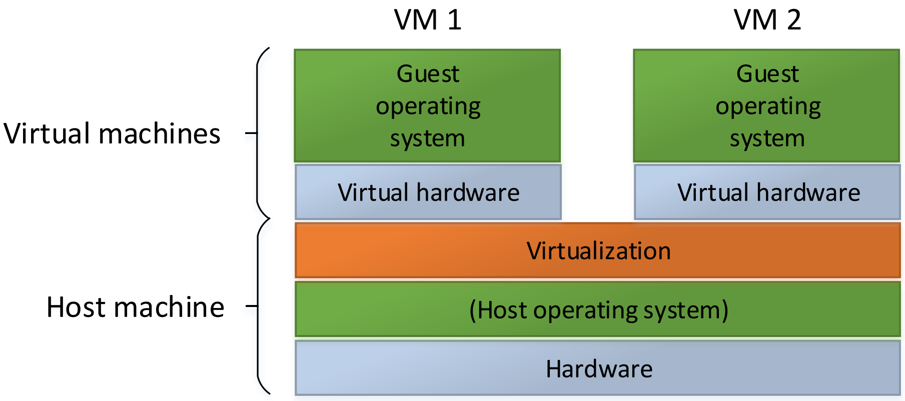


***Figure* 3.2** *System composition of hardware virtualization.*

Some sources and vendors (mainly VMware) use the term VMM to mean the software component responsible for the hardware abstraction and functionality of a single virtual machine, while still calling the virtualization layer in its entirety a hypervisor. [46] A hypervisor may itself contain kernel functions needed in communicating with underlying hardware. Such hypervisors are called *type 1 hypervisors* and are installed directly on computer hardware. They are also called native or bare-metal hypervisors. Prominent examples of type 1 hypervisors are Oracle VM, VMware ESXi, and Citrix XenServer. *Type 2 hypervisors*, also known as hosted hypervisors, lack these kernel features and must be installed on a host operating system that provides the channel for hypervisor-hardware communication. Type 2 hypervisors include VMware Workstation, VMware Fusion, Solaris Zones, and Oracle Virtual Box. There are hypervisors that don't fit in either of the above types. A well-known example is the Kernel-based Virtual Machine (KVM). It is a Linux kernel module that adds virtualization capabilities to the kernel.

Modern processors compatible with the x86 instruction set architecture use a special operational mode called *protected mode* during run-time. This is their native operational mode and introduces four *privilege levels* to restrict software's direct access to kernel functions. The privilege levels, also called *rings* as they're often depicted as concentric rings, are pictured in Figure 3.3. Software running at the most privileged level 0 has direct access to kernel functions. This level is also known as the *kernel mode*. Typically only the operating system's integral functions run at level 0, with levels 1 and 2 used by OS services such as drivers, and user applications running at the least privileged level 3, also called the *user mode*. These privilege levels enable the operating system to control different tasks' access to system resources.[8]

As type 1 hypervisors are installed directly on to the host machine and act as the operating system for the hardware, they operate at level 0. Hosted virtual machines run at level 3.

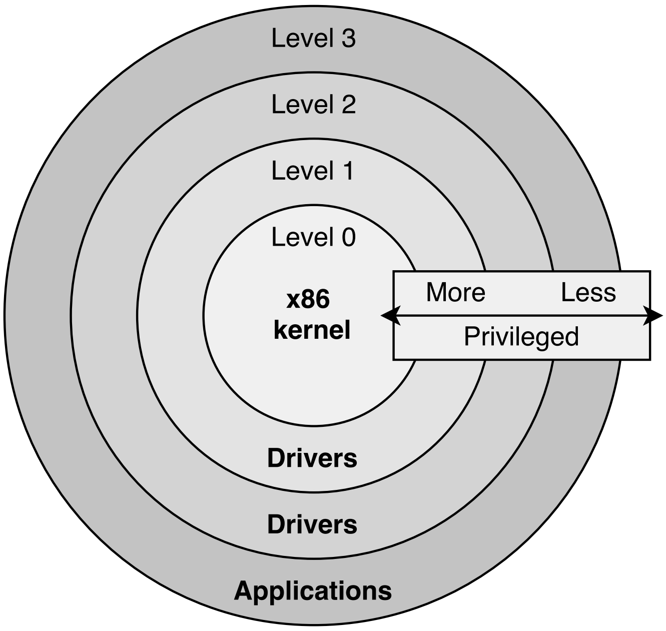


Figure 3.3 *The privilege levels of the x86 architecture.*

Type 2 hypervisors are installed on top of an operating system and thus operate at the same level 3 as the VMs they host and access the kernel functions through the underlying OS. Still, the objective of both types of hypervisor is to create a virtualization layer between themselves and the hosted virtual machines and to allow the VMs to operate without directly accessing the actual level 0 functionality. The hypervisor presents the VMs with a virtual version of the privilege level 0, so the guest can make standard system calls when it requires system resources.[39]

3.1.2 Virtual machines

The main reason for utilizing virtualization is in most cases the ability to use virtual machines in lieu of physical servers. A VM consists of virtual hardware allocated by a hypervisor and uses its dedicated share of the underlying physical computer's resources: its own virtual CPU, RAM, hard drive, and networking capabilities. Generally VMs behave and interact with other machines just like their physical counterparts and can run unmodified operating systems. Multiple VMs can coexist on a single computer and work in isolation.

More concretely, virtual machines are made up of files containing their configuration information and content. For example, a VM called "VM1" created on VMware ESXi would consist of the following core files:

VM1.vmx Primary configuration file defining the VM's virtual hardware and the availability of several optional features. This file is plain text and intentionally easy to edit.

VM1.nvram The BIOS configuration file of the VM.

VM1.vmdk Configuration file for the VM's virtual disks. Specifies the file system information and location of the virtual disk used.

VM1-flat.vmdk The actual virtual equivalent of a hard drive where all the VM's raw data is stored. A VM can have several of these disk files in use depending on how the disk is created.

vmware.log The VM's current log file. Depending on logging settings, older logs can be found as vmware-n.log files, where n is a running number.

Some additional files related to snapshots, memory swapping and configuration of the VM may also be present initially or later on depending on the machine's configuration and runtime operations. Snapshots preserve a VM's state at the point when the snapshot is created. A snapshot consists of all of the files that make up the virtual machine, such as the virtual disks, memory, and other virtual devices. These files include include information on the power state and by default the full state of the VM's memory which makes it possible to roll back and resume the VM's execution from the exact state at the time of the snapshot.

3.2 Network virtualization

Network virtualization is a term used to refer to abstraction technology that decouples a network from the underlying physical equipment. It can be used to create virtual networks by altering network topology without changing the layout of physical network components, or by creating entirely virtual networks of software components and virtual machines inside a server. The former is called an external virtual network and the latter an internal virtual network. Virtual networks can be used to isolate already existing traffic in separate zones for increased security or to achieve higher utilization of the network. Network virtualization can make the design and management tasks significantly easier and more flexible, as software can be used to change the logical network structure.

The logical separation of the physical network into several virtual ones is achieved by a method called *tagging* or *labeling*. It refers to tagging transferred data with information on which virtual segment of the network it belongs to. Such tags along with additional control information are added to network frames to tell network devices where to forward the data. For example, using the IEEE 802.1Q protocol[32], an additional 802.1Q header is inserted into the Ethernet frame, and the tag is stored in a 12-bit identifier field allowing 4094 unique VLANs (4096 minus two values reserved for special cases). VLANs can be configured using various different protocols, IEEE 802.1Q being the most widely used one. IEEE 802.1Q was introduced as a standardized non-proprietary solution to creating VLANs. There are various other protocols that can be used to tag network traffic and achieve VLANs. Some are derivative protocols like the 802 .1Q amendment 802.1ad (also known as Q-in-Q) which makes it possible to assign multiple tags to a single frame, and thus enabling some advanced network topologies. There are also proprietary protocols like Cisco's *Inter-Switch Link Protocol* (ISL) and *Dynamic ISL*, and more collaborative efforts such as *Virtual Extensible LAN* (VXLAN)[24], which was developed primarily to increase the number of VLANs on a network to 16 million especially to accommodate the growing numbers of virtual machines in large data centers. ESXi hosts offer three options for tagging traffic with 802.1Q headers and thus enabling VLANs to be used with VMware virtual environments: External Switch Tagging (EST), Virtual Switch Tagging (VST), and Virtual Guest Tagging (VGT). With EST, all of the tagging is performed by an external physical switch. With VST, the tagging is handled by a virtual switch before the traffic leaves the host. VGT also does internal tagging, but the tagging is performed by a virtual machine.

Software-defined networking (SDN) is a network architecture that aims to make networks more programmable and dynamic by decoupling network control and traffic forwarding functions (i.e. the *control plane* and *data plane*) from each other.[12] The control plane refers to all functionality related to routing the traffic through the network. The data plane handles the forwarding of network traffic between interfaces according to packet header information. An additional *management plane* is used to configure and monitor the control plane settings. The distinction between traditional and software defined networking principles can be seen in Figure 3.4. In addition to the separation of the control and data planes, SDN is flow-oriented, meaning both the network traffic origin and destination can be considered in forwarding decisions instead of just the packet destinations. There are several protocols that comply (to somewhat differing extents) with SDN's core principles of open standards, programmability, central management and dynamic reconfiguration. One of the most prominent protocols, and earliest to adapt the SDN architecture, is OpenFlow. OpenFlow is an open protocol proposed in a 2008 whitepaper[27] as a way to reprogram the internal flow-tables of switches and routers from various vendors.

Other SDN protocols and technologies include ones developed by companies, such as Cisco's *Open Network Environment* (ONE), as well as ones developed by open com-

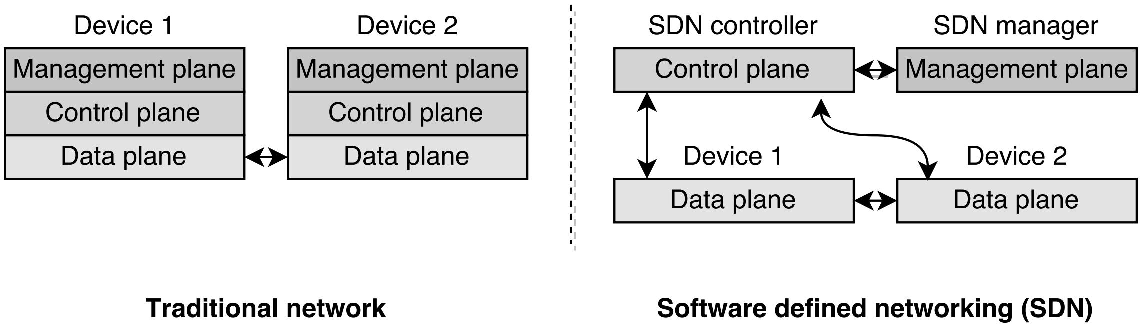


Figure 3.4 *The difference between traditional and software defined networking.*

munities, such as the Internet Engineering Task Force's (IETF) *Network Configuration Protocol* (NETCONF), and *Multiprotocol Label Switching - Transport Profile* (MPLS-TP).

Decoupling network functions from underlying hardware is known as *network function virtualization* (NFV). Setting up or reconfiguring networks utilizing traditional physical devices such as firewalls, load balancers, and intrusion detection/prevention systems can be time-consuming and expensive. NFV aims to increase efficiency and reduce the operating and capital expenses by using virtual implementations of the appliances. The difference between the traditional network functions and NFV can be seen in Figure 3.5. Whereas traditional networks rely on a plethora of proprietary hardware devices to perform specific network functions, NFV performs them as fully software *virtual network functions* (VNF) on generic hardware. This makes it easier to setup and modify networks as software changes on COTS hardware are generally faster and more inexpensive to perform than ordering and setting up specialized physical appliances. A specification of NFV released by the European Telecommunications Standards Institute (ETSI) in 2013[17] introduced three main domains that form the NFV framework as seen in Figure 3.6.

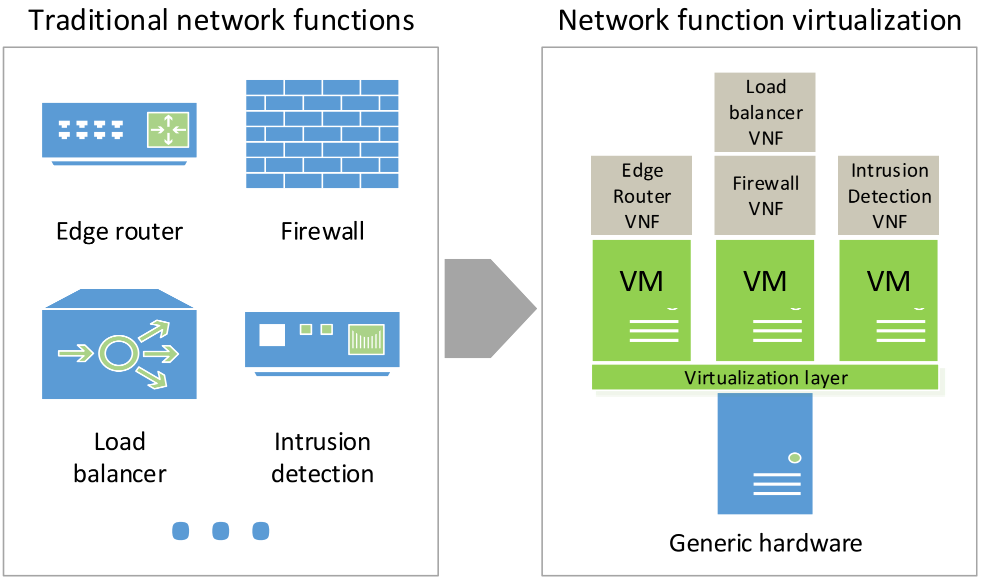


Figure 3.5 The basic principle of network function virtualization.

