TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

PETTERI SIIK

# MANAGEMENT OF OPERATING SYSTEM HARDENING IN INDUSTRIAL CONTROL SYSTEMS

Master of Science Thesis

Examiners: Prof. Hannu Koivisto,
University teacher Mikko Salmenperä

Examiner and subject approved
on 9th August 2017

# ABSTRACT

**Petteri Siik**: Management of operating system hardening in industrial control systems
Tampere University of Technology
Master of Science Thesis, 46 pages
November 2017
Master's Degree Programme in Automation Technology
Major: Information Systems in Automation
Examiner: Professor Hannu Koivisto, University teacher Mikko Salmenperä

Keywords: hardening, Windows, security, PowerShell, configuration management, lifecycle

Hardening improves security by removing unnecessary features from the system. Hardening can be performed for a network, a device, an operating system and single applications. As virtualization is added, the virtualization environment must also be hardened. In this thesis, the focus is on operating system hardening and its management. Frequent operating system updates cause system changes that make hardening management challenging. System hardening is presented using the ICS lifecycle model. This includes tasks such as designing of the hardening configuration, implementation and testing, and maintaining the system hardening. To make implementation and maintaining of the hardening configuration possible two PowerShell scripts are made. One for automating hardening and other for auditing of Windows hosts. The scripts use a new hardening configuration template which is designed in this thesis.

As a result, effective scripts were implemented, though some features had to be dropped due to lack of proper tools. Discarded features and other development ideas are presented in further development section. Additionally, several challenges for hardening and using Windows 10 in control systems, are observed in this thesis. Most notable discovery is that Windows 10 restores hardened settings and even broke the operation of system without any apparent reason. For this reason, the hardening configuration should be monitored and its management continued through the systems lifecycle.

# TIIVISTELMÄ

**PETTERI SIIK**: Käyttöjärjestelmän kovennuskonfiguraation hallinta automaatioympäristössä
Tampereen teknillinen yliopisto
Diplomityö, 46 sivua
Marraskuu 2017
Automaatiotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Automaation tietotekniikka
Tarkastaja: Professori Hannu Koivisto, Yliopisto-opettaja Mikko Salmenperä

Avainsanat: koventaminen, Windows, tietoturva, PowerShell, konfiguraation hallinta, elinkaari

Koventaminen parantaa tietoturvaa poistamalla järjestelmästä turhia toimintoja. Koventamista voidaan suorittaa tietoverkolle, laitteelle, käyttöjärjestelmälle ja yksittäisille ohjelmille. Virtualisoinnin tullessa mukaan, täytyy myös virtuaaliympäristö koventaa. Tässä työssä keskitytään käyttöjärjestelmätason koventamiseen ja sen hallintaan. Usein päivittyvät käyttöjärjestelmät aiheuttavat järjestelmän muutoksia, mikä tekee kovennuksen hallinnasta haastavaa. Järjestelmän kovennusta tarkastellaan automaatiojärjestelmän elinkaarimallin avulla. Tämä sisältää tehtäviä kuten kovennuskonfiguraation suunnittelu, kovennuksen toteutus ja testaus, sekä järjestelmän kovennuksen ylläpito. Kovennuksen ja kovennuskonfiguraation ylläpidon mahdollistamiseksi toteutettiin kaksi PowerShell skriptiä. Toinen automatisoimaan kovennus ja toinen kovennuksen tarkastamiseksi Windows-käyttöjärjestelmässä. Skriptit käyttävät uutta kovennuskonfiguraatiomallia joka kehitettiin tässä työssä.

Lopputuloksena saatiin toteutettua toimivat skriptit, joskin joitakin ominaisuuksia jouduttiin jättämään pois työkalujen puuttumisen johdosta. Pois jääneet ominaisuudet ja muita kehitysideoita esitellään jatkokehitysmahdollisuuksissa. Lisähuomiona työssä havaittiin lukuisat Windows 10 -käyttöjärjestelmän aiheuttamat haasteet kovennukselle ja yleisesti automaatioasemien toiminnalle. Huomionarvoisin havainto oli, että ilman näkyvää syytä Windows 10 palautti kovennettuja asetuksia takaisin käyttöön hajottaen samalla automaatiojärjestelmän toiminnan. Tästä syystä kovennuskonfiguraatio tulee tarkastaa säännöllisesti ja kovennuksen hallinta tulee jatkaa koko järjestelmän elinkaaren ajan.

# PREFACE

This Master of Science thesis was written for Laboratory of Automation and Hydraulics, in Tampere University of Technology, to increase hardening management in the distillation column control system. This project gave me experience in security functions and configuration management in industrial control systems, and much more understanding of hardening in Windows operating systems. This was useful while upgrading the distillation column computer hosts to new operating system versions.

I would like to thank Professor Hannu Koivisto for providing this thesis opportunity, and Mikko Salmenperä and Jari Seppälä for good advices, guidance and feedback during this thesis.

Tampere, 30.11.2017

Petteri Siik

# CONTENTS

## LIST OF FIGURES

# LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| AD | Active Directory |
| COTS | Commercial off-the-shelf |
| CIA | Confidentiality, Integrity, Availability |
| CSV | Comma Separated Value |
| DCS | Distributed Control System |
| DMZ | Demilitarized Zone |
| FAT | Factory Acceptance Test |
| HIDS | Host Intrusion Detection System |
| HMI | Human-Machine Interface |
| ICS | Industrial Control Systems |
| IDS | Intrusion Detection System |
| IDPS | Intrusion Detection and Prevention System |
| IED | Intelligent Electronic Device |
| IP | Internet Protocol |
| IPS | Intrusion Prevention System |
| ICT | Information and Communications Technology |
| MOF | Managed Object Format |
| NIC | Network Interface Card |
| NIST | National Institute of Standards and Technology |
| OS | Operating System |
| OSI | Open Systems Interconnection |
| PLC | Programmable Logic Controller |
| SAT | Site Acceptance Test |
| SIEM | Security Information and Event Management |
| SCADA | Supervisory Control and Data Acquisition |
| SMB | Server Message Block |
| STIG | Security Technical Implementation Guide |
| TCP | Transmission Control Protocol |
| UDP | User Datagram Protocol |
| VLAN | Virtual Local Area Network |
| VPN | Virtual Private Network |
| WinRM | Windows Remote Management |
| XML | Extensible Markup Language |

# 1. INTRODUCTION

The biggest ongoing trend in industrial control systems (ICSs) is the increase of networking capabilities. When control networks are connected to organization network or clouds, information security becomes more important than if the ICS network is isolated. Compromises in security can lead to unnecessary system shutdowns, quality corruptions or even safety issues. The basic idea of ICS security is to use multiple layers of protection. This includes separating the automation network from office network with firewalls and a demilitarized zone, segmenting the automation network and protecting computers with software firewalls, virus protection and system hardening. In an ICS, the purpose for increasing security is to guarantee safety and reliability of operation, and this must be the primary concern when securing the system.

The term hardening means removing or disabling system features, programs or ports to reduce systems attack surface thus improving security. The result of hardening is hardening configuration or secure configuration for the system. Hardening must be done in different layers from hardware hardening to application hardening. Figure 1 shows the hardening layers. In the left model is hardening layers of a system running directly at a hardware. This includes hardware hardening, operating system hardening and hardening all individual applications. If virtualization is used, there can be added hardening of the virtualization environment, like in the model at right in Figure 1. In addition, the ICS network where all the hosts are operating, must also be hardened with firewalls. This thesis focuses on operating system hardening though some measures presented can also be used in other hardening layers.