The three main domains identified in the specification are:

Virtual network functions A single virtual appliance in the NFV architecture is called a virtualized network function (VNF) and consists of one or more virtual machines running the required software.

NFV infrastructure NFVI domain relates to all of the hardware resources used by the NFV system and their virtualization. The VNFs are run on virtual machines hosted on COTS computer hardware as part of the NFVI.

NFV management and orchestration NFV-MANO domain manages the orchestration and lifecycles of the VNFs as well as the hardware and software components needed in NFVI.

Network virtualization, SDN, and NFV share common goals. They're all aimed at bringing about more programmable software-based networking to achieve more dynamic, efficient, and flexible networks. While related, they aren't conceptually dependent on one another. However, the complementary symbiosis of the concepts has been recognized. They are often used in concert in implementations of modern network systems, as well as in novel networking technologies. SDN can be used to enable network virtualization without any support from the underlying hardware by utilizing overlay networking with virtual switches forwarding the traffic. Network virtualization can be used in testing SDN before deploying it on the real production network. NFV can be used to provide an infrastructure to run the SDN software on. [12, 29]

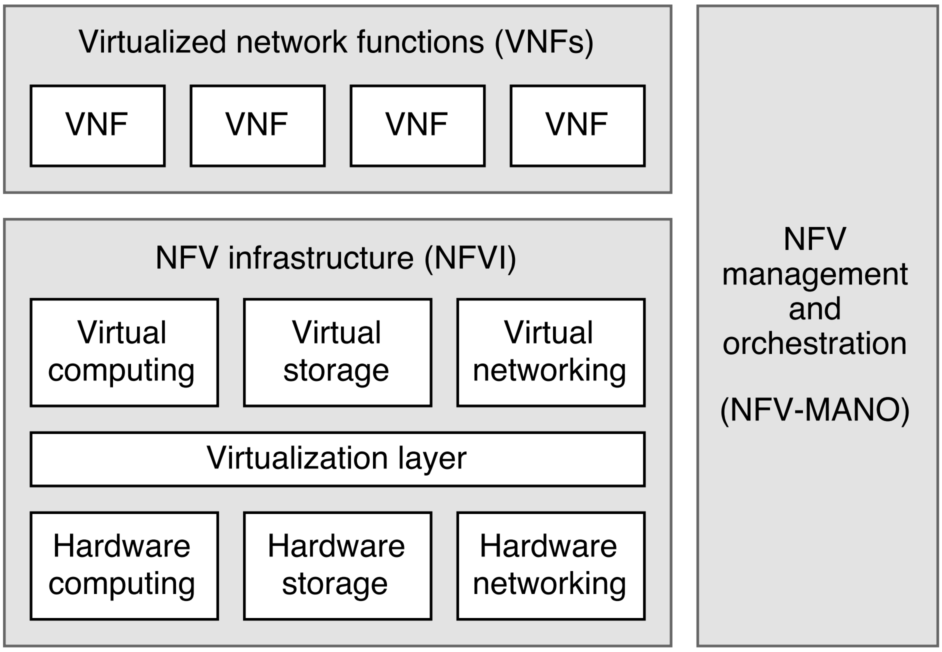


Figure 3.6 *NFV framework (modified from [17]).*

Virtualization vendors offer varying ranges of virtual versions of networking components such as switches, network interface controllers (NIC), firewalls and miscellaneous virtual appliances specialized in virtual traffic monitoring and control. Virtual switches are used to connect virtual network components together just like physical switches on traditional networks.

VMware offers two types of virtual switches to use with current ESXi hypervisors depending on the chosen license: the regular virtual switch called the *vSphere Standard Switch* or *vSwitch*, and a so-called *vSphere Distributed Switch* (vDS or VDS), which is a virtual switch that exists across two or more hosts on the same cluster. The vDS technology utilizes methods familiar from SDN described above. The vSwitches consist of two logical planes with their separate responsibilities. The data plane handles the actual data packet related functions like switching and tagging, while the control plane is used to control the data plane behavior. When using regular vSwitches, both the control and data planes reside on the host machine. The distributed switches are basically an abstraction of normal vSwitches to achieve a cluster-wide centrally accessible control plane while the data planes remain local on every host. The principle of vDS can be seen in Figure 3.7. The vDSs enable virtual machines to maintain their network configuration while migrating from host to host. The vDSs are managed by vCenter, with one vCenter instance being able to theoretically support up to 128 vDSs that in turn can manage up to 500 hosts each. The vDS can be used to simplify the virtual network structure and make configuration operations easier. Not having to configure vSwitches on numerous hosts can be a huge time-saver if the desired network layout and traffic type isolation allows using distributed switches.

In addition to providing switching spanning multiple hosts, VMware's Distributed Switches offer additional security features compared to the standard vSwitches. When using vDSs, reliable vCenter availability becomes even more important. Even though the vDS structure with the data planes remaining locally on the hosts means that a vCenter outage doesn't stop the network traffic, it does at least momentarily stop administrators from accessing vDS operations.

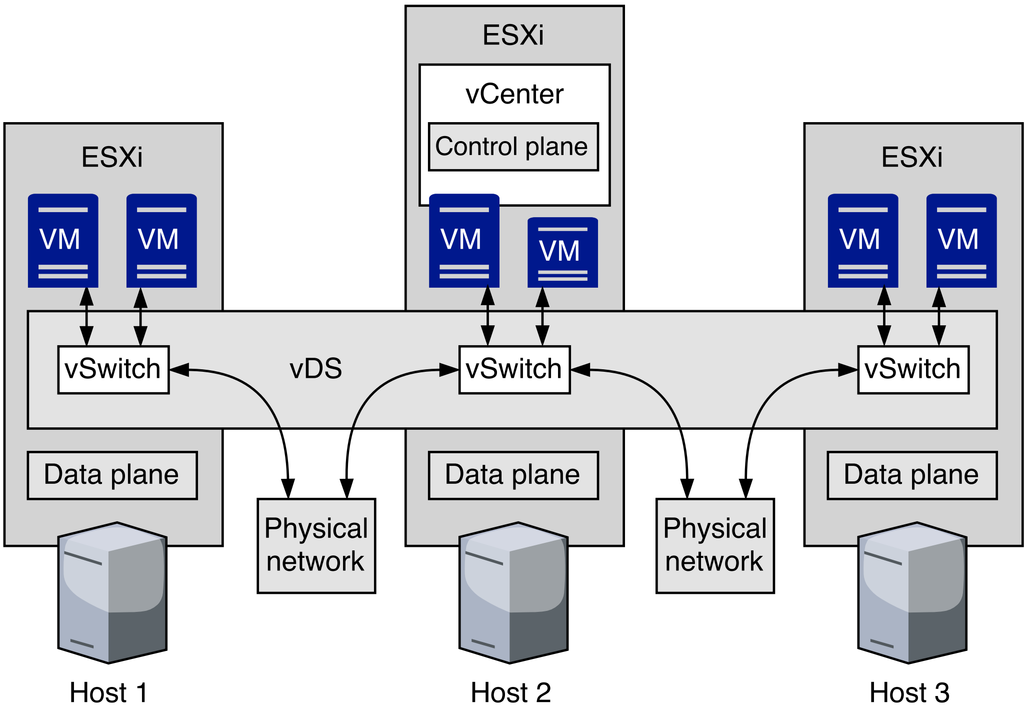


***Figure* 3.7** *VMware vSphere Distributed Switch (vDS).*

Virtual local area networks (VLAN) are an example of external virtual networks. VLANs allow for combining physically separated network segments to behave as if connected to a common local area network (LAN). VLANs can also be used to separate traffic types traveling on the same physical network. Figure 3.8 shows a network setup with both external and internal virtual networks. Servers on different LANs have been joined together to form VLANs. The VLANs work in isolation despite using shared physical network cables and devices. One server hosts a virtual environment containing an internal virtual network. If the internal virtual network has connections outside its physical host through a physical NIC it is considered part of an external network. The individual virtual machines can belong to different internal or external VLANs.

Virtual private networks (VPN) are another widely utilized virtual networking technology.

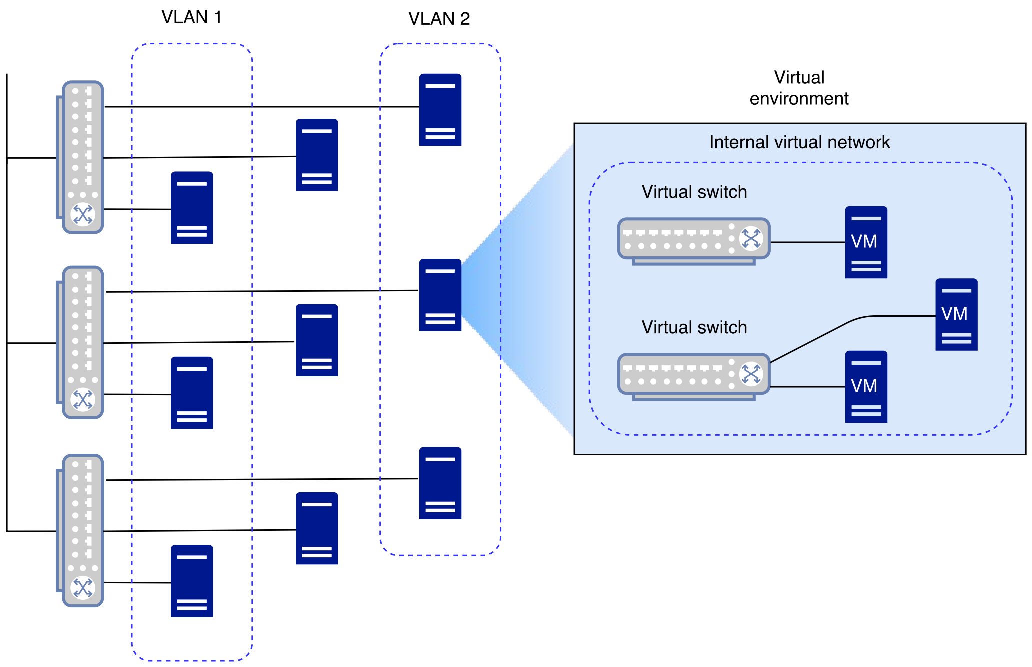


Figure 3.8 *Network setup with external and internal virtual networks.*

A VPN can be created to remotely access closed networks over an untrusted network, such as the internet. This is achieved by creating an encrypted communication channel from the remote machine to the desired network. The remote client can then access the use an IP address from the local subnet to access the target network.[26]

In ICS, the more traditional layer 3 VPN solutions, as well as other network virtualization methods, can't often be used. This is due to ICS systems utilizing special industrial protocols to achieve the required real-time communications. These protocols can come with their own limitations and requirements regarding e.g. packet size, used applications, and licensing, and can already make use of tagging traffic. Tunneling these protocols without losing the functionality they provide can often only be done by using layer 2 tunneling. The subject of layer 2 Ethernet tunneling possibilities is covered in-depth in Matias Leinonen's aptly named master of science thesis " Layer 2 Ethernet Communication Tunneling Possibilities In Automation Systems" [22].

3.3 Advantages and disadvantages of virtualization

While virtualization can offer solutions to various problems in both ICS and ICT contexts, the administrational and security challenges it entails can't be overlooked. This section discusses some of the major pros and cons brought upon by virtualization.

Perhaps the main motivation behind using virtualization is efficiency. Being able to use virtual machines instead of separate physical computers to achieve isolation for system components can result in savings in time, hardware costs, maintenance, energy consumption and physical space requirements. A simplified example of the resource efficiency of a virtual system compared to a physical one if presented in Figure 3.9. If the resource usages of the individual hardware servers are low compared to their overall capacities, their functions could be more efficiently implemented by running corresponding virtual machines on a single hardware server. Setting up traditional servers can take days or weeks with having to order, wait for, and set up the new hardware. Deploying additional VMs can be done in a matter of minutes or hours, available resources permitting. The ease of VM creation may lead to problems in a poorly managed environment. Undisciplined addition of VMs that are over- provisioned with shared resources, patched improperly, or simply forgotten after outliving their usefulness, can lead to so-called VM sprawl.[48] VM sprawl can cause unneeded use of resources and security issues if the system isn't being monitored and maintained properly. It's crucial to keep good track of a system's virtual inventory and to ensure all VMs are properly provisioned, updated and necessary in the first place. The threat of sprawl applies to other aspects of virtualization as well. For example, the amount virtual networks may get out of hand and make it difficult to relate the virtual resources to the utilized physical ones.

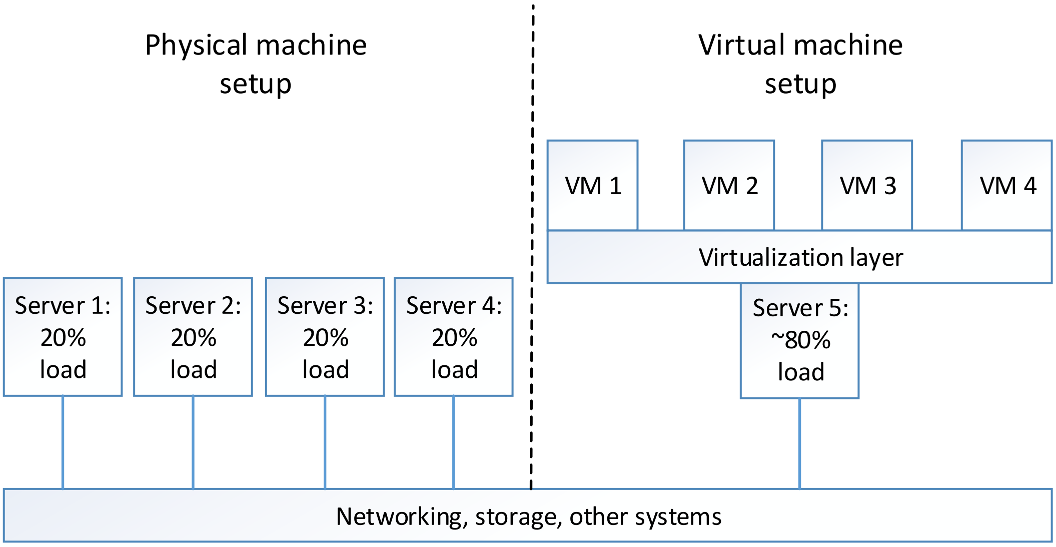


Figure 3.9 *Comparison of physical and virtual system setups.*

While virtualization can make maintaining some aspects of a system faster and easier, it also creates additional maintenance needs. Moving from a physical setup to a virtual one as they're presented in Figure 3.9 would reduce the amount of hardware maintenance and physical space needed as the amount of hardware machines goes from four to one. On the other hand, the amount of workload related to the internal function of the servers is likely to stay the same, as the virtual machines need overseeing just as their physical counterparts performing the same tasks. Factoring in the additional overhead of maintaining the virtual environment, the final operational workload of managing a virtualized solution can be higher. [19]

Among the things not portrayed in Figure 3.9 is the resource usage overhead caused by virtualization. This overhead isn't usually very dramatic, but depends on the function and amount of the virtual machines. A VM doesn't communicate directly with hardware, but instead through an interface offered by the virtualization layer. Therefore there's an added cost attached to all hardware calls made by the virtual machines and the virtualization software itself takes up additional resources. [6]

Whereas the ability to achieve high utilization of hardware resources is one of the main attractions of virtualization, having several services on the same physical host or network will make the consequences of hardware failures more severe. Virtualization platforms offer virtualization specific fail-safe and recovery features, however, and virtual infrastructure being used in a performance critical setting should be designed to tolerate errors just as any other subsystem. The design principles are addressed in chapter 6.

Several virtualization software providers offer support host clustering and resource pools. This means joining several hardware computer's resources into a larger virtual resource pool shared between virtual machines being run on the system. This lets the user manage the resources as if the cluster was a single host and a virtual machine can use resources from anywhere in the cluster. Resource pooling makes possible some advanced availability and optimization features. If there are enough resources left, virtual machines can be automatically relocated and restarted if a single host on the cluster goes down. Some virtualization platforms offer automated optimization inside the cluster whenever there are hardware changes or VM resource usage changes.

4. VIRTUALIZATION IN INDUSTRIAL AUTOMATION

This chapter briefly discusses special considerations needed in using virtualization in industrial automation. Current modern virtualization technologies haven't been in use for long and industry is generally slower in adopting advancements compared to ICT. Companies have their own virtualization research and implementations, but the amount of academic research on the subject of utilizing virtualization in industry is still lacking.

Currently ICS virtualization consists mainly of running SCADA and other higher level components (see Figure 2.1) on virtual machines. This is due to most of the COTS hypervisors being optimized for ICT use and not for real-time applications. The system elements that need to comply with stricter real-time requirements, i.e. PLCs and the like, are still less commonly virtualized. However, even virtual PLCs have been recently deemed feasible at least in systems without absolutely uncompromising real-time requirements[10]. Ari Lappalainen presents additional studies supporting virtualization's potential in real- time ICS in the literature review portion of his thesis[20].

4.1 Virtual redundancy

As brought forth in chapter 2, industrial control systems have strict reliability requirements. Virtualization can be used to reduce single points of failure by adding redundancy and recovery measures. This can be achieved by utilizing the fault tolerance features of modern virtualization platforms. Live migration or keeping live duplicates of virtual machines in case of failures allows high availability and reliability. Available VMware platform features are presented in the following chapter 5.

4.2 Development and testing

Industrial control systems can be very heterogeneous. It's typical for a system to consist of thousands of hardware components from different manufacturers along with a overabundance of associated software. Verifying the proper function of systems of this magnitude is a complicated task. Virtualization can be used in the development phase to try out different design approaches thanks to the flexible nature of virtual environments. Several development teams can use shared hardware resources if provided with separate virtual environments to work with. Making such use of virtual machines and networks can make the design and testing processes more agile as well as lower the costs when compared to physical servers. Changes to the system setup in a virtual environment can be applied quickly as centralized file operations and snapshots of the system allow for easy resets and roll- backs when necessary. This makes it easier to try out different setups and scenarios and the tests can easily be repeated from a similar start point to validate the results. It also makes it easier to switch the test environment to represent possible different versions and configurations of software in use by different clients or facilities.

However, if a system is going to be used in a physical environment, it should finally be tested in its actual working setting. While virtual test environments can aid in the development process, they aren't identical to the actual physical production environments.

4.3 An interview with professionals in the field

An interview with ICS professionals dealing with virtualized systems was conducted to gain insight into current real-world security practices and to find out needs for improvement. The interview revealed that while there were public and nascent company guidelines for virtualization security, they hadn't yet been properly adapted in actual system implementations. This goes to show the demand for not only proper motivation and training but also for continuous feedback from personnel on how the planned methods are being applied in practice and what needs to change.

Another interesting takeaway was the unwillingness of customers to invest in offered security measures. Especially for smaller companies the added cost of better security can be substantial and they opt out as the costs are seen as outweighing the benefits. This attitude may be due to ICS security risks still not being taken seriously, or misunderstanding the main reason to invest in security. Security should be viewed as a tool to achieve more dependable systems and higher availability brought upon by the decreased downtime it provides. Another reason might be a so-called illusion of insignificance: why would anyone attack a small company while there are more attractive bigger targets available? It is worthwhile to emphasize to potential customers and system developers alike that a smaller less secure target can be more attractive than a larger secure one. The motivation for all cyber attacks isn't potential gain for the attacker or harm to the target either. Some attacks are just acts of recreational vandalism arising from opportunity and challenge.[53]

5. SOFTWARE TOOLS

Virtual environment size can vary greatly from small setups with a single host and a few VMs to extremely vast systems with thousands of hardware servers hosting several hundreds of thousands of virtual machines. While the amount of machines in an ICS context isn't likely to rise to such extremes, it's best practice to use available tools to configure and automate system management as well as possible. This chapter introduces some essential VMware and third-party tools and considers their usefulness in industrial automation.

5.1 VMware vSphere

VMware Inc., a subsidiary of Dell Technologies is a company specialized in virtualization software and services. vSphere is the name of VMware's virtualization platform and the related product family. vSphere includes, i.a., the type 1 hypervisor ESXi and the virtual environment management tool vCenter, which can be used to access and manage the product family's other components. The information and proposed practices presented in this thesis are related to the most recent version, which is vSphere 6.5 (released in October 2016) as of writing. The core components and features included in vSphere are presented in Table 5.1. The availability of the more advanced features depends on the acquired license. Their usefulness in the system being designed should always be weighed against the current license packages and pricing.

The vSphere Hypervisor also known as ESXi is VMware's latest type 1, or bare-metal, hypervisor. ESXi is installed directly on a hardware system with no intermediary operating system. It includes a so-called VMkernel OS that includes the kernel capabilities for accessing physical hardware and provides the hypervisor with necessary functionality such as a file system, resource allocation, and tools for managing the virtualization framework.

VMware vCenter Server is a centralized management platform for vSphere. It let's the user manage both virtual machines and hosts. It is necessary to use vCenter if you need access to vSphere's more advanced features such as vMotion, High Availability, Distributed

Table 5.1 *VMware vSphere Components and Features*

Name	Purpose
ESXi	VMware's hypervisor that abstracts hardware resources into virtual machines.
vCenter	The centralized management tool for the VMware virtual environment.
vMotion	A feature that enables the live migration of VMs across hosts without service interruption.
Storage vMotion	Enables the live migration of VM files across datastores without service interruption.
High Availability	Automated restart feature in case of hardware or software failures.
Fault Tolerance	Provides continuous availability by maintaining a live copy of a VM ready to take over in case of failures.
Data Protection	Backup and recovery functionality for VMs.
vShield Endpoint	Antivirus and antimalware protection for VMs.
Distributed Switch	A more advanced interhost virtual switch.
Distributed Resource Scheduler	DRS allocates and balances computing resource use of VMs across separate hardware resources.
Storage DRS	Allocates and balances storage resources across separate storage resources.
Host profiles	Simplifies host configuration management through user-defined per host configuration policies.

Resource Scheduler, or Update Manager.

VMware vMotion enables the migration of running virtual machines from one host to another without interfering with the VM's function. This is beneficial when maintenance, updates, or whatever hardware changes need to be performed. The new host must fulfill resource and compatibility requirements before a migration can commence. Possible compatibility problems might include trying to migrate from an IPv4 host to an IPv6 host, or to a host not able to access a physical device in use by the virtual machine. After compatibility and the stability of the current VM state is confirmed, the VM state information is copied to the new host. The VM then resumes its function on the new host. At least 250 Mbps of dedicated bandwith per possible concurrent migration should be reserved for vMotion. [52]

VMware High Availability (HA) and Distributed Resource Scheduler (DRS) are features made available by merging hosts into clusters in a vSphere environment. DRS makes shared resource pools in clusters possible. It also provides automated resource allocation and optimization when resource availability or usage in the cluster changes. HA is a recovery feature that attempts to relocate and restart virtual machines on another host in a cluster if a host machine fails. This isn't by any means a real-time recovery, as the new VM needs to power up before resuming the interrupted process. A near zero-latency redundancy solution can be achieved by using vSphere's Fault Tolerance (FT). FT creates a replica of a VM to run in continuous lockstep on another host. If the primary host crashes, the replica VM takes over instantly, and, resources permitting, another replica is created. Naturally, keeping a replica VM in near zero-latency sync takes up a lot of resources and can get costly if found necessary.

VMware's vRealize Orchestrator (vRO), formerly vCenter Orchestrator, is a tool for creating and running management processes for different VMware and third party components in a virtual environment managed by vCenter. vRO gives the user a more complete access to the VMware API than vCenter does making additional management operational available. vRO offers a graphical interface and presents operations as workflow charts. This can be seen as an advantage as the workflows can be easily perceived through their visual representations.

5.2 Scripting

Optimizing management and minimizing the administrative overhead is desirable for all enterprises. Although the vCenter GUI offers approachably presented graphical access to vSphere components, using it alone to manage a large-scale virtual environment isn't all that feasible. Many of the administrative tasks are repetitive with little variation. Applying changes to or reviewing numerous virtual machines or hosts through the GUI can be arduous and some features simply inaccessible. This is why additional command line use and scripting are needed to enable and automate efficient vSphere management. The performance of ICS systems needs to be closely followed and scripting also makes it easier to conform to case specific monitoring needs.

Scripting refers to the creation and running of small programs that automate some domain specific tasks. Various different types of scripting exist from operating system specific shell scripting to more specialized scripting languages to very specific vendor provided solutions. “Scripting languages" ' are interpreted high-level languages that don't need to (but most still can) be compiled before execution, e.g. JavaScript, Lua, and Perl. Then there are languages that can be interpreted even though they are often compiled, like Python and Ruby. In the end, the terms scripting and scripting language aren't that much about the language itself but the way it's applied. If a program's goal, rather than being a stand-alone application, is to control or gather information from other software, it can be called a script.

Several virtual platform vendors, including VMware, offer extensive APIs and SDKs that can be leveraged to make virtual infrastructure management more efficient. These can be accessed via various proprietary as well as open source interfaces for different tools and languages. Short descriptions of the most prominent choices for managing vSphere environments follow.

PowerShell (PS) is the name for Microsoft's command-line shell and the associated scripting language. It's designed to execute and automate tasks in a Windows environment. PS can execute special functions, scripts, and so- called *cmdlets*. Cmdlets are small .NET programs that interact with PS. VMware PowerCLI is a collection of cmdlets designed for automating the management and configuration of several vSphere products through PS. It provides a PS interface for accessing the VMware product APIs.Whereas traditional PowerCLI only works on Windows machines with PowerShell enabled, PowerCLI Core is a port that extends PowerCLI support to Linux, Mac, Docker, and VMware Photon OS. However, at the time of writing PowerCLI Core only includes a portion of PowerCLI's cmdlets, and as such can't yet be considered a full-fledged alternative.

VMware's vSphere Command-Line Interface or vCLI is set of Perl language commands used to remotely access vSphere. It's paired with vSphere SDK for Perl in that many of its commands require the SDK. As vCLI can be run on any machine with Perl installed, it offers better cross-platform flexibility than PowerCLI. It can be used to administer ESXi servers from any networked Perl-capable machine. It can additionally control vCenter servers and thus also indirectly target the ESXi servers managed by the vCenter.

Python is a general-purpose high-level programming language and pyVmomi is the official Python SDK for the vSphere API. It was released by VMware and its function and object names map directly to those presented in the vSphere Web Services SDK. This offers a extensive cross- platform method for managing ESXi and vCenter on any Python enabled machine.

6. INDUSTRIAL VIRTUAL ENVIRONMENT DESIGN

As with physical systems, virtual environments can be set up in various ways and often with no objectively best option. Decisions regarding the environment's logical structure affect things such as availability, security, scalability and performance. This chapter introduces some general best practices in virtual environment design while considering the requirements of industrial control systems. While the practical focus of this thesis is on creating and testing an environment using VMware products, the core design principles can be universally useful.

As mentioned in chapter 2, industrial control systems have strict performance requirements that also need to be met by any and all virtual solutions used. Unplanned pauses in the hosts' or virtual machines' operations are unacceptable. Resource allocation, patching and migration processes among other things must be setup in ways that won't interfere with system performance.

Designing and later utilizing a virtual environment in any setting should follow some thought-out systems development life cycle. Careful planning and testing helps ensure not only proper system functionality but also security and compliance with relevant policies once the system gets implemented into actual use. One such life cycle model proposed by the National Institute of Standards and Technology[18] consists of five phases. The phases and their significance in designing virtual environments are as follows:

Initiation During the initiation phase, the need for virtualization is acknowledged. The purpose and benefits of the desired implementation as well as arising operational and business requirements are evaluated and documented. A review on what system components can and should be virtualized is required. A virtualization strategy and policies can be implemented to outline how to effectively make use of virtualization now and in the future.