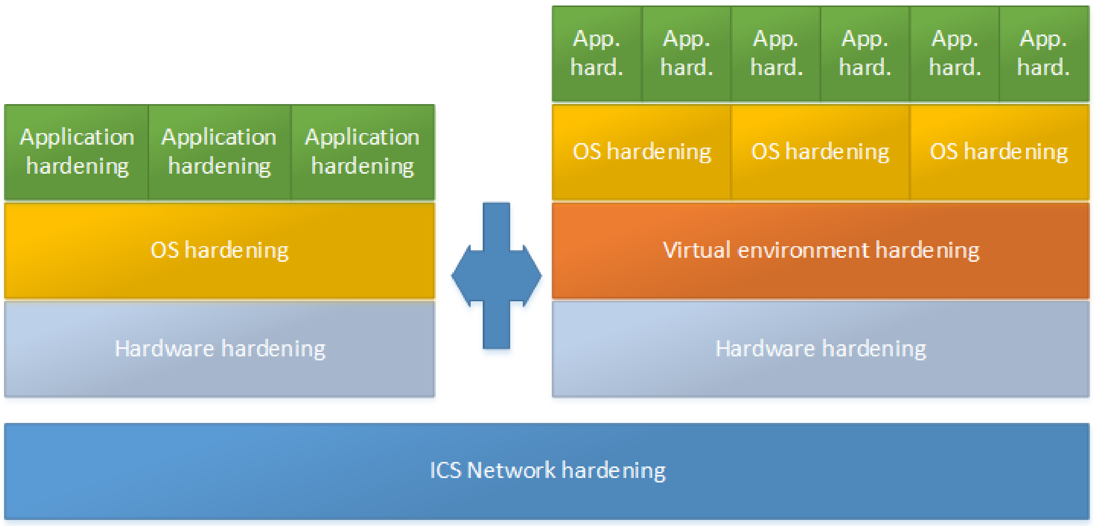

***Figure 1.*** *Layers of hardening.*

Operating system (OS) hardening is typically done just after OS is installed, and hardening configuration is assumed to remain the same after that. However, the reality is that the configuration changes over time due to patches and other system changes. While ICSs adapt more commercial-of-the-shelf (COTS) technologies, the need for frequent patches and other changes increases. The importance of patching software regularly has been increasing in recent years. This is because operating systems and applications are becoming more complex, which leads to more vulnerabilities. Vulnerabilities are also searched and fixed more than before. New vulnerabilities are found all the time and some of them such as recent Meltdown and Spectre vulnerabilities received global attention [1]. Criminals have realized the possibilities of vulnerabilities and are creating new malicious software to exploit those vulnerabilities. Hardening management is the way to keep the system hardened in a changing environment.

This thesis studies hardening management in context of the lifecycle of an industrial control system. The lifecycle includes design, build, operation, and decommission phases. Theses phases include tasks for developing, implementing and maintaining the hardening configuration. These phases and tasks are used to develop a lifecycle model for hardening management. The idea for this is based on Finnish KYBER-TEO-project [2]. Maintaining the hardening configuration in a regularly patched system needs auditing of required settings and reconfiguring of incorrect settings. Manual auditing and configuring is really time consuming and because of that periodical audits are usually not included in the patching process. This problem can be solved by creating automated scripts for these measures. Hardening and auditing scripts are implemented in this thesis to prove this concept in a real control system.

Chapter 2 introduces industrial control systems characteristics, the lifecycle model, as well as risks and threats that affects an ICS. The third chapter is about how security of ICSs can be obtained, and what kind of security functions and features are used for securing control systems. These include security measures for an ICS network and host devices, but also security-focused configure management which is used for the development of hardening management. Security testing is also needed in hardening management. Security testing in ICSs is presented in chapter 4. In chapter 5, the lifecycle approach for hardening management is introduced. The chapter focuses on hardening related tasks in each phase of the ICS lifecycle and visualizing the tasks with flow charts. The Sixth chapter describes the environment where this concept is to be used. It presents the distillation column network and especially focuses on operating systems, introducing useful tools, and features that should be considered in hardening. Implementation of the scripts needed for hardening management is presented in chapter 7. The implementation also includes an XML based hardening template. The template contains required hardening settings and exceptions to manage all possible hardening baselines. The final chapter has conclusions about hardening management, using PowerShell for hardening, and the need for hardening when using Windows operating systems in an ICS environment.

# 2. INDUSTRIAL CONTROL SYSTEMS

The term industrial control system (ICS) is used for different types of control systems, such as Supervisory Control and Data Acquisition (SCADA), Distributed Control Systems (DCS), and Programmable Logic Controllers (PLC). SCADA systems are used for highly distributed large area control systems, such as water and electricity distribution, where the need for a centralized data acquisition is as important as control. SCADA is used to collect data from remote stations to a centralized control center where operator can monitor or control the entire system in near real time. This requires long distance communication networks between field sites and the control center. Unlike SCADA, DCS is used in production systems located in smaller areas and uses local area networks. It is used in water treatment, electric power generation plants and other production systems, using process control or discrete control systems. Programmable logic controllers are used in both SCADA systems and DCSs as control components to provide local management of process. PLCs can also be implemented as primary controllers in small control systems. [3]

Traditionally ICSs have been very isolated systems, but this is not the case nowadays. Connectivity with business operations, remote maintenance and other data exchange needs caused ICS to emerge with corporate network [4]. Connectivity does not include only network connections, but also removable media and laptops used for maintenance. In addition, today's ICSs use many Commercial-off-the-shelf (COTS) technologies that are more targeted by online criminals than special automation technologies. Default communications are using Ethernet and TCP/IP, and operating systems such as Windows and Linux are found from engineering and monitoring stations as well as embedded devices. [5]

## 2.1 ICS characteristics

Although today's industrial control systems have lot in common with traditional ICT systems, ICSs have some unique characteristics that must be understood when securing a system. These characteristics differ from typical ICT systems and must be considered when designing security features in ICS.

The most fundamental requirement of ICS is dependability, which consist of separate means such as robustness, security, safety, usability and other features that affect stability and reliability of a control system [6]. Due to dependability requirement, ICSs have different security objectives than traditional ICT systems. In information technology, security seeks to achieve confidentiality, integrity, and availability (CIA). These objectives are organized usually by confidentiality in priority. Integrity then falls into second place and availability after that. In ICSs, these objectives are arranged in different order, availability as the priority. Unexpected outages are not acceptable in control systems [3]. Control and monitoring operations also cause integrity to come at second place. Confidentiality has usually lesser importance, because data is often in raw form and must be analyzed within the context to have value. [7]

Another great difference is that while typical lifetime of ICT components is around 3-5 years, lifetime of an ICS system can be 10-15 years or sometimes longer. In this lifetime, the ICS can have major renewals and modifications during development and operation. [8] The longest lifecycle only applies to the core system that consists of valves actuators, PLCs. Update cycle for other hardware can be for example every 5 years. [6] Long lifetime also leads to control systems having dependence on legacy ICS technology and can include a lot of old technology. For this reason, these systems can have devices that do have resources for modern ICT security features.

ICSs are very time dependent systems, with lot of real-time requirements. This means that operations in time critical loops must be periodic and deterministic. The time frame of measurements and control signals must be predetermined. Real-time requirements affect in automation protocols, operating systems and software used in control systems. Applying common ICT measures into control systems needs deep knowledge of the system and careful consideration and planning. [6] Also unlike typical ICT systems, high throughput connections are not usually essential for ICS. [3]

Finally, processes in control systems are usually continuous by nature. This means that planning and scheduling of outages must be done days or even weeks in advance. Unexpected outages cannot be accepted in systems controlling the processes. For this reason, ICS systems cannot be easily stopped and started. Unplanned outages can even cause physical consequences. [3]

## 2.2 The ICS lifecycle

The lifecycle of an industrial control system can be divided into several phases. In this thesis, the lifecycle consists of 4 phases which are design, build, operation and decommission phases [8]. These phases consist of multiple steps and actions to build and operate ICS. The ICS lifecycle and its phases are presented in Figure 2 with main actions of each phase.

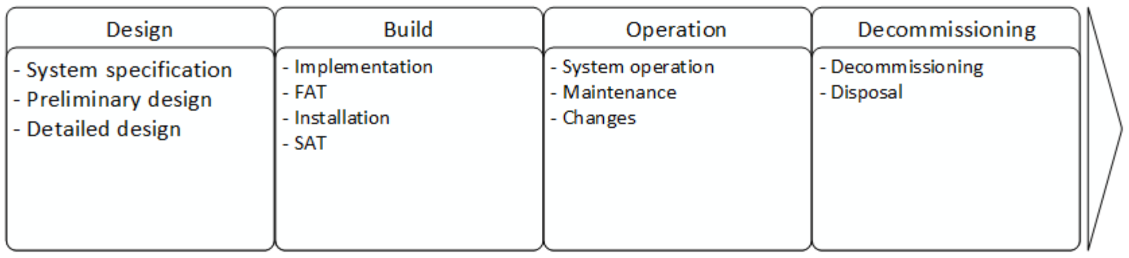

***Figure 2.*** *The Industrial control systems lifecycle [8].*

First phase in the ICS lifecycle is a design phase which purpose is to design a target system so that it can be built, operated and maintained [9]. The phase includes preliminary design for developing system specifications, and basic design to describe the system accurately enough for beginning detailed design. Detailed design improves results from previous design steps with more hardware and software dependent instructions. Results of system design consists of software and hardware design specifications, which are further developed with implementation design to detailed software module design specification. Testing plans should also be developed while designing the system. [10]

After system is designed, it can be implemented. This means that a control system vendor prepares software and hardware components and assemblies it in vendors facility. In large systems, implementation can be divided into multiple parts. The implemented system is then tested with factory acceptance tests (FAT) to ensure that the control system is ready to delivery. Delivery release requires approval from both the vendor and the customer. Then the system is delivered into the customer's facilities, where it will be built. The complete system is then tested with cold and hot commissioning to ensure correct functioning of the system. Site acceptance testing (SAT) is done last to demonstrate that the system matches its design and can be released to the customer. [10]