Development/acquisition During this phase, the technical aspects of the environment as well as the necessary hardware and software components are specified. These components are then purchased or developed.

Implementation/assessment In this phase, the acquired system components are set up and configured to comply with all performance and security requirements. After further assessment and acceptance testing, the system is installed in the live environment.

Operation/maintenance The virtual environment performs as part of the system during this phase. The system needs to be monitored and maintained as well as almost certainly to be supplied with hardware and software updates at some point as ICS life cycles are long. In the case of major updates, the updated system should go through the earlier steps of this development life cycle before getting fielded.

Disposal This phase consists of all activities related to proper termination of the virtual environment while safe guarding vital information. This includes disposing of equipment properly as well as orderly handling of data, whether there's need to transfer it to new systems, archive it, or dispose of it.

6.1 Hardware considerations

As with any system, the design of a virtual environments depends on the resources available. While virtual elements are generally free-to-add (license-wise, computing resources permitting) once you've paid for a virtualization platform license of choice, there's still the need for physical gear running the virtual environment and connecting it to other systems. When designing a virtual environment, it's necessary to verify compatibility of all intended hardware and software. There's some compatibility differences between different versions and features of virtualization platforms even from the same developer. Checking for compatibility might seem obvious but there are several virtualization features enabled or enhanced by specific hardware capabilities. Some hardware vendors provide setups guaranteed to support certain virtual platforms as well as specially tailored hypervisor images with proprietary drivers and default settings for increased compatibility.

As well as confirming compatibility, it's necessary to ensure that the hardware servers as well as the network and storage solutions can accommodate the proposed virtual system. The virtualization platform might require more powerful hardware than the currently used or planned platforms. Moving separate servers to a more centralized virtual environment can also create network bandwidth issues and bottlenecks in access routes to commonly used shared resources can ensue.[37]

Some virtualization platforms can utilize a CPUs hyper-threading abilities to even more efficiently use computing resources. Hyper-threading is a technology that predates modern multi-core processors that enabled running multiple simultaneous threads on a single

CPU. On modern CPUs, it allows for running multiple (usually two) simultaneous threads on single cores. The potential benefit from hyper-threading is based on the assumption that the shared CPU resources aren't constantly in use. In performance critical systems, utilizing hyper-threading should mostly be avoided unless there's absolute certainty that applications being used support it properly to avoid performance degradation. It's important to note that on systems with hyper- threading enabled every CPU core appears as two virtual CPUs to the hypervisor. For example, ESXi would display the first core as CPU 0 and CPU 1, the second core as CPU 2 and CPU 3, and so on.[50]

When using the VMware vSphere Update Manager (VUM), there are some considerations needed to ensure good system performance. Here are some performance related VUM best practices proposed by VMware[50]:

- The VUM and vCenter databases should be separated when there are more than 300 virtual machines or more than 30 hosts being managed by the system.
- The VUM and vCenter servers should additionally be separated when there are more than 1000 virtual machines or more than 100 hosts being managed by the system.
- VUM should be supplied with dedicated physical disks for storage.
- Fast networking with few network hops should be used to connect the VUM system with the managed hosts to keep latency and packet losses down.
- VUM server should be allocated with at least 2 GB of RAM in order for it to be able to cache frequently used patch files.

6.2 Image management

Proper image management can provide many security and operational benefits when setting up and maintaining virtual environments. There are two main types of images to consider when dealing with vSphere platforms: *templates* and *snapshots*. The images can be used to store validated configurations to be used in starting new machines and restore machine states in case they get compromised. These images contain sensitive data about the related machines, and should be properly protected against unauthorized access.

A template is like a master copy of a virtual machine that can be used to instantiate many copies of itself also referred to as *clones*. The foremost reason for using templates is efficiency. Setting up several VMs with similar functionality can be very time-consuming.

Instead, a single virtual machine can be configured to comply to security needs, installed with necessary software, tested and used as a basis for a template. This template can then be used to create further clones without having to repeat the full installation process. Additional changes to clones can then be applied, and further templates created as seen fit. Using templates is also a less error prone method of setting up multiple machines with repetitive installations. However, it's also quite possible to quickly fill a system with poorly configured VMs if the initial template isn't set up correctly.

Whereas templates are used as general molds to create virtual machines with similar configurations, a snapshot is an exact copy of a virtual machine's state at a given point in time that allows the VM's execution to be resumed from that moment of capture. Snapshots are even more security critical than templates, because they contain the full contents or the VM's virtual RAM, and the memory might contain critical information not even present on the virtual hard drive.[37] Even though they can be used to roll back VMs and present some backup capabilities, snapshots shouldn't be thought of or used as actual backups. While the functionality of generating complete snapshots can be available, by default and by design snapshots are only change logs of the original VM, and other solutions should be used for backups. If the virtual disks of a VM get deleted, the snapshots aren't sufficient to recreate the VM. Because snapshots are change logs, they continue to grow in size as time goes by. Therefore, rather than preserving snapshots for extended periods, new snapshots should be regularly created. Further, it's better for system performance to only maintain a few snapshots. If older snapshots need to be stored, they should be moved into a more permanent storage location. In addition to providing rollback possibilities, snapshots can be used to perform forensic analysis on a virtual machine after a detected or suspected breach or other security issue.

A vSphere feature called *Host Profiles* is a baseline tool for keeping several hosts compliant with the configuration of pre-configured and validated *reference host*. This can make keeping an environment with several hosts updated easier by cutting down repetitive manual update tasks when applicable. While not regarded as templates in the VMware vocabulary, custom ESXi images can also be created and used as the mold to instantiate several hosts with. Some hardware vendors provide their own customized hypervisor images, often pre-installed. The images usually include added drivers and patches related to the specific hardware. This can be seen as a security issue if the documentation on these custom images and their detailed contents isn't always readily available.

6.3 Networking

ICS network infrastructure defines how the automation, monitoring, and control components are interconnected and how information travels in the system. There are generally various workable ways to set up any network and the implementation will effect the system's security as well as system performance. As stated in chapter 2, ICS systems have strict performance requirements and historically poor security, while the push is towards more connectivity. Legacy systems might still use plain text communications, and while newer and updated systems are likely to have better security measures, it's good practice to further separate network segments according to their intended use and level of confidentiality for easier management and better security. Networks can be categorized by their intended level of exposure to outside networks as follows[25]:

Open network A network with uncontrolled access to and from WAN. E.g. a home network connected directly to the internet with no special security concerns.

Restricted network A network with strict rules and monitoring regarding connections to outside networks and WAN. E.g. an ICS network behind a demilitarized zone connected to a less security strict and WAN connected office network.

Isolated network A network completely physically and logically separated from other networks. E.g. historical ICS systems with no connections to outside networks.

Truly isolated networks have become very rare. Modern ICS networks belong dominantly in the restricted category. The benefits of data availability across network segments and remote access have outweighed the security gains of total isolation. While the increased connectivity brings greater accessibility by presenting information and interfaces for less restricted networks like the enterprise level LAN and even across open networks, it also makes the system more susceptible to attacks.

Demilitarized zones (DMZ), sometimes referred to as *perimeter networks*, can be used to prevent direct access to the more critical segments of the network. A DMZ is a security buffer zone, a sub-network between firewalls, that mediates traffic between the networks it separates. Figure 6.1 depicts the basic principle of a DMZ setup. There should be no available routes between the separated segments, and ideally there should be no shared protocols between the trusted and untrusted entry-points of the DMZ.[26] All communications from either side of the DMZ are terminated and the DMZ acts as an interface only if the requests conform to preconfigured rules. For example, if there's an authorized request from the untrusted zone, a machine in the DMZ acknowledges it, fetches the data from trusted zone, and passes it back to the untrusted zone.

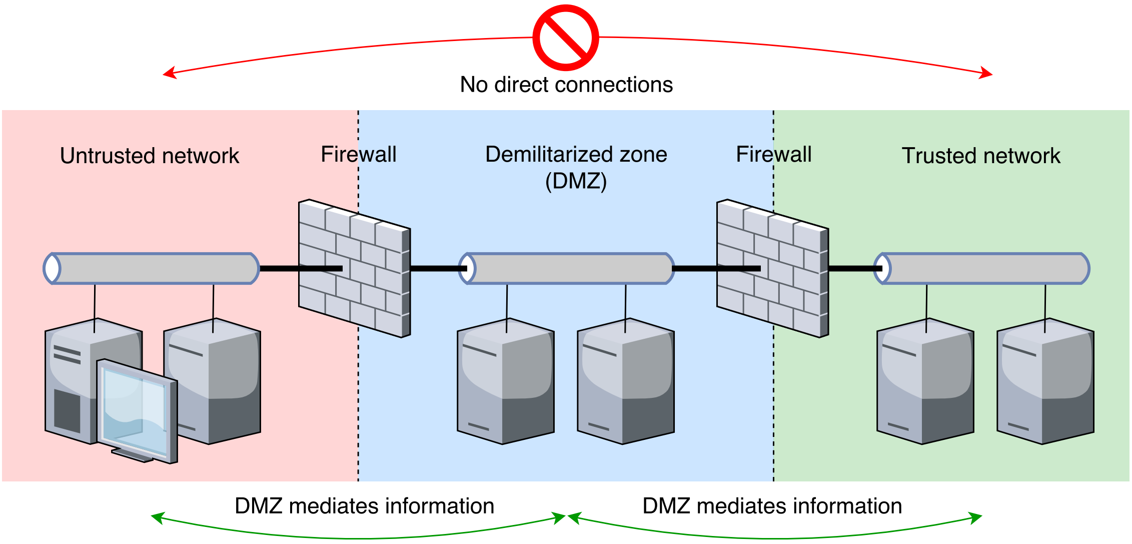


Figure 6.1 *Demilitarized zone principle.*

In the case of an industrial enterprise, the “untrusted” side of the DMZ could be the enterprise network (itself hopefully connected to the internet though a firewall or DMZ setup) and the trusted side would consist of the actual industrial control network. A well-designed DMZ doesn’t affect the performance of the control system and will allow it to continue its operation even if the DMZ gets disconnected or has issues. In fact, a DMZ should be able to quickly severe even the mediated communications if an intrusion or some other notable security anomaly is detected in the enterprise zone.

When dealing with virtualization, the separation of trust zones at its simplest could mean just physically separating the trust zones by keeping them hosted on isolated hypervisors. The hypervisors would run on physical servers separated by a traditional firewall or DMZ setup. This solution has the downside of not really utilizing the potential of virtualization and still requiring the hardware gear to separate the trust zones. Another way to separate trust zones that utilizes virtualization more would be to run the trust zones on a single hypervisor, but to still route the traffic between the two through an external physical firewall. The next logical step would be to use completely virtualized separation of trust zones. Both the trust zones and the now virtual firewall or DMZ would reside on the same hypervisor. The more virtualized the solution gets, the more one can benefit from the general benefits of virtualization: lower hardware costs, less need for physical setup, and more centralized management and monitoring. This could also mean creating a single point of failure situation, however. If the hypervisor gets compromised, the intended security benefits of separating the trust zones could instantly be undone.[39]

Even when planning internal virtual networks, it’s crucial to plan according to available physical components. While adding virtual network components is easy, hardware is likely a more finite resource whether because of cost, space restrictions or limited availability. The amount of physical NICs available on a host is a main consideration when designing internal virtual networks, as the NICs are the interface to external networks for all of the internal traffic. For increased security and performance, different types of network traffic should be separated logically and also physically as much as hardware restrictions allow. Internal virtual network traffic can be categorized into three main types:

Production traffic Alternatively virtual machine traffic. Traffic related to the actual purpose of the system going in and out of the virtual machines. This is likely to make up most of the virtual network's traffic.

Management traffic Traffic consisting of management tasks such as maintenance, changes or updates to the virtual environment by accessing for example ESXi or vCenter.

Operations traffic Traffic related to specialized services such as migration, fault tolerance, and storage.

Ideally, all three main types of traffic should be isolated as management and operations traffic may contain critical system information. For example, during vMotion migration the guest operating system's memory is transferred in plain text over the network. In a poorly planned and secured network an attacker could intercept and gather or modify critical data during the migration process. Older ESX hypervisors offered separate options for creating each type of network, but ESXi has merged management and operations networks under one option. This doesn't mean that management and operations traffic shouldn't still be isolated from each other.

Another aspect of network design possibly limited by hardware is redundancy. In high reliability systems, it's important for at least the most critical data to have backup routes in case of failures. In practice, there aren't necessarily enough physical NICs available to allow for physical isolation and redundancy of all traffic types. Figure 6.2 shows a virtual network connected to an external network though two physical NICs. The traffic types should naturally be kept separate in the external network also. If, for example, the NICs are actually connected to a shared physical switch, separate VLANs should be used. The physical NICs are internally connected to virtual switches, which are then used to access the virtual environment, and, through virtual NICs, the virtual machines. With only two NICs, even the isolation of all traffic types, let alone redundancy, is not possible. In such cases, it's necessary to prioritize system needs and implement accordingly. For example, to achieve redundancy, all traffic could be routed through both NICs without isolation.

The solution pictured has production traffic isolated from management and operations traffic for better security at the cost of reliability. Both of the afore-mentioned options are far from optimal and shouldn't be used in an environment even remotely interested in security or reliability.

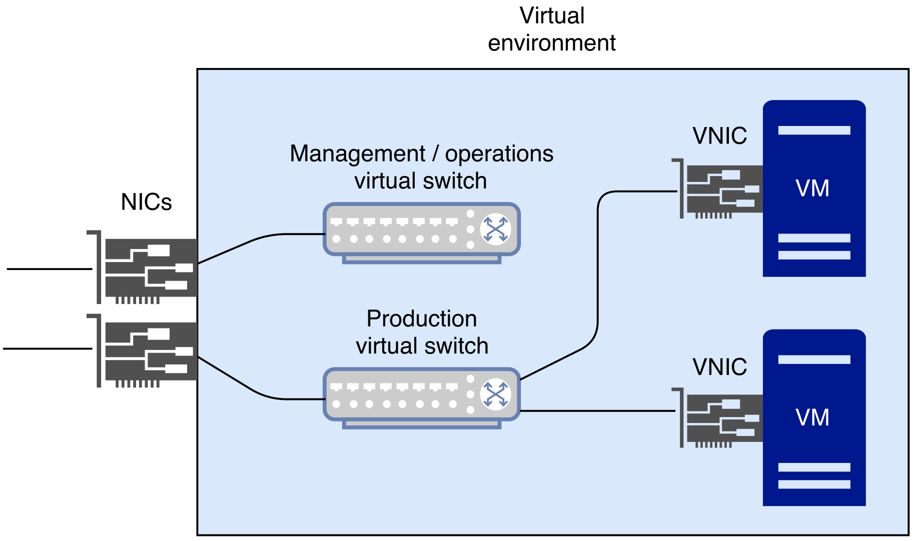

Virtual environment
NICs
Management / operations virtual switch
Production virtual switch
VNIC
VM
VNIC
VM

***Figure* 6.2** *Virtual network setup with two physical network interface controllers.*

Figure 6.3 depicts network topology with six available hardware NICs. This setup makes possible the isolation and redundancy for all traffic types with no compromises to security or reliability. While this type of setup would be ideal, the hardware isn't necessarily available. In practice server blades often have four NICs that could be used for example to separate all three traffic types while providing redundancy for production traffic. This is something that is up to the specific system priorities.

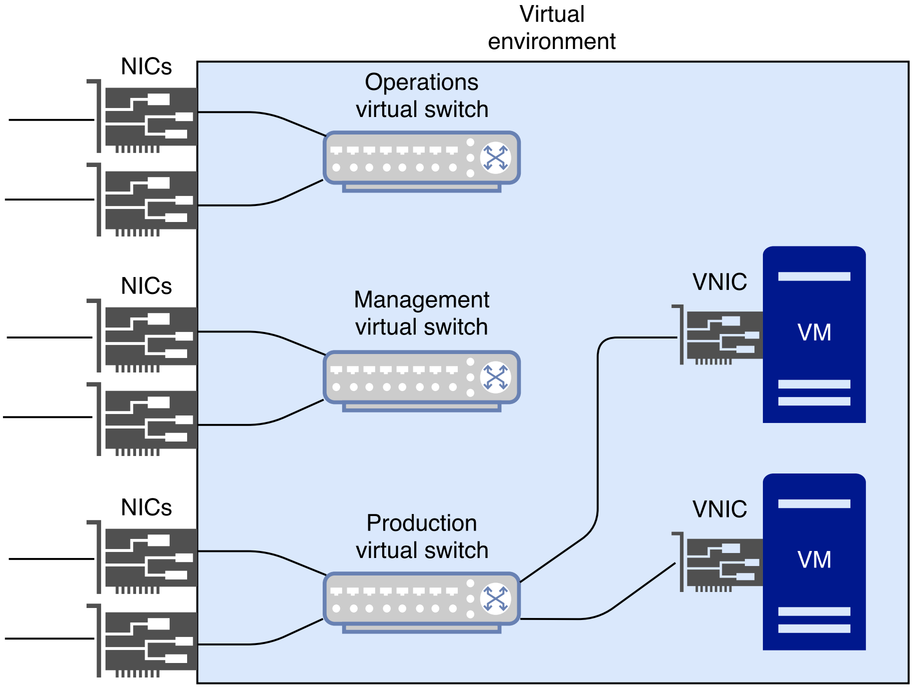

Virtual environment
NICs
Operations virtual switch
NICs
Management virtual switch
VNIC
VM
NICs
Production virtual switch
VNIC
VM

Figure 6.3 *Virtual network setup with six physical network interface controllers.*

6.4 Monitoring and diagnostics

Industrial control systems relay vast amounts of different types of information. *Control information* is used in the actual real-time control loops of the system. *Monitoring and diagnostic information* is a type of information collected by the system and not used directly in control functions, but rather to monitor component health and system performance.[15]

In complex computer systems, there are numerous subsystems and interfaces to gather information from. It would be beneficial to gather necessary data from the different sources and to present it in centralized diagnostics interfaces in a consistent manner to prevent confusion. However, even tools in the same software suite from a single vendor can present system information in a ambiguous manner. For example, there may seem to be discrepancies between resource usage indicators of ESXi, vCenter and the virtual machine's operating system. This is mostly due to current ESXi and vCenter graphical user interfaces defaulting to different metrics of resource usage and not being clear about it. For example, whereas the ESXi reports consumed host memory, vCenter shows the current active guest memory. The consumed host memory is the amount of the host's RAM allocated to the VM. Active guest memory refers to the amount of physical RAM currently in active use by the VM. It's important to be sure of what's being measured especially when presented with vague metrics such as “CPU” or “ memory” without further clarification.

Many of the modern servers from the major hardware providers ship with some type of proprietary management tools, such as Dell iDrac (integrated Dell Remote Access Controller), HP iLO (integrated Lights-Out), and Oracle ILOM (Integrated Lights Out Manager). All of these tools handle out-of-band (OOB) management. OOB refers to having a dedicated channel for hardware monitoring and maintenance. An OOB management channel makes it possible to remotely access equipment even if the machine isn't powered on.

Monitoring network traffic is a crucial part of maintaining a secure and well-performing environment. A monitoring strategy is needed not only to detect and stop attacks, but also to profile the network traffic to reveal potential bottle necks and to ensure and improve system efficiency. Using virtualization, dedicated virtual machines can be used as specialized monitoring devices, such as intrusion detection and sniffing systems.

Port mirroring is one way of forwarding traffic for analysis. It means copying traffic of a port of interest to another port connected to specialized traffic analysis functions. Port mirroring is referred to by different names by different vendors of network hardware. For example, Cisco calls its solution *Switched Port Analyzer* (SPAN), while 3Com's technology is called *Roving Analysis Port* (RAP). In vSphere, distributed switches provide port mirroring functionality to enable both intrahost and interhost traffic monitoring. Port mirroring has a few shortcomings, however. It is susceptible to packet loss and filters out some of the noise and errors in the traffic. This is an issue, as it would be best to be able to forward all of the traffic to the monitoring systems. [25] Another way to monitor network activity is with devices that duplicate traffic on the fly. These devices are so-called *network taps* (sometimes TAP for *test access point*) that create a split copy of the data stream to be analyzed. Unlike port mirroring, where a switch has to actively copy and send the traffic forward, network tapping is a passive process using electrical or optical means of copying the network packets onto a tap port. Taps duplicate all of the traffic passing through them, including erroneous data, which makes them generally more desirable for directing traffic to monitoring systems. There are, however, various types of network taps, including ones with active filtering capabilities. Filtering that actively alters the data flow is where the line between taps and another technology, *network packet brokers* (NPB), starts to get blurry. NPBs are designed to further split or merge traffic coming from mirrored ports or network taps according to header information and forward it accordingly to monitoring systems. Other names for this type of device are *data monitoring switch*, *data access switch*, and *distributed filter tap*. An NPB can be a separate product or a software feature on some advanced switches.[16] Figure 6.4 depicts use of the above-described ways of forwarding traffic to monitoring systems.

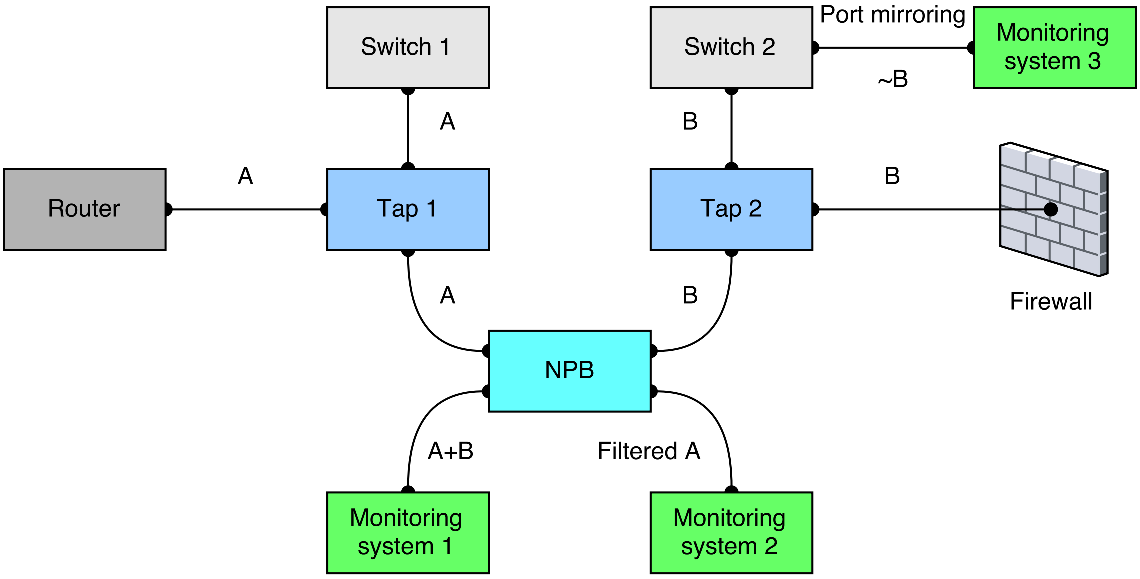


Figure 6.4 *Different methods of forwarding network traffic for analysis.*

The traffic from switch 1 to the router, denoted by A, is duplicated by a network tap and forwarded to an NPB. The NPB further manipulates the traffic and sends an aggregate of A and B (B coming from another network tap) to a monitoring system. The NPB also forwards a filtered portion of A to another monitoring system. The traffic from switch 2 to the monitoring system 3 is indicated by B as a way of saying 'almost B' due to the possible packet losses and filtering effects intrinsic to port mirroring.

VMware has included several functions in vSphere products that either actively monitor or aid in monitoring network traffic. The standard vSwitch includes some basic functions to make monitoring easier, such as the *promiscuous mode*. Promiscuous mode in computer networking refers to an operational mode in which a network adapter passes through all traffic on the same network segment rather than just the packets specifically addressed to it. This allows a connected device to receive and further analyse traffic addressed to other devices on the segment for signs of reconnaissance, exploits, malware, DoS attacks, and other potentially harmful network activity. Promiscuous mode can also be used to diagnose connectivity issues.[45]

7. INDUSTRIAL VIRTUAL ENVIRONMENT SECURITY

Nothing that can be done to better a system's security will completely eliminate security risks. Realistically, the goal is knowing the risks, making the threat levels acceptable, and being prepared to recover from incidents. Managing security often involves having to make compromises between security, productivity, usability and system performance. What constitutes acceptable threat levels and compromises varies greatly by industry, organization and current project. In any case, security is of utmost importance when designing and maintaining modern computer systems. And yet, it can get pushed aside in the design process just to be implemented later on as an afterthought much like an add-on. The causes for this can be inexperience or attempt to keep development costs low. Retrofitting complex systems with security features can be a monumental task that ends up costing a lot more then incorporating security into the design process from the very beginning.[41, 37]

Virtualization decouples the physical and logical states of a hardware machine by adding a hypervisor software layer. This can be seen as additional layer of protection and on the other hand as an additional layer of potential vulnerabilities to exploit. Some of the new security issues might not arise from virtualization per se, but from traditional ICT threats taking new forms or needing special consideration in a virtual environment. [34] All of the security aspects known before-hand as well as those emerging during virtual environment design should be comprehensively documented in the system security plan. This includes all threats and related risk assessments, both planned and implemented security measures, incident scenarios with reaction and recovery plans, and so forth.

This chapter discusses virtual environment security in general as well as special security considerations needed when utilizing virtualization in industrial automation and covers research conducted on the subject. As mentioned in chapter 2, there hasn't yet been large amounts of academic research conducted on virtualization in an industrial setting. As this thesis' focus is on the, narrower still, subject of security considerations related to virtualized ICS, completed research is scant. Virtualization brings an immense amount of new elements and configuration options to the already complex ICS systems. Studying and evaluating them comprehensively is outside the scope of this thesis as there are whole companies and faculties built around figuring out and keeping up with all of the related nuances. What this chapter aims to do is compile the fundamental best practices that can be used as a baseline for securing a virtualized ICS system.

7.1 Former research

In their paper "Security Implications of SCADA ICS Virtualization: Survey and Future"[10] Tiago Crus et al. analyze "the application of virtualization technologies for communications and computing resources in ICS contexts, with a focus on recent developments, open challenges and benefits, from a security and reliability-oriented perspective." They open by acknowledging the ICS paradigm shift from isolation to connectivity and the challenges arising from the increased coupling of ICS and ICT systems and technologies. The use of insecure legacy protocols and system components beyond their end-of-life support in ICS combined with poorly handled new connections to external systems gives rise to serious security threats and incidents. The paper further states that the proven ICT solutions to security problems aren't applicable in the ICS domain; ICS frequently prioritizes availability and reliability over confidentiality and integrity, and regardless of the push towards utilizing generic hardware and open standards, ICS still requires more ad hoc problem solving. ISC/SCADA specific problems, such as update procedures and anomaly detection possibly pausing or hindering system functions, are discussed. Despite the strict ICS requirements, the potential benefits to the security and performance of these systems gained by introducing virtualization technologies are acknowledged: hypervisors' partitioning and isolation features can be used to create managed execution environments, with rollback functionality making recovery and testing easier. SDN and NFV technologies are presented as ways to achieve greater network flexibility and programmability as well as better security due to more efficient centralized monitoring and reaction mechanisms. While adopting ICT-like virtualization techniques is deemed potentially beneficial, it's noted that just as with other solutions that are proven staples in the ICT domain, their introduction to ICS isn't straight-forward. The effects on the ICS must be carefully studied especially when planning to virtualize more real-time strict system components. While real-time virtualization is recognized as problematic, the paper concludes by presenting advancements in real-time hypervisors, deterministic low-latency networks, and network virtualization as a way to realize workable virtual PLCs.