The operation phase can start when the control system is released and process qualification has ended. The operation phase of the ICS can be very long and include major renewals, modifications and business integration needs, that changes the system. Changes in control systems need precise change management process and might need re-applying tasks from previous phases. [8] [10]

The decommissioning phase is the last phase of the industrial control systems lifecycle. The phase consists of decommission and disposal of the system. These actions should be considered and planned already in the design phase. Environmental and recycling aspects must also to be considered. Usually a lot of data from the system is also stored after the system is disposed. Decommissioning should be controlled the same way as the system modifications in previous phases by using configuration change management. Any new security risks related to decommissioning also needs to be understood. [8] [10]

## 2.3 Risks and threats in ICS

Due to physical operations, risks in ICS includes health and safety risks and physical damage in addition to information leaks and financial risks. Risks in a single factory focuses mostly on human safety and health, but attacks against critical infrastructure industries can lead to even nationwide danger. This can also lead to criminal or national interests in attacking such systems. Attacks can vary from targeted attacks such as the Stuxnet, which was governmental attack against single facility, to widespread malicious software. [3] [6]

Threats in modern information systems include three architypes that differ from their motivation, goals and resources. Hacktivists are typically amateurs, who are testing their skills and showing off to their peers. They usually have rather limited resources and their objectives are not always clear. Damage done by hacktivists is not always intentional, but they can still cause accidental harm, especially if attacking against a fragile real-time control system. Organized online criminals purpose is to make money, their resources are beyond hacktivists. They can sell attacks or tools to commit attacks. The rise of ransomware also suggests that digital blackmailing can be profitable [11]. The last type is governmental operators. They have the biggest resources and their operations are well coordinated and difficult to defend. [6]

To manage these risks, organizations need risk management to evaluate risks associated with their businesses, and to make decisions how to deal with those risks. Risk management is an interactive ongoing process that is part of normal operations. Risk management should be deployed through an organization in three levels: An organization level, a business process level and an information system level (ICT or ICS). At the organizational level, the first component of the risk management is implemented. This is risk framing, which provides the context for all activities in risk management and provides the basis for risk management throughout the organization. The business process level addresses risks from a mission or a business process perspective guided with the risk context, risk decisions, and risk activities at previous level. At this level, the core ICS functions and processes are defined and prioritized with respect to overall goals of the organization. Also, an ICS information protection strategy is developed, and ICS security requirements incorporated into the operational processes. Activities at information system level are guided by decisions and activities at both previous levels. Risk management activities at this level include five functions which are performed as a part of a system development life cycle process. These functions are identify, protect, detect, respond and recover. [8]

Improving cybersecurity needs proper knowledge of risks included. Developing and understanding business risks associated with cybersecurity is the base for implementing the defense-in-depth strategy in ICSs. [4]

# 3. INDUSTRIAL CONTROL SYSTEMS SECURITY

Industrial control systems are large and complex systems. Therefore, no single information security solution can protect the whole control system. Using multiple solutions and layers of defense is called defense in depth. The defense-in-depth strategy consists of multiple different defensive measures varying from security training and management to more technical security features, such as firewalls and virus scanners. United States Department of Homeland Security listed elements of the defense-in-depth strategy in their Recommended Practice: Improving Industrial Control System Cybersecurity with Defense-in-Depth Strategies guide. These are compiled into a mind map in Figure 3. The defense-in-depth strategy starts with understanding and managing the business risks associated with the ICS security. A risk model that contains functional requirements of the ICS is essential when designing an effective cyber security architecture. [4]

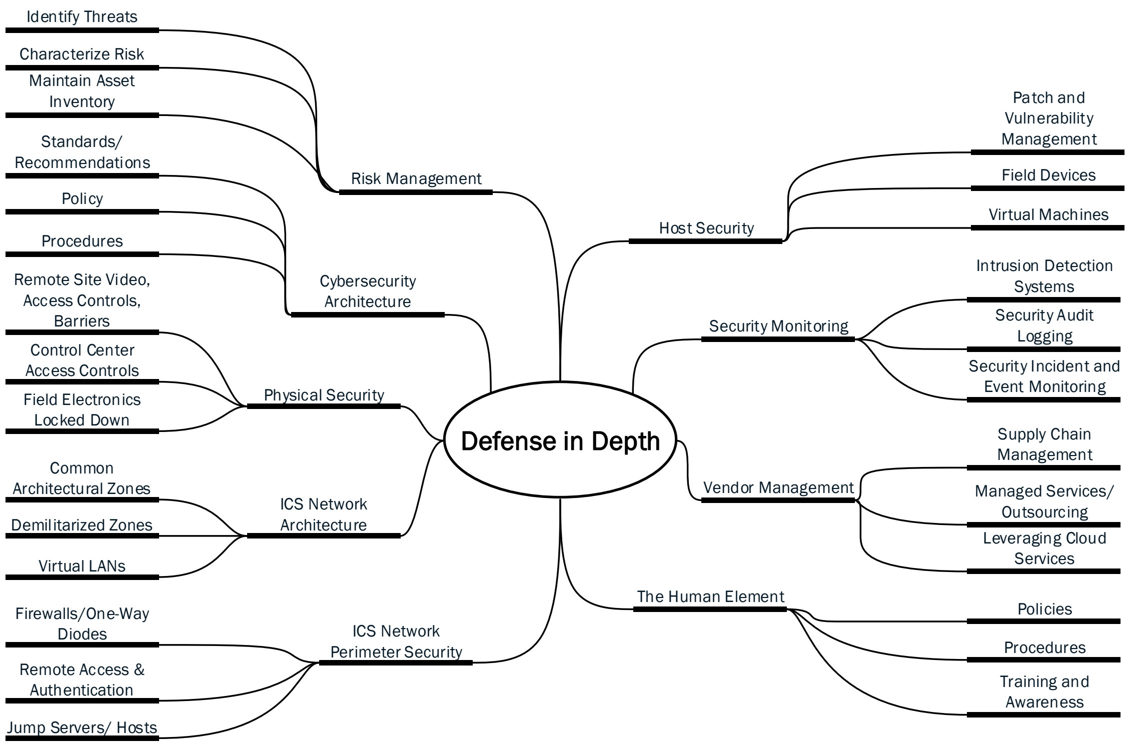

***Figure 3.*** *Defense-in-depth elements [4].*

Using multiple defensive layers defines that while the ICS network is isolated from a corporate network with firewalls and a demilitarized zone, it is still important to increase security of each device with host security. The reason for this is that there is always some possibility that an attack penetrates or circumvent defenses, for example, by plugging an infected device into the control system. Host security will be the last line of defense.

Following sections describe protecting and monitoring features for both network side and host device.

## 3.1 Securing the ICS network

Formerly ICS networks have been secured by physically isolating them from organizational networks. Today’s control systems share information with enterprise level systems and have remote management possibilities. For this reason, total isolation of the ICS network is usually not an option. Network security tools provide means to isolate the ICS network as much as possible, and to control and monitor network access. It mostly relies on secure network architecture, and controlling and monitoring network traffic with firewalls or other devices, such as intrusion detection systems.

While this thesis focuses on host security, it is important to understand how the surrounding ICS network is secured. This section includes some basic knowledge about the secure ICS network architecture and tools used to protect the network.

### 3.1.1 Network firewalls

Network firewalls are used to control network traffic between different networks. Network firewalls typically restrict connectivity with the internal and more hostile networks. This can be, for example, between an office network and internet or an ICS network and the office network. Network firewalls can operate at different layers in the Open Systems Interconnection model (OSI), OSI model divides networking into 7 different layers, which are physical, data link, network, transport, session, presentation, and application layers. Firewalls provide protection by blocking all communications except between essential devices, enforcing secure authentication of all users that are accessing the network, allowing users to connect only allowed destinations, and recording network traffic information. [3]

Different classes of firewall include packet filtering firewalls, stateful inspection firewalls and application-proxy gateway firewalls. Packet filtering firewalls are the most basic type of firewalls. These firewalls inspect each network packet for their basic information and makes a decision by using a set of criteria. The decision can be either to drop the packet, forward it or send the message back. While packet filtering firewalls are checking packets only at the OSI level 3 (network layer), stateful inspection firewalls also use the transport layer (OSI level 4). This adds tracking for active sessions and uses that information to determine if packets should be forwarded or blocked. Application-Proxy Gateway Firewalls increase inspecting of packets at the application layer, so firewall rules can be application or protocol specific. These kinds of firewalls can effectively increase security, but can lead to increased delays that are intolerable in ICSs. [3]

## 3.1.2 The secure network architecture and segmentation

The ICS security architecture relies on separating the industrial network from more hostile networks, such as a corporate network and the internet. Figure 4 includes a model of the recommended secure network architecture. Different segments are separated with firewalls, and a demilitarized zone (DMZ) is added between the ICS network and the corporate network. All servers that need to be accessible from outside the control system must be included in the DMZ segment. Network segmentation can also be increased by partitioning the ICS into smaller networks, for example, based on different functions or uniform policies. This can minimize accessing sensitive information, communications or configurations, and prevent effects, from malicious software or non-malicious errors and accidents, spreading in the system. [3]