In another paper called "An isolated virtual cluster for SCADA network security research"[3] Lemay et al. propose using a virtualized "ICS sandbox" as a way to perform security experiments. To achieve this sandbox they use an isolated cluster of virtual machines that emulate a SCADA network interfaced with a simulated model of the controlled physical system, in this case a power grid. This paper also starts by acknowledging the advantages and challenges introduced by the merging of ICS and ICT technologies. Stuxnet is mentioned as a turning point that directed considerably more attention to ICS security. An overview on some earlier proposed frameworks for ICS security research is provided. Deploying an actual large-scale physical system to perform the security experiments on is described as the most basic and one of the more realistic approaches. These kinds of systems make possible tests in an environment that very closely resembles an actual live implementation, but setting up, running, and reconfiguring such test systems requires significant capital and operational expenses. Another research setup uses a small-scale representation of a physical system of interest and virtual machines to represent subsystems when applicable. This has the benefit of being less expensive and virtual machines being more easily reconfigurable while still incorporating physical effects. While more flexible, the setup still consists partly of physical components, which restricts configuration options. As the next logical step, a fully software research environment is presented. While extensively and easily configurable, this solution lacks the ability to display, measure, and react to physical effects, and as such can't be used to reliably study systems involving physical aspects. Simulation is then introduced as a way to address this. The major drawback of simulations is developing and validating an accurate enough model of the wanted system. Emulation is presented as a step up from simulation when testing software, as it replicates the function of a system or component precisely instead of just giving similar results. All of these past approaches are then deemed non- ideal with physical deployment being too costly or inflexible, and both simulated and emulated approaches not being able to properly represent the physical properties. The authors aim to find the optimal way to integrate the physical ICS aspect into an existing SCADA infrastructure emulated on virtual machines on servers running "VMware software". While having just stated that "simulations cannot fully capture the interaction between the physical and computer system", they choose to use Hypersim to simulate their physical power flow system. This results in a system consisting of an accurately emulated SCADA system connected to an accurate-enough power flow simulator. The ICS sandbox was then used as a training tool for industry practitioners dealing with different aspects of ICS security, including system administrators and engineers, security experts, policy practitioners, and compliance consultants. By the feedback gathered, the paper comes to the conclusion that such training is valuable and virtualization is a good way to enable such test environments.

7.2 Security benefits and challenges of virtualization

The most fundamental aspect of hardware virtualization is also the core security feature it offers: isolated virtual machines sharing hardware resources. Isolation can help prevent unauthorized use of system resources in case a single VM gets breached. Malicious software can't be injected from a VM to another or onto the host, and denial-of-service attacks can't critically hamper performance as access to resources is limited.[37]

The flexible networking facilitated by virtual networking technologies makes possible several beneficial security solutions. Though most of the methods can be used in a non-virtualized environment, the virtual aspect makes setting up and modifying the network generally easier. The programmable nature of SDN makes it possible to react to detected hostile traffic by dynamically forwarding it to honeypots instead of just blocking it[10]. A honeypot is a decoy for attracting and detecting intrusions. Instead of just blocking on intrusion, a honeypot can emulate or even enable some extent of real interaction with the system. This way more can be found out about the attacker and their intentions than by simply stopping the intrusion as soon as it's detected. In the ICS domain, a honeypot could be set up to simulate the operation of any attractive attack target, such as a network server or a remote terminal unit.[40] Virtualization could also be used to create simulated honeypot environments with interfaces similar to the real system but with none of the real critical information.

Even though virtualization offers potential for many beneficial effects on efficiency and flexibility as well as security of computer systems, it also brings new vulnerabilities into the mix.

Virtual machine sprawl was mentioned earlier in the thesis in chapter 3. The ease of VM creation and virtual network modification can lead to VMs and connections being forgotten. Such unmonitored ghost VMs and connections can lead to security issues. The creation of any unneeded virtual instances should be minimized. Active inventory tracking should be conducted and official rules concerning creation, modification, and removal of items in the virtual environment defined.

Perhaps the most serious threat to a virtual environment is a guest to host breach also known as *virtual machine escape*. In a successful breach, an attacker is able to break out of a single virtual machine and into the underlying host system thus gaining possible access to other virtual machines and management options or even other networked subsystems. For some time, this was only a theoretical threat until 2015 when the so- called Venom vulnerability was discovered by a cybersecurity company CrowdStrike [9]. The vulnerability was caused by a long-standing bug in QEMU's (Quick Emulator, an open-source hypervisor) virtual floppy disk controller, which went undetected for seven years. The compromising controller code made VM escape possible and was used in many virtualization applications such as KVM, VirtualBox and Xen. In an ICS environment such a breach could lead to severe consequences. However, VM escape is a bigger threat in public cloud services that can house VMs of numerous actors. In the ICS domain it's principally less likely as third parties shouldn't have any direct access to the VMs dealing with critical information.

7.3 Hardening

Modern software is often complicated. Desired software products may by default include features not necessary for the main functions of the system being designed. In security and performance critical environments such nonessential features should be viewed as liabilities. Hardening is the act removing or disabling all unneeded features and leaving only the necessary functionality as a way to minimize a system's attack surface and the amount of points of failure. In addition to removing or disabling unused or vulnerable software components, hardening should also cover changing default passwords, replacing possible place-holder certificates, reviewing user account privileges and removing unneeded logins altogether, as well as restricting network connections to a necessary minimum. Virtualization platforms are no exception to most software in that they aren't inherently secure either. Some of the default settings are recommended to be changed in almost all use cases even by the platform providers (see for example vSphere 6.5 Security Configuration Guide [47]). Virtualization software providers and several third parties offer basic guidelines detailing how to configure a system for enhanced security. A good example of such guidelines are Security Technical Implementation Guides (STIG)[14]. STIGs are guidelines designed to help make various types of cyber systems more secure and to help enterprises conform to compliance requirements of their fields. The hardening requirements and restrictions of systems vary greatly, and while the public guidelines can be consulted to get started, thorough case specific assessment and testing has to be conducted.

Hardening can lead to compromises between security and usability. Removing some features might provide increases in security while negatively impacting work morale and productivity. Security consists of intrinsically inconvenient measures and these piling obstructions can lead to users circumventing the intended security features and recommendations to improve their workflow, creating possibly unpredictable security problems in the process.[4] Though outside the scope of this thesis, these kinds of non-technical consequences and user experience in general should not be over-looked in the security design and hardening processes.

7.4 User roles and separation of duties

Any functionality of an ICS should only be accessed by users who work in a related sector and are qualified and authorized to use the specific functionality. In addition to reducing the amount of power of single individuals and defining permissions, this clarifies responsibilities and reduces conflict of interest situations when managing and operating the system. Such separation of rights and duties also makes it easier to analyse security problems and breaches by clarifying accountability. User roles define the tasks that an associated user is allowed to carry out. They are used to restrict access to system information and features that all users don't need to access. Several users can be assigned to the same user role.

Following a password policy should be a well-established practice in any enterprise dealing with computer systems, and user names and passwords created in a virtual environment should also follow the current policy. Password rotation can be ensured by setting expiration dates for all passwords. Expiration intervals for less critical user roles shouldn't be too short, however, as frequent password changes can lead to choosing weaker passwords, and the security benefits of password expiration policies in general haven't been properly quantified[7, 54].

Besides in the technical context of controlling security and privileges, user roles can be understood as defining and separating personnel responsibilities. In a traditional non-virtual setting, the management of subsystems can be separated between administrator teams all with their distinct responsibilities. Such responsibilities could include tasks related to networking, storage, Unix, and Windows systems. While in practice there will be some overlap and conflicts of interest, each team working on a clearly confined area of expertise makes division or work and accountability straight-forward. In a virtualized environment, defining these kinds of clear-scoped management duties and boundaries can get complicated. For example, is the network team or the virtualization team responsible for the internal virtual network? A problem with the virtual network might need the know-how of both teams. These blurry lines of responsibilities need to be addressed. A solution to handling the emerging compound issues could be assembling new specialized teams with enough internal knowledge or establishing adequate inter-team collaboration between existing specialized admin groups.

7.5 Hypervisor security

Thorough planning and maintaining of hypervisor security is of critical importance to the overall virtual environment security. The hypervisor acts as the interface to the hardware on the physical host computer. It's the core tool that allows the user to create and modify virtual machines as well as perform other managerial operations. Therefore the security of the virtual system as a whole relies on keeping hypervisors, and higher level management systems such as vCenter, secured. Access to these systems should be restricted to authorized administrators only and system activity should be actively monitored. This section focuses on securing hypervisors, and with this thesis' main focus on VMware products, the more specific recommendations will be related to the their type 1 hypervisor ESXi. There are several local and remote management interfaces related to ESXi (and other hypervisors), and it's essential to make sure that none of these interfaces gets overlooked.

In his book *Virtualization Security: Protecting Virtualized Environments*[39] Shackleford presents the following as the main areas of concern in general hypervisor security:

Patching Keeping hypervisors updated should be considered a fundamental practice in maintaining a secure virtual environment.

Establishing secure communications Host machines are regularly networked with remote clients, management platforms and other subsystems. Securing and validating these connections before live operation is essential.

Changing default settings As mentioned in the hardening section, hypervisors don't ship with secure settings, and some of the features can be removed outright.

Enabling operational security Configuring the virtual environment for consistent and accurate operational tasks such as monitoring and logging is needed to assure reliable and verifiable operation.

Securing and monitoring critical configuration files Special attention should be paid to the security and monitoring of the critical files associated with the configuration and control of the virtual environment.

Securing users and groups Reviewing and limiting the amount of user roles and their privileges will make the system less suspect to harmful actions, whether intentional on unintentional.

Locking down access to the hypervisor platform Hypervisors can be accessed through various local and remote interfaces. All access channels to the hypervisor should be carefully set up and monitored.

While the above considerations (and the book in general) are aimed at virtualization in the ICT domain, they hold true in the ICS domain as well. When designing and maintaining an ICS implementation, you just need to pay some additional attention to things like availability and real-time performance if virtualizing components with critical requirements. We'll now use the above list as foundation and go through security best practices related to each area.

7.5.1 Patching ESXi

New vulnerabilities in virtualization software are discovered and new safety measures implemented regularly[13][51]. It is crucial to stay informed on recent developments and keep the system updated. As mentioned, ICSs often strive for close to a 100% availability, and as such caution in performing any system updates should be exercised. The update process itself or an unstable patch can't be allowed to adversely effect system function. Virtual platforms offer some possible ways to make the update process more flexible and reliable, namely live migration and virtual test environments for testing the patches before committing to updating a live physical system.

It's generally a good idea to install the latest updates released by your virtualization software vendor of choice, and there's various ways to go about updating ESXi. Perhaps the most prominent tool is the VMware's purpose- built patch management framework vSphere Update Manager (VUM). It's designed for automating and simplifying the updating of ESXi systems. VUM can make use of clusters and live migration to enable continuous execution of virtual machines while a host is getting patched. VMs are automatically relocated to another host on the cluster given that there are enough free resources to accommodate them. VUM allows the user to configure where to fetch the updates and how to apply them. For security critical systems of the ICS domain it's best practice to use local repositories and strict control over when and what to patch. This makes it possible to thoroughly test new updates for possible overlooked vulnerabilities and effects on system performance before applying them to the live production environment. These local repositories can be set up in vCenter as “shared repositories”. A tool called Update Manager Download Services can be used to download the updates and create the shared repository accessible to local vCenter systems. The shared repository can be hosted on a web server or a local disk. In performance critical environments, the service could be scheduled to regularly download the updates to a repository in a test laboratory. After testing, the updates would then be manually (perhaps even physically) moved to a production repository.

The production environment can make further use of VUM's features. Baseline profiles can be used to control host-by-host priority of different types of updates. For example, less critical hosts and virtual appliances could be allowed to be updated with important patches automatically. For the more critical hosts it's recommended to apply every update manually.[39] VUM also has a setting for automatically generating temporary or permanent snapshots of virtual machines before applying updates to them. Making backups before modifying a live control system is advised, and this feature can help automate the process. VUM also supports so-called *udpate staging*. Staging allows the hosts to download updates from VUM beforehand without actually applying them. This will speed up the installation process once it's scheduled or manually initiated.

7.5.2 Securing ESXi communications

By default, communications with the ESXi management interface are encrypted using self-signed SSL or TLS certificates.[49] These certificates are created by *VMware Certificate Authority* (VMCA) during ESXi installation. While they enable the initial encrypted communication and aren't necessarily insecure, they should ideally be replaced by certificates signed by trusted certificate authorities (CA), whether third-party or internal. VMCA initially uses a self-signed root certificate to sign further certificates it creates. The root certificate can be replaced by one signed by a trusted CA. This approach combines the security of a trusted root certificate with VMCA's automated certificate management features.

Securing ESXi communications is also possible by using *Internet Protocol Security* (IPsec). However, ESXi hosts only support IPsec for IPv6 traffic.

7.5.3 Changing ESXi's default settings

As pointed out earlier in this chapter's section about hardening, the default settings of any given software shouldn't be assumed secure. This holds true for VMware products also. As the hypervisor is the main orchestrator of virtualization and the channel between hardware and the virtual machines it creates, it should be made as secure as possible. Many of ESXi's default settings are not secure and should be changed. Some of the default features, accounts, or whole components are likely unnecessary for the main function of the system in use and should be disabled. Reconfiguring these settings and available features to meet the actual needs of the current system (and nothing more) is a main part of ESXi hardening. What the actual needs are, however, varies case by case and there's no catch-all solution. VMware and third security and hardening guidelines can and should be consulted for basic best practices, but a more extensive review on what needs to and can be changed is always necessary when dealing with security critical systems.

7.5.4 ESXi operational security

The operation of virtual environments is upheld and made sustainable by several ancillary services, protocols, and procedures, such as logging, auditing, time services, and monitoring systems. ESXi needs to be configured to enable these services. Two prominent examples of these support functions are the *Network Time Protocol* (NTP) and Simple Network Management Protocol (SNMP). NTP is a protocol for clock synchronization between separate computer systems. It makes reliable logging possible. Without proper clock synchronization, time stamps between logs of different subsystems won't match up, and correlating events becomes difficult if not impossible. NTP is one of the oldest internet protocols still in use. As with many legacy protocols, NTP has some security issues.[45] With industrial control systems, using a local NTP server instead of a public one is a given. SNMP is a network management protocol that can be used to collect information from network devices. Similar to NTP, SNMP is an old protocol and suffers from security vulnerabilities. The newer SNMP version 3 provides improvements on security by introducing user name and password authentication as well as encryption of traffic. The ESXi SNMP agent can use SNMP version 3. Older versions of SNMP use so-called *community strings* transferred in plain text to identify devices, and the default strings on many SNMP implementations are commonly known. When having to use older SNMP versions, the community strings need to be changed as soon as possible.

7.5.5 Securing ESXi configuration files

ESXi hosts include numerous files that can be considered critical from a security standpoint. These files contain sensitive information regarding configuration specifics, enabled services, system events, login information, and so on. The security of these files should be carefully planned and monitored, but the same approach won't work with all of them due to differences in the files' natures; some are static and should stay the same over time unless major system changes are performed, others dynamic with constantly changing content and size. [39] As mentioned, these files are numerous and their contents and importance can vary by time. The actual importance of each of the files and appropriate security measures should be established case by case. Some generally important files can be found by accessing https://[host address]/host as well as in /etc/ and /var/log/ on the host machine. The more static files' integrity can be monitored by performing hash checks. On ESXi systems, *sha1sum* hashing program can be used to create and verify SHA-1 hashes for files. However, SHA-1 can't be considered unbreakable anymore[45], and a more robust modern hash function is preferable in critical applications.

7.5.6 Managing ESXi user roles

As discussed earlier in a more general sense, also the amount of user roles specific to ESXi should be kept to a minimum and the privileges of the remaining roles should be reviewed and restricted to reduce the risk of oversights in access to administrative ESXi functions. It's also important to ensure that the passwords meet the requirements of an approved password policy. Two primary ways to define and authenticate users able to access ESXi management interfaces exist. Authorized local accounts can be created directly on the host machine. These accounts are exclusive to the host and can't be used to access other hosts in the environment. The other way is using a centralized authentication directory, such as Microsoft *Active Directory* (AD). Using this method, login information isn't stored locally. The host checks user names and passwords against the central AD store, and only users belonging to an authorized AD group are given access to the ESXi. However, using AD isn't unproblematic either, as there's the issue of where the AD is located and how to access it securely if you want to use a centralized shared enterprise AD.

There are some default ESXi user roles that are necessary. The most significant one is called 'root'. As traditional with computer systems, the *root user* is a superuser with full administrative access to all functionality and files of the system. The root account is used during the ESXi installation process as the initial access point to the host. The administrators can use the root account to manage the system after the installation, but this isn't recommended and should be avoided if not absolutely necessary. Whether using local or centrally managed accounts, individual named accounts should be setup for everyone with administrative access to the hypervisor, so all activity on the host can always be associated with a specific account. As root access to the host should be kept to a minimum, it can also be restricted to prevent access. This can be done by enabling *lockdown mode*, which restricts all non-vCenter remote access to the host. Remote SSH root logins can also be restricted by editing the /etc/ssh/sshd_config file's parameter 'PermitRootLogin' to have a value of 'no'[39]. Another important default user is called 'vpxuser'. This user is generated by vCenter on every ESXi host that's connected to it. It's a privileged account acting as a proxy for all of the actions initiated in vCenter. The 'dcui' user is used to access the Direct Console User Interface and 'daemon' is a noninteractive account used by ESXi services. All of the above default users are (almost always) needed. There's also a less necessary default user 'nfsnobody' that can be used in configuring Network File System storage.

7.5.7 Locking down access to the ESXi platform

Access to both the local and remote management interfaces must be restricted. Management traffic should travel encrypted through isolated channels separate from production and operations traffic. Basic network topology and network hardware requirements to achieve this isolation were discussed earlier in chapter 6. ESXi also ships with a built-in firewall that can be used to restrict access to the management interface.

The ESXi shell and SSH services are disabled by default, and only users assigned to the 'localadmin' role are enable to access the Direct Console User Interface used to configure the hypervisor. If the shell needs to be accessed locally, or remotely by SSH, the services can be enabled with timeouts to reduce the risk of forgetting to disable shell access again afterward. When using vCenter to manage a system of multiple ESXi hosts, the individual ESXi web interfaces should also be disabled.

Using the basic command line tools to keep ESXi systems updated is viable in smaller environments. By using the APIs and SDKs presented in section 5.2, more advanced scripts or even larger-scale custom applications can be developed to better fit the needs of a particular system. Doing so instead of using VUM is often likely to be just re- inventing the wheel, but it's up to case-by-case deliberation.

7.6 Virtual machine security

As described in chapter 3, virtual machines access hardware through a virtual representation provided by the hypervisor. VMs aren't meant to be able to access the hypervisor itself as direct access to the hypervisor means direct and administrative access to the actual CPU, memory, storage, and network components. If a hypervisor gets compromised, the attacker is likely able to access and modify the hosted virtual machines as well as plant malicious content on the hypervisor itself with possible implications on the ICS environment as a whole. One possible way of reaching the hypervisor is by managing to break out of a virtual machine. While the VMs' contents in ICS are likely critical in their own right, this is why securing them is essential for the security of the system in general.

Even though VMs are designed to appear and behave almost exactly like their physical counterparts, there's some major differences to consider when securing virtual systems. The most distinct difference is that VMs consist of files (see subsection 3.1.2 for a detailed description). Compared to physical machines, files can be more easily altered, corrupted, copied or scanned even without a trace in case of improper security measures. As Shackleford put it: "People will notice you walking out of the building with a server but not with a USB stick containing a set of VM files."[39] Protecting these files by restricting access and encrypting them should be a high priority. This includes any stored snapshots and backups. Encryption can take place in many places when it comes to protecting VM data. Encryption inside the VM itself only protects the guest operating system's disk files just as with physical servers, and does not protect the files that make up the VM. VMware provides VM encryption within their ESXi hypervisor since version 6.5. This hypervisor side encryption protects both the core VM files and the disk files regardless of the used guest OS or datastore type. Encryption can also be done within the data storage solution of choice. These choices provide different levels of protection for the different VM files and (as has become the motto of this thesis) their benefits and restrictions need to be evaluated to fit the specific system needs. More detailed suggestions and considerations for securing VMware VMs can be found in VMware's official security guidelines, third-party STIGs and in Shackleford's *Virtualization Security: Protecting Virtualized Environments*[39].

The operating systems being run in VMs should be secured according to system needs just as when dealing with physical servers. This thesis won't address how to do this as the focus is on the special considerations needed in securing the virtual environment. However, securing the OS can't be overlooked and OS specific security guidelines should be consulted when designing or maintaining secure systems.

Although snapshots can act as effective fail-safes against virtual machines getting compromised or corrupted, some organizations have policies against storing snapshots. This is due to the snapshots potentially encompassing malware from earlier infected systems. Reloading an infested snapshot would introduce the malware into the system once more.[37] A snapshot could also have been taken of a VM using now-obsolete security policies and reverting to such setting could bring back disabled accounts and passwords not currently accounted for. Not using snapshots at all is an extreme measure as snapshots can provide significant benefits as a rollback tool and in system analysis, but the reasoning is something to consider when planning system security and defining an image management strategy.

8. PRACTICAL HARDENING APPLICATION

As discussed earlier in this thesis, virtualization can bring very worth-while benefits when utilized correctly in the ICS field. Introducing virtualization to the mix also comes with new problems to solve. While the workload of managing several physical machines and networks decreases, the new workload of managing the virtual counterparts and the virtual platform itself is introduced. If some of the operational workload related to maintaining and monitoring the virtual system can be automated, the trade-off becomes more favourable. This chapter presents a way to simplify the security compliance checking and reconfiguration of virtual machines in a vSphere environment.

To test the applicability of the interfaces and scripting tool presented in chapter 5 in practice, a simple script for checking virtual machine configurations against hardening guidelines was developed. The guideline the VM configurations are compared to is a modified and updated version of VMware's 6.0 hardening guide. An additional script for configuring the virtual machines to comply with said hardening guidelines was also developed. These proof-of-concept scripts deal only with VM settings for the simple reason that they're easiest to access and modify. Extending these scripts to check and change for example hypervisor and network configurations is fairly straight-forward but more time consuming as the configuration parameters aren't as uniformly located or configurable.

The chosen approach was to use pyVmomi to access the virtual machines through vCenter or directly through ESXi hosts. The script can be used locally or remotely on any machine with network access to the desired host or vCenter. The script then accesses the primary .vmx configuration file of all the virtual machines residing on the host or hosts managed by vCenter. The settings in these configuration files are compared to those found in the hardening guideline. The guideline is stored in a file as comma-separated values (CSV). CSV is a widely supported and simple format for storing tabular data. The data in a CSV file can be displayed by most major office software and interpreted by most major programming languages. This makes it a flexible and accessible format that can be viewed and further manipulated on nearly all platforms.

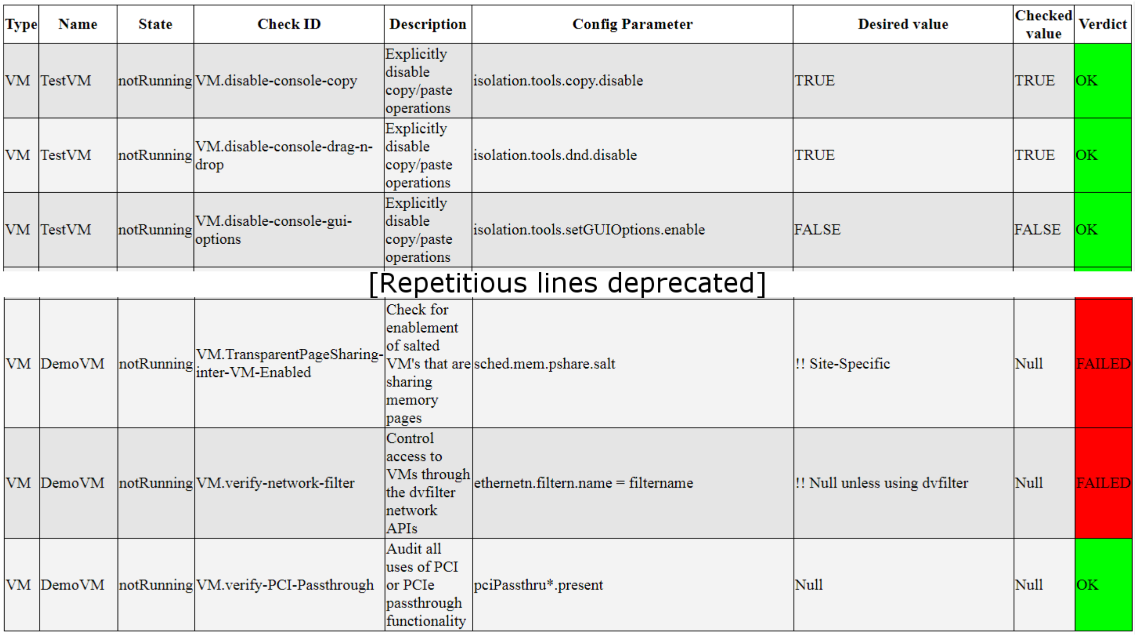
Terminal output of an example hardening check can be seen below.

```
Host or vCenter address: 192.168.123.123
User: admin
Enter password:
Port (blank for 443):

Ran check 1/120 on TestVM: VM.disable-console-copy OK
Ran check 2/120 on TestVM: VM.disable-console-drag-n-drop OK
Ran check 3/120 on TestVM: VM.disable-console-gui-options OK
Ran check 4/120 on TestVM: VM.disable-console-paste OK
[Repetitious lines deprecated]
Ran check 117/120 on DemoVM: VM.restrict-host-info FAILED
Ran check 118/120 on DemoVM: VM.TransparentPageSharing-inter-VM-Enabled FAILED
Ran check 119/120 on DemoVM: VM.verify-network-filter FAILED
Ran check 120/120 on DemoVM: VM.verify-PCI-Passthrough OK

Security check complete. Results in report.csv and report.html.
```

The script asks for a server address and user credentials (the password input is hidden) and goes on to scan all the virtual machines on the host or all of the hosts available if a vCenter server is accessed. The results of the security check are printed in a CSV file. The script also exports an HTML file formatted for easier interpretation. Figure 8.1 shows an excerpt of the generated HTML page.



Type	Name	State	Check ID	Description	Config Parameter	Desired value	Checked value	Verdict
VM	TestVM	notRunning	VM.disable-console-copy	Explicitly disable copy/paste operations	isolation.tools.copy.disable	TRUE	TRUE	OK
VM	TestVM	notRunning	VM.disable-console-drag-n-drop	Explicitly disable copy/paste operations	isolation.tools.dnd.disable	TRUE	TRUE	OK
VM	TestVM	notRunning	VM.disable-console-gui-options	Explicitly disable copy/paste operations	isolation.tools.setGUIOptions.enable	FALSE	FALSE	OK

[Repetitious lines deprecated]

VM	DemoVM	notRunning	VM.TransparentPageSharing-inter-VM-Enabled	Check for enablement of salted VM's that are sharing memory pages	sched.mem.pshare.salt	!! Site-Specific	Null	FAILED
VM	DemoVM	notRunning	VM.verify-network-filter	Control access to VMs through the dvfilter network APIs	ethernetn.filtern.name = filtername	!! Null unless using dvfilter	Null	FAILED
VM	DemoVM	notRunning	VM.verify-PCI-Passthrough	Audit all uses of PCI or PCIe passthrough functionality	pciPassthru*.present	Null	Null	OK

Figure 8.1 Parts of the HTML page generated by the security check script.