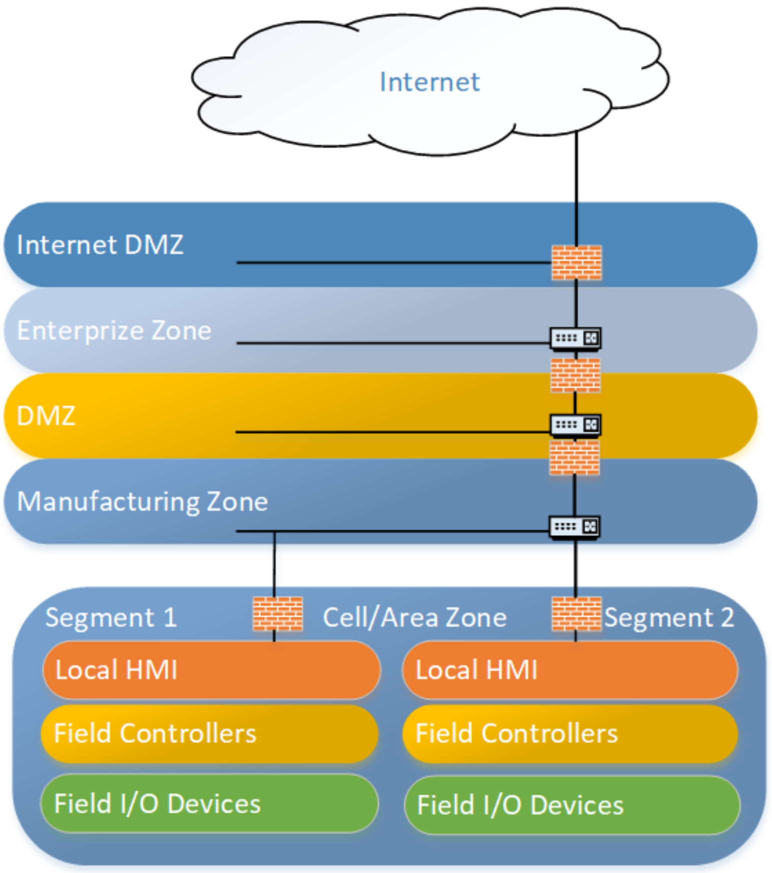

***Figure 4.*** *Security levels and segmentation. [4]*

There are some different technologies and methods that are used for network segmentation. These include logical network separation with encryption or network device-enforced partitioning using virtual local area networks (VLANs), virtual private networks (VPNs) or unidirectional gateways to restrict communication to a single direction. Secondly there is physical separation to prevent any connectivity between network segments, and the third technique is network traffic filtering using firewalls in different network layers. [4]

### 3.1.3 Intrusion detection and prevention systems

Firewalls can be good at controlling the network traffic, but they do not monitor and analyze the events for signs of possible threats and violations of security policies. This can be done with intrusion detection systems (IDS). While IDSs are only monitoring incidents, intrusion prevention systems (IPS) also have capabilities to prevent those incidents. IDS and IPS technologies share many key functions, but an IPS can also respond to a detected threat with methods that attempt to prevent the intrusion from succeeding. [12]

Intrusion detection and prevention systems (IDPS) have three key functions. They record information of observed events. This is usually done locally, but data can be sent to centralized servers or security information and event management (SIEM) systems. Secondly, IDPSs also notify administrators about events that are important. These alert notifications usually include only some basic information. Additional information is then found by accessing the IDPS. Thirdly, IDPSs produce summarized reports concerning monitored events or other events of interest. Intrusion prevention introduced with an IPS can be performed, for example, in following ways. The IPS can terminate the network connection or the user session used for the attack, or block access to targets by using attributes, such as a user account or an IP address. The IPS can also block all access to targeted hosts. [12]

IDSs and IPSs are both usable solutions in control systems. ICSs use many unique communications and protocols that are not seen in typical IT. This adds a need for control system related signatures, which is now being addressed by control system vendors and organizations specialized in ICS security. Network traffic in control systems is very predictable, this provides an opportunity for IDS solutions to detect traffic that is outside normal operations. While IDSs use more passive methods, IPSs active protection methods need to be carefully considered in control systems because they can have a potential to stop the process and operation. Ideally, intrusion and prevention systems are used in between organizational and control system networks, to track possible intrusions and increase the speed for countermeasures if the system is being exploited. [4]

## 3.2 Securing ICS host devices

Protecting your network, as well as individual machines, is a part of good network management principles [13]. While network security protects devices from outside attacks, host-based security features add another layer of security, which is the basic idea of defense in depth [4]. Host security can also increase protection of inside attacks, threats caused by human errors, and it can prevent or slow down malware or intrusion from spreading over the control system. Host security features can also increase the chance of detecting intrusions. Hosts include network devices, workstations and server computers, with different operating systems and system resources. [14]

Methods of securing an ICS host are the same as in common ICT, but there are some special requirements for implementation and usage. Following sections will introduce a few basic security features and how they are used in industrial control systems.

### 3.2.1 Host hardening

Host or system hardening means changing the default configuration of a network device to reduce its attack surface. It also includes configuration changes that can make abusing of necessary system features more difficult. This can be done by configuring the operating system, user accounts, applications and other required software. [13] To limit possible entry points, only services that are used in control systems for operations or maintenance, should be enabled. All unused services in host operating systems are possible entry points for exploitation. Also, these services are usually not monitored, because services are not in use. [15]

Every software package in a system increases its attack surface.  This includes applications, services and other software. All these can have known and unknown vulnerabilities that can be exploited for attacking purposes. By removing those resources, the attack surface can be reduced. This also decreases the number of updates needed to maintain a secure application stack. [16] Figure 5 illustrates the basic idea of reducing the attack surface. Removing software also removes unknown vulnerabilities included in them. However, hardening does not remove the need for patching of software that cannot be removed.

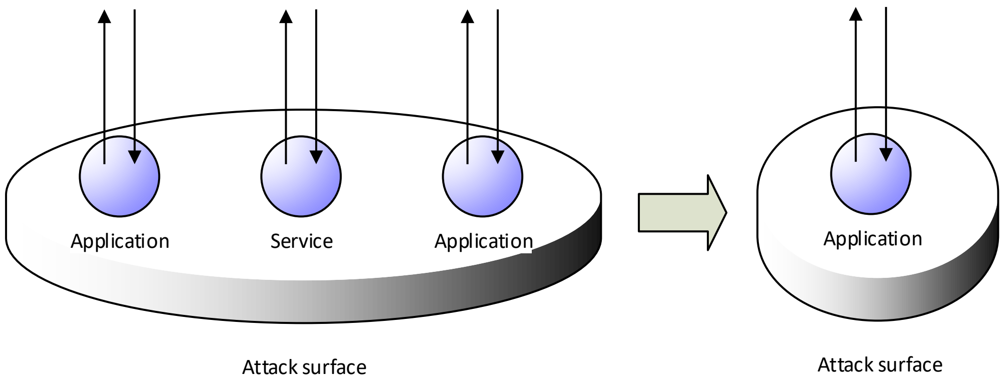

***Figure 5.*** *Reducing the attack surface.*

The reason for why there are parts that can be removed is because modern computer operating systems are usually designed for multipurpose use. This means that the operating system should offer all resources needed in all kinds of purposes like workstation use or gaming. Industrial control system usage for a single device is much more restricted than in a normal desktop use, but many ICS computers still have a basic desktop operating system installed. Even server operating systems usually include multiple unneeded functions for a computer used in an ICS. [15]

In addition to removing features, it is necessary that user accounts only have permissions that are needed in control operation and unnecessary permissions are removed. Using default usernames and passwords or too easy and short passwords is one critical flaw in many network devices. This makes configuring users an important part of hardening. It means removing default and unused usernames, using minimum password requirements and setting password expiration. [13] [15]

Successful hardening requires that hardening is performed to every new system before connecting the computer into any network, not even in a local network. Secondly, user accounts must not have permissions that are unnecessary for operations. Finally, it is most important that hardening does not disturb the operation of control system programs. [13] Additionally, hardening often requires support from vendors and service providers. Especially when hardening ICS hosts with special protocols and real-time requirements. [17]

### 3.2.2 Host firewall

In addition to network firewalls that provide protection for network segments, level of security can be increased by firewalls that protect single hosts. Host firewalls are software based firewalls that protect ports and services on the device they are installed. Many modern operating systems have integrated firewalls, but there are also many third-party solutions. Host firewalls have rules for tracking, allowing and denying incoming and outgoing traffic. A typical firewall configuration denies all traffic except what is necessary for operation. [14] Restricting outgoing traffic can also prevent certain malware from spreading outside of the host.

Host firewalls can usually be configured to allow certain applications and deny other applications, in addition to traditional stateful filtering. This kind of firewall is called an application-based firewall. Some host firewall programs also include antimalware, intrusion detection and intrusion prevention features. Additionally, host firewalls can be used to achieve increased network traffic logging. [18]

Various network technologies used by automation software cause major challenges for using host firewalls in ICSs. [13] Host firewall also uses host's recourses, which can interfere with the operation.

### 3.2.3 Application whitelisting

Unlike in blacklist solutions, where objects that are listed as bad are blocked, in whitelisting all objects are blocked and only whitelisted are allowed. Application whitelisting solutions allows only listed applications to run and everything else is blocked. Benefit in this method, compared to blacklisting, is that there is no list of harmful applications that must be continuously updated. Secondly, whitelisting can also help to protect against zero-day attacks. [14]

In ICSs, usage of different programs, ports and services is well defined, and continuous updating is difficult to achieve. This makes application whitelisting well suited for control systems [3]. Also, the set of applications running in an ICS is essentially static, which makes application whitelisting more practical. [14]

### 3.2.4 Virus detection

Virus detection or antivirus software are programs that scan and inspect files to find malware. To do this, the program needs a list of known malware signatures to which the file signatures are then compared. When the signature of a file matches, the file is usually quarantined and deleted or cleaned later. Virus detection can only work if the malware is on the list. For this reason, the signature list must be regularly updated. [3]

To use virus detection programs in an ICS, some special practices must be followed every time new signatures or software updates are installed. These practices include compatibility checks, change management issues, and performance impact metrics. Computers used as consoles, engineering workstations, data historians, human-machine interfaces (HMIs), and general purpose SCADA and backup servers can be secured much like commercial IT equipment. However, other servers and computers with special features should follow vendor recommendations. [3] In addition, it is necessary to remember that scanning can introduce latency changes in delicate systems. Continuous updating of virus definitions and software updates are also difficult to manage in control systems, where downtime is minimal.

### 3.2.5 Host IDS

Host-based intrusion detection systems (HIDSs) operate basically the same way as network IDSs. They monitor network traffic and the state of a host computer, try to detect any threats, and alert the administrator if an intruder is compromising the host. HIDSs can use two types of detection methods, signature-based and anomaly-based detection. Signature based detection watches for specific events that match the signatures. They can only detect known threats, but are easier to tone manually. Anomaly-based detection systems are detecting threats by watching changes in trends. This means that they can possibly detect even unknown threats. [4]

The challenges of using a HIDS in ICS host device are mostly the same as using virus scanners. Computer resources can be very limited, so intrusion detection systems using these resources can cause harm to control software and operation. Another challenge of using HIDS in an ICS, is the need for continuous updates which is typically quite difficult to manage in ICSs. [4]

### 3.2.6 Patch management

Providing system fixes for software bugs and vulnerabilities is called patching. Process for identifying, acquiring, installing, and verifying patches for products and systems is called patch management [19]. Generally, ICSs have been very static environments. However, while the ICSs use more and more typical information system operation systems, protocols and software, the need for patching and other configuration changes increases. [20] Still, the ICS environment causes challenges for patch management that are less common in normal ICT environments.

Modern-day computers and software are very complex. Operating systems and even third-party applications can have millions of lines of code, and because of that, there can be many possible locations to have vulnerabilities. This means that almost every operating system and other application will have some vulnerabilities and new vulnerabilities are found every day. Patches can fix those vulnerabilities or add new features. Many software developers are patching their programs if the program support time has not ended. [19]

In an ideal world, patches would be delivered immediately when available, but because of limited organizational resources, that is not possible. Also, in industrial control systems, patching is even more difficult than in ICT systems. Typical patching needs frequent downtime, but in ICSs this is not possible. Any unexpected downtime in a control system can lead to catastrophic failures in operation. Therefore, patching process must be carefully scheduled and planned, devices must be updated only during maintenance, and patches need to have extensive operational testing before deploying into a production system. [20]

Developers are not patching their software forever, and even after support has ended, new vulnerabilities are still found. This must be considered already when choosing software, because later software changes are usually more difficult. To avoid this, it is better to use programs with a long support time. Due to the long lifecycle of ICSs, there can usually be found old, unsupported software [20].

## 3.3 Security-focused configuration management

As ICSs become closer to traditional information systems, changes become more frequent. Information systems are constantly changing with new updated hardware and software patches. To ensure that system changes do not harm operation of the system and therefore the organization, an information security integrated configuration management process is needed.

Security-focused configuration management is based on four different phases which are seen in Figure 6. These are planning, identifying and implementing configurations, controlling configuration changes, and monitoring. These phases have multiple tasks such as developing configuration baselines, reviewing and implementing those baselines, change management, and security auditing. Monitoring activities may also loop back to any of the previous phases. [21]

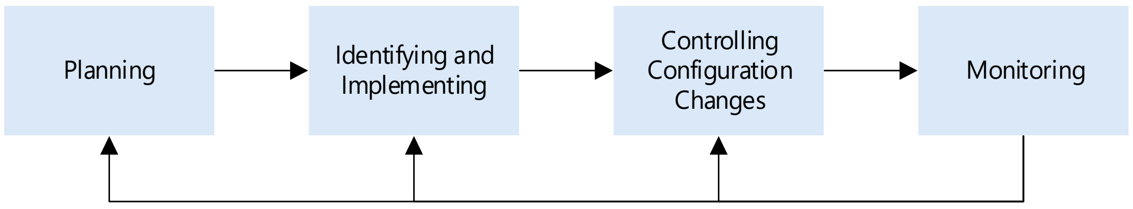
Planning
Identifying and Implementing
Controlling Configuration Changes
Monitoring

***Figure 6.*** *Security-focused configuration management [20].*

The objective for the planning phase is to develop policies and procedures for security configuration management. This involves organizational and system level planning. Organizational level planning normally includes establishing an organization-wide configuration management program, and developing organizational policies and procedures for security-focused configuration management. On the other hand, system level planning consists of a system level configuration management plan to provide system-specific information that is related to configuration management. This includes a component inventory and a list of identifiable parts which are targeted to configuration control. These are called configuration items. These configuration items can be hardware, software or firmware, but also documents or combinations of former items. This kind of documentation helps to provide a comprehensive view of the components and configurable parts which are managed. [21]

Identifying and implementing activities are usually done at the system level following the organizational policies and procedures. The result of this phase is a documented and approved secure configuration baseline for each configuration item. Identifying and implementing actions include establishing secure configurations. These can be based on organization policies, federal laws and standards or generally accepted common security guidelines such as NISTs the United States Government Configuration Baseline (USGCB) [22]. Commercial product vendors are also important source for secure baseline configurations. Implementing these configurations can be a difficult task due to many different products each with countless possible configuration parameters. The following approach is recommended when implementing the secure configuration. The first part is to prioritize configurations by system impact, risk assessment and security testing, to which systems, products or configuration items are first to apply security configuration management. The second part is to fully test the configurations to find issues, such as software compatibility or device driver issues. After that, the problems or deviations found in testing must be resolved and documented. The fourth part is recording and approving the baseline configuration to support change control. The recorded and approved baseline becomes the initial configuration baseline for the system. Finally, the baseline can be deployed to the system. This is encouraged to be managed in centralized and automated manner by using configuration management tools or automated scripts [21].

Continuous technological development and new, possibly more serious threats, are driving changes in information systems. Maintaining security in a changing environment requires system configuration changes to be managed and controlled. Controlling changes requires access restrictions, controlling of who is able make changes, and implementing the configuration change control process. This process can have several steps that generally are: requesting a change, recording the request, determining if the change requires configuration control, analyzing security impacts, testing of security and functional impacts, approving the change, implementing the approved change, and finally verifying that the change was implemented correctly and closing the request [21].

The purpose for monitoring is to confirm that the current configuration is identical with the approved baseline. Monitoring also helps to identify if all components included in the component inventory are available, and no unapproved components are found. Unapproved components are usually left without patches and might not have a proper hardening configuration implemented. Monitoring is performed with assessment and reporting activities. These activities can be impractical or impossible to do manually because of the large number of components available in the system, so using automated solutions is recommended. Monitoring activities includes scanning of components and configurations, using automated change monitoring tools, monitoring logs and records, running system integrity tests, and reviewing configuration change control records [21].