Some of the configuration parameters' desired values are marked as site-specific and are yet to be specified. When applying such scripts in actual ICS environments, desired values for each parameter should be reviewed and changed as needed to create a hardening profile that won't hinder or prevent necessary system functions.

The hardening check script was tested in a small virtual environment set up specifically for this thesis as well as in a larger development and testing laboratory of a major ICS provider.

The additional functionality that actually performs the configuration changes instead of just checking compliance was also achieved by using pyVmomi and the same CSV guideline used by the checking script. The script is used to contact an ESXi or vCenter server and then select the VMs to be reconfigured according to the hardening guideline.

In addition to being useful security tools, these simple scripts were developed to test the applicability of the readily available APIs and SDKs provided by VMware for automating administrative tasks in a virtual environment. The tools and interfaces provided by VMware are very comprehensive and make it possible to develop scripts and applications with more specialized and automated ad hoc features compared to the VMware's default GUIs and command line tools for host and cluster management.

9. CONCLUSIONS

As ICS solutions are mostly very complex and utilize many obscure technologies, the integration of commercial virtualization platforms designed mainly for information and communications technology systems can be challenging. This means that the capabilities of the platforms and the special requirements of ICS need to be understood. Some ICS components have very strict real-time requirements, and virtualization has long been considered incompatible with the most latency-sensitive applications. There are, however, virtualization solutions capable of reaching consistent millisecond scale cycle times. Even the more commercial ICT focused platforms can be used to virtualize less real-time strict higher level ICS components such as SCADA.

The hardening and general security of virtualization when used as part of ICS is a multifaceted subject with few catch-all solutions that can be directly applied to all virtualized ICS systems. This thesis presents a review on some of the challenges and benefits, as well as generic best practices, related to designing and managing virtual environments in the industrial control system context. Virtual environments can be set up in numerous ways, and the structure of the system can make a significant difference on how much it can be hardened and how secure the final environment is. Security should be included as a integral part of the design process from the very beginning of the development lifecycle. Security shouldn't be viewed as a necessary evil that obstructs development and work flow, but as an important tool that allows the system to achieve and maintain it's main goals of high availability and reliability. Virtualization is a boon for security, with advanced high availability and fault tolerance features, virtual machine snapshots for rollbacks, hypervisors as an additional layer of security and so on. From a hardening point of view, adding virtualization is also adding another layer of complexity and potential vulnerabilities into the already complex mix that is ICS.

The decision on whether to use virtualization in ICS or not might not be all that self-evident. While the amount of work related to managing physical servers and networks decreases with virtualization, the operational workload of managing hypervisors and virtual machines is introduced. To skew the decision in virtualization's favor, the possibilities of automating some of the administrative tasks were studied. The chosen virtualization vendor VMware provides comprehensive APIs, SDKs, and scripting tools to access their virtualization platform. The proof-of-concept solution developed is a set of Python scripts that can remotely assess and harden all virtual machines on a hypervisor or a group of hypervisors centrally managed by VMware's vCenter. Being able to create such tools that automate and speed up the management and monitoring of virtual environments are very valuable. This is not only because managing virtual environments consists of many time consuming repetitive tasks some of which the off-the-shelf user interfaces can't handle efficiently, but also because the specific needs of ICS setups vary greatly and require many specially tailored functions. Automating such tasks also reduces human error.

9.1 Future work

The subject of hardening and general security of virtualization solutions when applied in industrial automation is vast, and this thesis only scratches the surface by presenting some basic best practices and what to take into account. All of this gathered information on challenges and benefits is still just a small part of the virtual ICS puzzle; the suggestions in this thesis, while useful and practical, can only be used as a starting point. Actually designing and integrating a practical and secure virtualization solution in an actual ICS environment is something that requires large amounts of hands-on work beyond an academic review on general best practices. There's a plethora of things such as policies, practices, protocols, connections, ways to represent data, personnel responsibilities and resources both internal and external from a company's perspective to consider.

Follow-up work could include more practical work related to specific individual problems related to actual live implementation of ICS virtualization. Or perhaps more in-depth academic work on some sections of this thesis.

REFERENCES

[1] M. Abrams and J. Weiss, “Malicious control system cyber security attack case study - maroochy water services, australia,” 2008. [Online]. Available: http://csrc.nist.gov/groups/SMA/fisma/ics/documents/Maroochy-Water-Services-Case-Study_report.pdf

[2] P. Ahonen *et al.*, *KYBER-TEO — tuloksia 2014–2016*. Teknologian tutkimuskeskus VTT Oy, 2017. [Online]. Available: http://www.vtt.fi/julkaisut

[3] L. Antoine, J. Fernandez, and S. Knight, “An isolated virtual cluster for scada network security research,” in *Proceedings of the 1st International Symposium for ICS SCADA Cyber Security Research 2013*, 2013.

[4] S. Bacik, “Productivity vs. security,” in *Information Security Management Handbook, Sixth Edition, Volume 5*, H. F. Tipton and M. K. Nozaki, Eds. Auerbach Publications, 2012, ch. 28.

[5] D. U. Case, “Analysis of the cyber attack on the ukrainian power grid,” 2016.

[6] L. Chen, S. Patel, H. Shen, and Z. Zhou, “Profiling and understanding virtualization overhead in cloud,” in *2015 44th International Conference on Parallel Processing*, Sept 2015, pp. 31–40.

[7] S. Chiasson and P. C. Oorschot, “Quantifying the security advantage of password expiration policies,” *Des. Codes Cryptography*, vol. 77, no. 2-3, pp. 401–408, Dec. 2015. [Online]. Available: http://dx.doi.org/10.1007/s10623-015-0071-9

[8] I. Corporation, “IntelÂ® 64 and ia-32 architectures software developer’s manual volume 1: Basic architecture,” Tech. Rep., 2017.

[9] CrowdStrike. (2015) Venom - virtualized environment neglected operations manipulation. [Online]. Available: http://http://venom.crowdstrike.com/

[10] T. Cruz, R. Queiroz, P. Simoes, and E. Monteiro, “Security implications of scada ics virtualization: Survey and future trends.” Reading: Academic Conferences International Limited, 07 2016, pp. 74–83.

[11] R. Dittner and D. Rule, *The Best Damn Server Virtualization Book Period.* Elsevier Science, 2007.

[12] N. Feamster, J. Rexford, and E. Zegura, “The road to sdn: An intellectual history of programmable networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, Apr. 2014. [Online]. Available: http://doi.acm.org/10.1145/2602204.2602219

[13] L. Foundation. (2017) Xen security advisories. [Online]. Available: https://xenbits.xen.org/xsa/

[14] U. C. Framework. Complete stig list. [Online]. Available: https://www.stigviewer.com/stigs

[15] B. Galloway and G. P. Hancke, “Introduction to industrial control networks,” *IEEE Communications Surveys Tutorials*, vol. 15, no. 2, pp. 860–880, Second 2013.

[16] I. Gartner. (2016) Market guide for network packet brokers. [Online]. Available: https://www.gartner.com/doc/reprints?id=1-2WO9JPT&ct=160120&st=sb

[17] E. T. S. Institute, “Network functions virtualisation (nfv); architectural framework,” Tech. Rep., 2013.

[18] R. Kissel, K. M. Stine, M. A. Scholl, H. Rossman, J. Fahlsing, and J. Gulick, “Sp 800-64 rev. 2. security considerations in the system development life cycle,” Gaithersburg, MD, United States, Tech. Rep., 2008.

[19] E. Kotsovinos, “Virtualization: blessing or curse?” *Queue*, vol. 8, no. 11, p. 40, 2010.

[20] A. Lappalainen, “Virtualization applicability to industrial automation,” Master’s thesis, Tampere University of Technology, 2016.

[21] M. Lehto and P. NeittaanmÃki, *Cyber Security: Analytics, Technology and Automation*, 2015.

[22] M. Leinonen, “Layer 2 ethernet communication tunneling possibilities in automation systems,” Master’s thesis, Tampere University of Technology, 2017.

[23] T. Macaulay, *Cybersecurity for Industrial Control Systems: SCADA, DCS, PLC, HMI, and SIS*, 1st ed. Boston, MA, USA: Auerbach Publications, 2012.

[24] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright. (2014) Virtual extensible local area network (vxlan): A framework for overlaying virtualized layer 2 networks over layer 3 networks. [Online]. Available: https://tools.ietf.org/html/rfc7348

[25] M. Mantere, "Network security monitoring and anomaly detection in industrial control system networks," Ph.D. dissertation, University of Oulu Graduate School, 2015.

[26] D. C. Mazur, R. A. Entzminger, P. A. Morell, J. A. Kay, and E. Syme, "Defining the industrial demilitarized zone and its benefits for mining applications," *IEEE Transactions on Industry Applications*, vol. 52, no. 3, pp. 2731–2736, May 2016.

[27] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[28] H. Meyer, *Manufacturing Execution Systems: Optimal Design, Planning, and Deployment*. The McGraw-Hill Companies, Inc., 2009.

[29] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.

[30] I. S. of Automation, "Isa-s95-1: Enterprise-control system integration part 1: Models and terminology," 2000.

[31] ——, "Isa-99.00.01: Security for industrial automation and control systems," 2007.

[32] I. of Electrical and E. Engineers, "802.1q-2014 - ieee standard for local and metropolitan area networks–bridges and bridged networks," Tech. Rep., 2014.

[33] F. office for Information Security, "Industrial control system security - top 10 threats and countermeasures 2016," Tech. Rep., 2016. [Online]. Available: https://www.allianz-fuer-cybersicherheit.de/ACS/DE/_/downloads/BSI-CS_005E.html

[34] M. Pearce, S. Zeadally, and R. Hunt, "Virtualization: Issues, security threats, and solutions." *ACM Comput. Surv.*, vol. 45, no. 2, p. 17, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/csur/csur45.html#PearceZH13

[35] R. Radvanovsky and J. Brodsky, *Handbook of SCADA/Control Systems Security*. Auerbach Publications, 2013.

[36] A. Rege-Patwardhan, "Cybercrimes against critical infrastructures: a study of online criminal organization and techniques," *Criminal Justice Studies*, vol. 22, no. 3, pp. 261–271, 2009.

[37] K. A. Scarfone, M. P. Souppaya, and P. Hoffman, "Sp 800-125. guide to security for full virtualization technologies," Gaithersburg, MD, United States, Tech. Rep., 2011.

[38] D. Schulz, "Fdi and the industrial internet of things," in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, Sept 2015, pp. 1–8.

[39] D. Shackleford, *Virtualization Security: Protecting Virtualized Environments*, 1st ed. Alameda, CA, USA: SYBEX Inc., 2012.

[40] P. Simoes, T. Cruz, J. Proenca, and E. Monteiro, "On the use of honeypots for detecting cyber attacks on industrial control networks," in *12th European Conf. on Information Warfare and Security (ECIW 2013)*, 2013.

[41] C. Solari, "Designing for security," *Bell Labs Technical Journa*, vol. 12, no. 3, 207.

[42] K. A. Stouffer, J. A. Falco, and K. A. Scarfone, "Guide to industrial control systems (ics) security - revision 2," Tech. Rep., 2014.

[43] T. Suomen Automaatioseura, *Teollisuusautomaation tietoturva: verkottumisen riskit ja niiden hallinta*, ser. SAS julkaisusarja. Suomen automaatioseura, 2005. [Online]. Available: https://books.google.fi/books?id=pz8vAwAACAAJ

[44] T. I. C. S. C. E. R. Team. Standards and references. [Online]. Available: https://ics-cert.us-cert.gov/Standards-and-References

[45] J. R. Vacca, *Computer and Information Security Handbook, Second Edition*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2013.

[46] VMware, "Understanding full virtualization, paravirtualization, and hardware assist," Tech. Rep. [Online]. Available: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/VMware_paravirtualization.pdf

[47] ——, "vsphere 6.5 security configuration guide," Tech. Rep. [Online]. Available: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/files/xls/vmware-6.5-security-configuration-guide-ga-13-apr-17.xlsx

[48] ——, "Controlling virtual machine sprawl - how to better utilize virtual infrastructure," Tech. Rep., 2012. [Online]. Available: http://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/vmware-controlling-virtual-machine-sprawl-white-paper.pdf

[49] ——, "Security of the vmware vsphere hypervisor," Tech. Rep., 2014. [Online]. Available: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/whitepaper/techpaper/vmw-white-paper-secrty-vsphr-hyprvsr-uslet-101.pdf

[50] ——, “Performance best practices for vmware vsphereÂ® 6.5,” Tech. Rep., 2017. [Online]. Available: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/performance/Perf_Best_Practices_vSphere65.pdf

[51] ——. (2017) Vmware security advisory. [Online]. Available: http://www.vmware.com/security/advisories.html

[52] ——. (2017) Vmware vsphere 6.5 documentation center. [Online]. Available: https://pubs.vmware.com/vsphere-65/index.jsp

[53] H.-j. Woo, Y. Kim, and J. Dominick, “Hackers: Militants or merry pranksters? a content analysis of defaced web pages,” *Media Psychology*, vol. 6, no. 1, pp. 63–82, 2004. [Online]. Available: http://dx.doi.org/10.1207/s1532785xmep0601_3

[54] Y. Zhang, F. Monrose, and M. K. Reiter, “The security of modern password expiration: An algorithmic framework and empirical analysis,” in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS ’10. New York, NY, USA: ACM, 2010, pp. 176–186. [Online]. Available: http://doi.acm.org/10.1145/1866307.1866328

[55] M. Zhivich and R. K. Cunningham, “The real cost of software errors,” *IEEE Security Privacy*, vol. 7, no. 2, pp. 87–90, March 2009.