# 4. SECURITY TESTING IN ICS

Security testing is done to assess the level of security and to detect new vulnerabilities. This can be valuable information for improving security and for security reports. Testing is done by targeting an assessment object with special conditions and comparing results with expected results. Information security assessment includes two methods that complement technical testing. These are examination and interviewing. Examination is a process of analyzing the object with reviews and checks, or other ways studying the object. Interviewing, on the other hand, is collecting information from individuals or groups in the organization, with discussions and surveys. [23]

Security testing or assessment includes at least 3 different phases. These are planning, execution, and post-execution. Planning is critical for successful security assessment. It includes gathering information, such as what is going to be assessed, what threats it has, and what security controls are being used. The goal for execution is to identify and validate vulnerabilities. This phase includes using security testing tools introduced in this chapter. The post-execution phase focuses on analyzing vulnerabilities, finding what caused them, establishing mitigation recommendations, and writing the final report. [23]

In an ICS network, latency or jitter can be harmful for many protocols, so security testing in an active system can affect operation and cause the system to fail. Most comprehensive security testing should be done before the ICS is in operation or during scheduled maintenance windows or other downtime periods. [8] Testing can also be performed in a testing environment with similar installation configuration as in the real system to avoid unwanted consequences. Following sections introduce some basic testing techniques and how they can be used in ICS networks.

## 4.1 Network scanning

Network scanning is used to collect device information and network traffic. The information can include open services, used ports, and operating systems used in hosts. To perform network scanning, it is necessary to be able to access the same network as the tested system, either directly or through the internet. [24]

Network scanning can be active or passive. In active scanning, network packets are sent to network devices to get responses. Response packets can then be analyzed for information about the system, such as used operating systems, open ports, and available services. Active scanners, such as port scanners, usually identify devices in given address range and run port checking for open TCP and UDP ports. Any vulnerability discoveries rely on conclusions made by the person doing the scanning.

Passive scanning or network monitoring is quite different. It relies on monitoring network packets traveling in the network. This usually takes more time than active scanning, but can still find information about devices and ports they use, as well as used operating systems. Network monitoring can also discover relations between devices. Sometimes it is possible to even find passwords or other valuable information that have been sent through the network, if encryption is not used. More quiet devices on the other hand may even be unnoticed during observation time. [23]

Both passive and active methods can be valuable in assessing industrial control systems security. Passive network monitoring can be used to find unnecessary network traffic in the network, and is more likely not to cause any trouble for the system. Active scanning can be more efficient in mapping ICS network devices or verifying the security state of devices. [23] Because active scanning can cause problems in ICS systems in operation, scanning is preferred to perform for a duplicate system or in a development or a testing environment.

## 4.2 Vulnerability scanning

Vulnerability scanners use network scanning tools to collect information about the network. The difference is that unlike network scanners, vulnerability scanners are identifying the known vulnerabilities from collected data and do not rely so much on the user. Vulnerability scans can be performed for information collected with network scanners, but many vulnerability scanners include their own network discovery and port scanning tools. Vulnerabilities are identified by comparing results with a database of known vulnerabilities. This means that vulnerability scanners cannot discover any unknown vulnerabilities, and that the database of the scanner needs to be updated regularly.

Vulnerability scanners use multiple tools to collect information. Capabilities can include identifying of active hosts, active services, applications, operating systems and vulnerabilities related to those operating systems. Vulnerability scanners can also include detecting misconfigured settings, testing application usage and security policies, and establishing the foundation for penetration testing. Vulnerability scanners can be host-based and network-based. Host-based scanners are installed in each host that is tested, and they are used to scan operating system and application misconfigurations and vulnerabilities. Network-based scanners, on the other hand, are used for mapping the organization's network and identifying open ports and related vulnerabilities. [25]

Because vulnerability scanners use active scanning methods, they cannot be easily used in an operational ICS network. The best way to perform vulnerability scanning for an ICS is to use it in a test or a development environment. This way it cannot harm physical processes, but could reveal useful vulnerability information if the testing environment is similar enough to the real system. [26]

## 4.3 Penetration testing

Penetration testing simulates a real-world attack to identify means for circumventing security. Testing can determine how a system tolerates the attack, a level of sophistication that the attack needs to successfully compromise the system, what additional security measures could be added against these threats, and system's ability to detect attacks. Penetration testing needs great expertise and can be very labor intensive. Penetration testing can be done with or without the organizations ICT staff's knowledge, but always with permission from upper management. These testing types are called Blue Teaming and Red Teaming. Simulated attacks can be designed to be inside or outside attacks, or both. In external testing, attackers are not provided with any information about the environment and the attack is done outside of the internal network. Internal testing assumes that the attacker is already inside the network and has some basic knowledge of the environment. [25]

Penetration testing generally includes multiple steps. At first, hosts and networks are identified. This can be done with ping sweep or examining routing tables. After the hosts are found, identification of each host's services can begin. This is done with port scanning or local port verification. The final step is identification of vulnerabilities, which is done by using vulnerability scanner or local banner grabbing. [26] Planning and reporting are also important parts of penetration testing. [25]

Penetration testing in an ICS has potential to affect the system and cause damage to it, and can lead to even deaths. For this reason, testing must be carefully considered. Penetration testing in the ICS network should only use actions that do not send traffic into an operational network or against an operating system. Port scanning or vulnerability scanning, if necessary, are done for a duplicate or a test system. Also, operations personnel must be aware of the testing that is occurring, and be prepared if any problems arise. [26]

# 5. MANAGING HARDENING THROUGH ICS LIFECYCLE

Industrial control systems typically have a long operational lifetime, which includes multiple changes and modifications of the system. For this reason, it is necessary to manage all security features through the control system's lifecycle. [8] The purpose of a hardening configuration is to increase security. In a static environment, changes in the hardening configuration are not needed. Due to differences in new software versions, the hardening configuration might need to be changed when the system is patched. System patches or users can also cause unauthorized configuration changes. These changes are driving the need for managing the hardening configuration. This chapter focuses on managing the hardening configuration through the ICS lifecycle. The idea for a lifecycle approach is based on the Finnish KYBER-TEO-project [2]. Hardening tasks are based on reviewing current installation and hardening practices, and guidelines of Department of Homeland Security: Cyber Security Procurement Language for Control Systems [15]. Security-focused configuration management is also greatly involved.

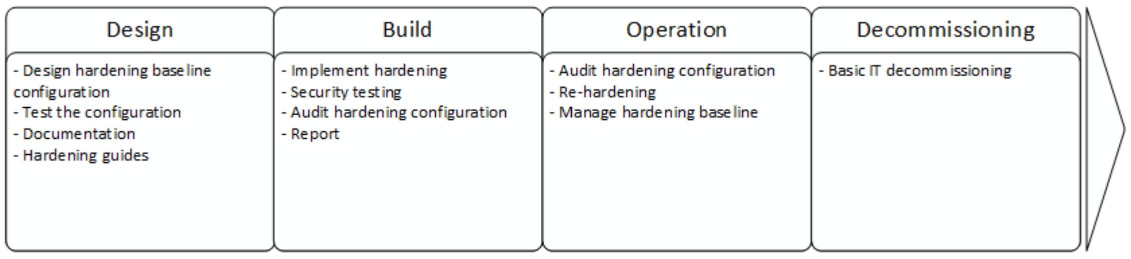
Design
- Design hardening baseline configuration
- Test the configuration
- Documentation
- Hardening guides
Build
- Implement hardening
- Security testing
- Audit hardening configuration
- Report
Operation
- Audit hardening configuration
- Re-hardening
- Manage hardening baseline
Decommissioning
- Basic IT decommissioning

***Figure** 7. Lifecycle management of the hardening configuration.*

Hardening contains multiple tasks in different phases of an industrial control system. This thesis uses 4 lifecycle phases presented in section 2.2. These are design, build, operation and decommissioning phases. Each phase includes special tasks for managing the hardening configuration. These tasks are summarized in Figure 7. In the design phase, a hardening template is created for each device in the system, including documentation such as hardening guides. In the build phase, a hardening template is created for each device in the system. Also, documentation and hardening guides are made. The build phase consists of implementing, testing and auditing measures for the hardening configuration. Testing and audits are done at least twice in the build phase, first in factory acceptance testing (FAT) and then in site acceptance testing (SAT), before the system is handed over to the customer. The operation phase can last decades and contain multiple changes in the system. This means that, for example, if extended patch support in Windows is promised to last 10 years, new version is installed at least once or twice in the ICS lifetime. Also, security patches are more frequently making smaller changes in operating systems. To maintain hardening configuration, patches should be tested in a test environment before applying into the production environment. Also, configuration audits must be done after patches are applied. The last phase is decommissioning which does not set any special requirements for system hardening.

## 5.1 The design phase

Defining and designing an ICS takes place in the design phase. From a security perspective, this phase is very important because the security choices in this phase will have huge effects in later phases. Integration of security should be considered early in the system development to keep cost-effectiveness high [27]. Adding security after the system is built, can be much more expensive and difficult. [8] Security tasks in this phase include conducting risk assessment, analyzing security requirements, performing security testing, preparing initial documents, and designing the security architecture. [28]

While the security-focused configuration management model includes both organizational and system level tasks of planning and identifying, hardening management in the design phase of the ICS lifecycle only consists of system related tasks. Organizational level tasks, such as creating organizational policies and procedures, are done before the ICS lifecycle has begun. Figure 8 shows a flow chart of hardening configuration development in the ICS design phase. The first task in managing operating system hardening is to collect system-specific configuration related information. This means identifying components, operating systems, drivers and programs used in the designed system. However, security should be considered already when designing the system, because OS and program design choices will have an impact on hardening and patch management, and therefore, the overall security. [21] Hardware and software choices must be made future support in mind, because these are usually difficult to change later. [15]

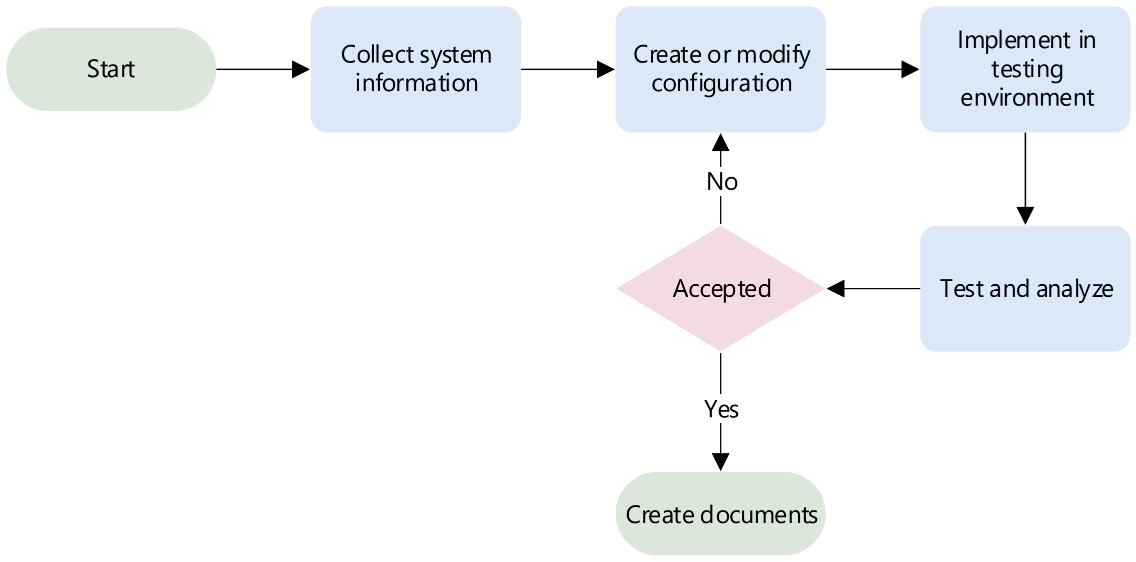
Start
Collect system information
Create or modify configuration
Implement in testing environment
Test and analyze
Accepted
No
Yes
Create documents

***Figure 8.*** *Hardening configuration development.*

When there is enough information about the system, it is possible to establish a hardening baseline configuration for each device. The configuration can be based on organizational policies, product developer's baselines or other security guides, such as security technical implementation guides (STIGs) or national checklists [21]. However, those guidelines are targeted for an ordinary ICT environment and might not deliver the best security for an ICS. A poorly designed hardening configuration can cause a lot of negative effects for ICS software and protocols. For this reason, testing the configuration is extremely important before implementing hardening configuration into a device in the operation environment. Configuration testing can be done on testing or development environment and it should include both functional testing and security testing.

There are two main objectives in testing when creating hardening configuration. The purpose for functional testing is that hardening would not have an impact on running automation software. The second objective is to validate the security with security testing. In industrial control systems, the first one is more important, because the main idea of hardening is to guarantee the reliability, not to compromise it. Security testing will help to find flaws in the hardening configuration, which can help the development of hardening. If testing reveals issues in hardening, they must be resolved or at least documented before the configuration is approved.

The last hardening task in the design phase is to create a final documentation of hardening configurations in different devices. Proper installation and hardening guides are also needed to provide guidance for the installation process and to provide security information for the customer.

## 5.2 The build phase

The building phase of the control system has two parts. The first is an implementation part and the second is the delivery and installation of the system to the customer's facility [10]. Security reviews should be performed through the build phase, for example, by including it with health and safety checks. Testing and assessing the systems security is an important part of both implementation and installation parts. Security testing in an operational ICS environment is difficult or impossible. Testing can also find unexpected vulnerabilities, which are important to find at the early stages of the lifecycle [8]. In addition, two main assurance gates, which are factory acceptance testing (FAT) and site acceptance testing (SAT), validate the system before delivery or commissioning is approved.

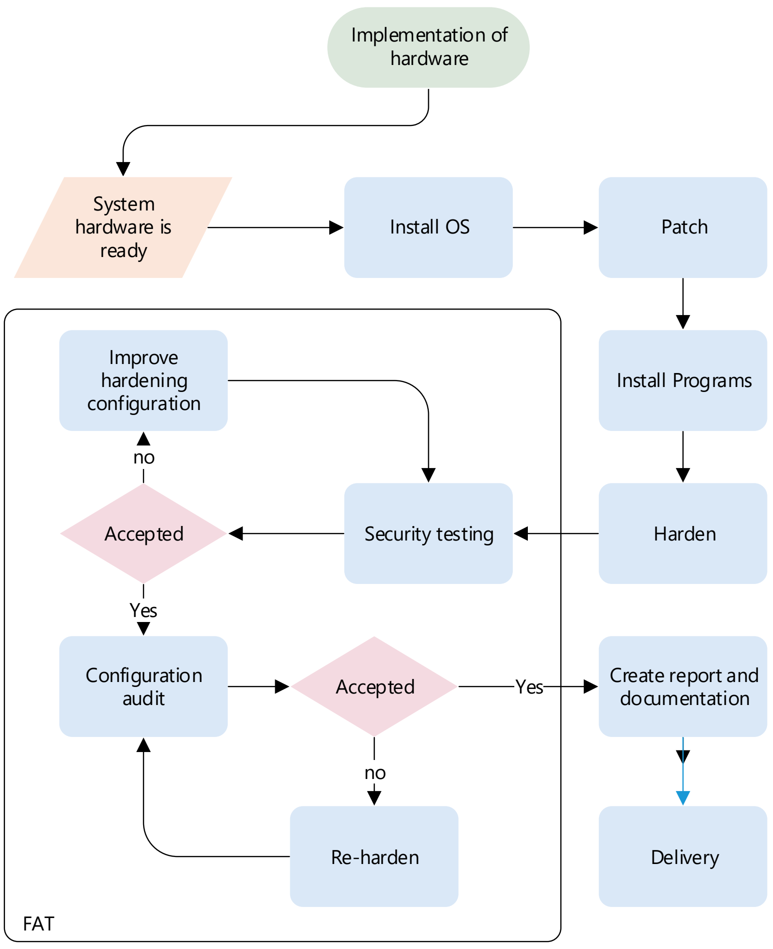

***Figure 9.*** *The build phase: implementation and FAT.*

For OS hardening, implementation means installing and configuring operating systems. This can begin when the development environment hardware is ready, which is shown in the build phase flow chart in Figure 9. The first task is installing the operating system with the help of installation guides or by using correctly installed OS images, and installing the latest tested operating system patches. After installation, the system can be hardened with the designed hardening baseline. Automated hardening scripts are more practical and a better way of hardening. Scripts reduce manual work, but also decrease misconfigurations. Installing necessary control system programs can be done after hardening, if the hardening script is not able to adapt to hardening exceptions of different programs. In this way, all necessary hardening exceptions need to be made when installing automation software. The implemented system is then carefully tested in factory acceptance testing (FAT) with functional and security testing. If need for security improvements or operational problems arise in testing, configuration must be changed and tested again. At the end of FAT, it is still necessary to audit the configuration against the most recent baseline, and create a report to represent that system is properly configured.

From a security perspective, factory acceptance testing (FAT) is necessary to verify that security features are functioning properly [15]. In FAT, the system is tested with the customer and documentation reviewed to match the implementation. Approved testing results lead to system delivery [10]. According to the Homeland Security Cyber Security Procurement language for control systems, there are several tasks for hardening in factory acceptance testing. These mainly include security scans and audits. Cyber security scans, at least vulnerability and active port scans, should be run for the control system. Results are then validated by comparing them with designed control system documents. The vendor shall also provide documentation of required network services and operating system services, necessary configuration parameters, and results of vulnerability scans. [15]

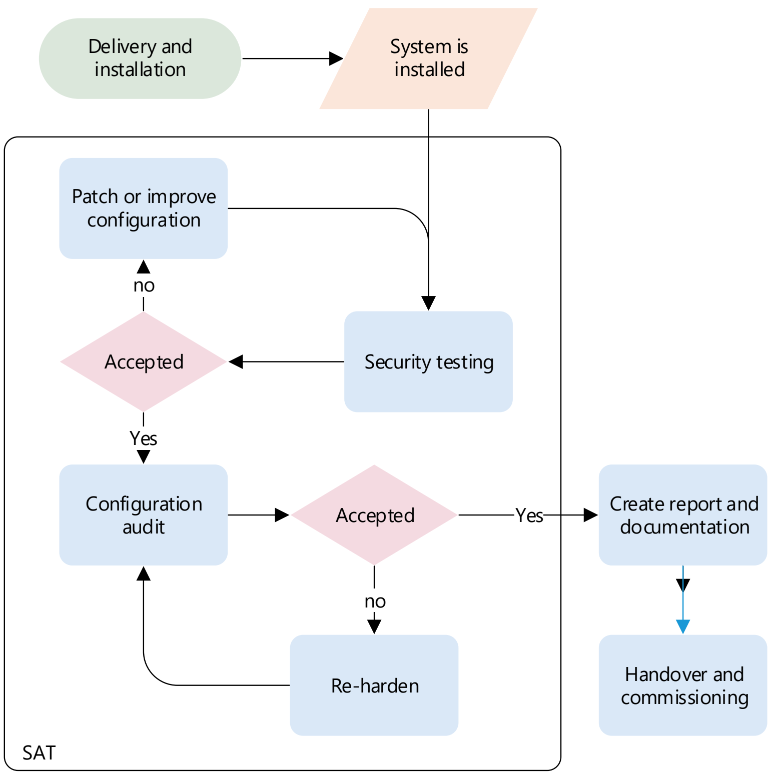
Delivery and installation
System is installed
Patch or improve configuration
no
Accepted
Security testing
Yes
Configuration audit
Accepted
Yes
Create report and documentation
no
Re-harden
Handover and commissioning
SAT

***Figure 10.*** *Site acceptance testing.*

Site acceptance testing is done after the system is delivered and installed to the customer's facility. It is done to demonstrate functionality of the system under its full load [10]. It typically repeats tests done in FAT, but does also add some integrated functions. From the perspective of hardening management, the flow chart for SAT in Figure 10 resembles the FAT chart in Figure 9. The reason for this is because SAT is done just before commissioning to validate that the installed system is equivalent to the system at the factory acceptance testing. To validate the hardening configuration, the vendor should compare results of cyber security scans with previous results and documentation. Security scanning should be done with the most recent signature files. [15] In the last part of SAT, a test report is written and approved in raw form or with a scheduled repair list. Errors or deviations can only be accepted if they do not have a direct or indirect impact in the systems performance [10]. This could be considered, for example, if new uncritical vulnerabilities were found in site acceptance testing. However, reporting those issues is important so they can be more easily repaired in the operation phase.

## 5.3 The operation phase

Usually the longest phase in the ICS lifecycle is the operation phase. Unlike the office information systems, the operation phase of an ICS can last decades after the system is deployed [10]. Effective management of ICS security risks must be continued throughout this phase, and provide security training for the operations staff, who control, operate and monitor the system. To maintain the systems security, certain security activities need to be managed through the operation phase. These activities include monitoring security event logs, managing vulnerabilities and pathing, change controls, and other operational security activities, such as maintaining access control. [8]

Security-focused configuration management phases that focused in the operation phase are controlling configuration changes and monitoring. Configuration changes can be system changes or system patches. Especially operating system patches might introduce changes to the hardening configuration, and therefore weakening security. The configuration should be audited after OS patching to verify the validity of the configuration. Some system patches can also require a new hardening baseline altogether, which increases the need for controlled configuration change management.

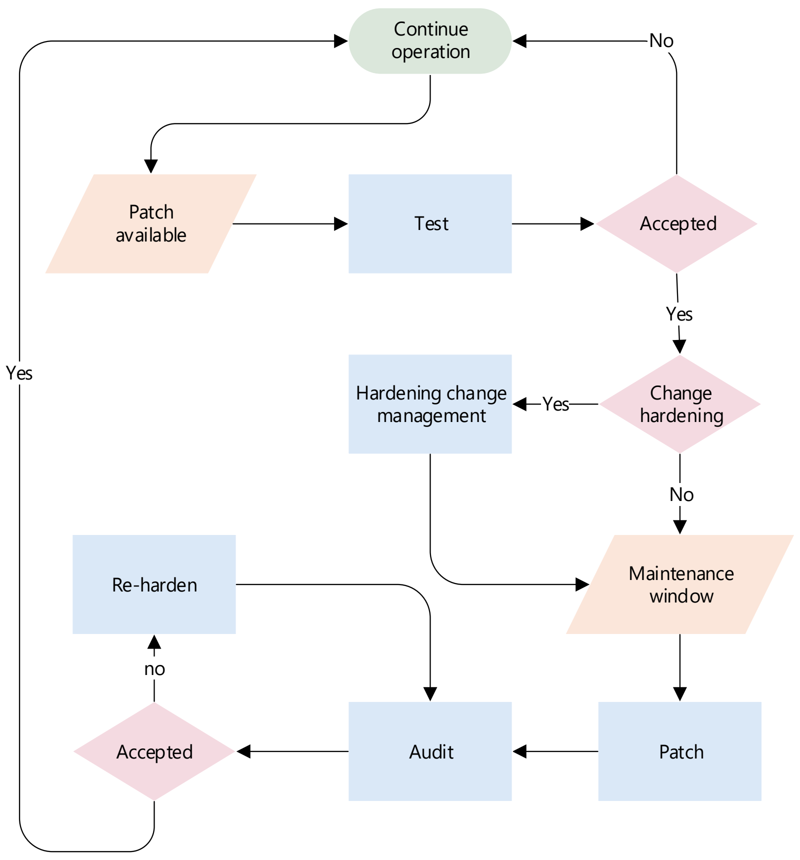

*Figure 11. The operation phase flow chart.*

Managing the process of hardening in the operation phase is a cycle that repeats through ICS operations. The operation cycle starts when the system is started after SAT and other testing. The flow chart in Figure 11 shows the management cycle. It indicates that while the system is in operation and a new patch available, it must first be tested in a test environment. This follows good patch management practices. [20] If the patch affects systems operations it cannot be imported into the production environment. Also, if the patch has such influence in the system, that a new hardening configuration is needed, the new hardening configuration should be made and implemented. The means to detect a need for hardening configuration change can vary from reading patch release information to executing security testing in the test environment. Hardening configuration changes need to be managed with change management practices, and the new configuration is designed like in Figure 8. Unexpected downtime could cause catastrophic failures in production environment, so patching can only be done in a maintenance window and with careful planning [20]. After the system is patched, the configuration is audited against the recent hardening baseline. If the configuration does not match the baseline, the system is re-hardened and audited again. When the configuration is valid, operation can be allowed to continue.

## 5.4 The decommissioning phase

The last phase of the ICS lifecycle is the decommissioning phase. Decommissioning and disposal of a system should be already considered at the design phase, because of possible environmental impacts and recycling measures required. This phase also includes storing the system information that is needed after the system is disposed [10]. The goal of this phase is to preserve necessary information and manage the disposal without leaking data outside. Key security activities include developing a disposal plan, archiving critical information, erasing media, and disposing the hardware and software.

Hardening management does not have any additional tasks in the decommissioning phase, but secure information system disposal practices should be used to secure the configuration and other information [2]. Decommissioning of an ICS should follow the configuration management process and be treated as a configuration change project, in same manner as any system configuration change. The decommissioning phase might introduce new security threats caused by changes in the system that undermines security controls. Secure data destruction should contain all sensitive information, such as names, addresses, passwords and other account information, product information, technical data, and customer details [8].

# 6. THE TARGET ENVIRONMENT

To test the lifecycle model of hardening management, it must be implemented into a testing environment. The target environment for the implementation is a distillation column control system found in Tampere University of Technology. This environment is used for research and teaching purposes. To make hardening management possible, new scripts designed for hardening and configuration auditing, are needed. This chapter introduces the control system environment and operating systems used in hosts. Also, OS features related to hardening and configuration management are presented.

## 6.1 The distillation column network

The distillation column control system operates with the Valmet DNA automation system. The automation system consists of a local area network and computers used for multiple purposes such as operation, alarms, engineering and process control. In automation systems, these are called servers or stations. Automation software is divided into different roles, such as an engineering server and an operation server. In the distillation column, almost every station has only one role, but a station may also have more than one role. The control system network of the distillation column is illustrated in Figure 12, where the network is divided into two segments, DMZ and ICS networks. In this environment the DMZ segment, and part of the ICS network and devices, are running in a virtualized environment. A virtual firewall is used between ICS, DMZ, and office networks to limit connections between the network segments. The virtualization environment uses the VMware ESXi bare-metal hypervisor. Using virtualization allows easier device management and using snapshots to save the state of a device allows the device to go easily back to the snapshotted state if anything goes wrong.

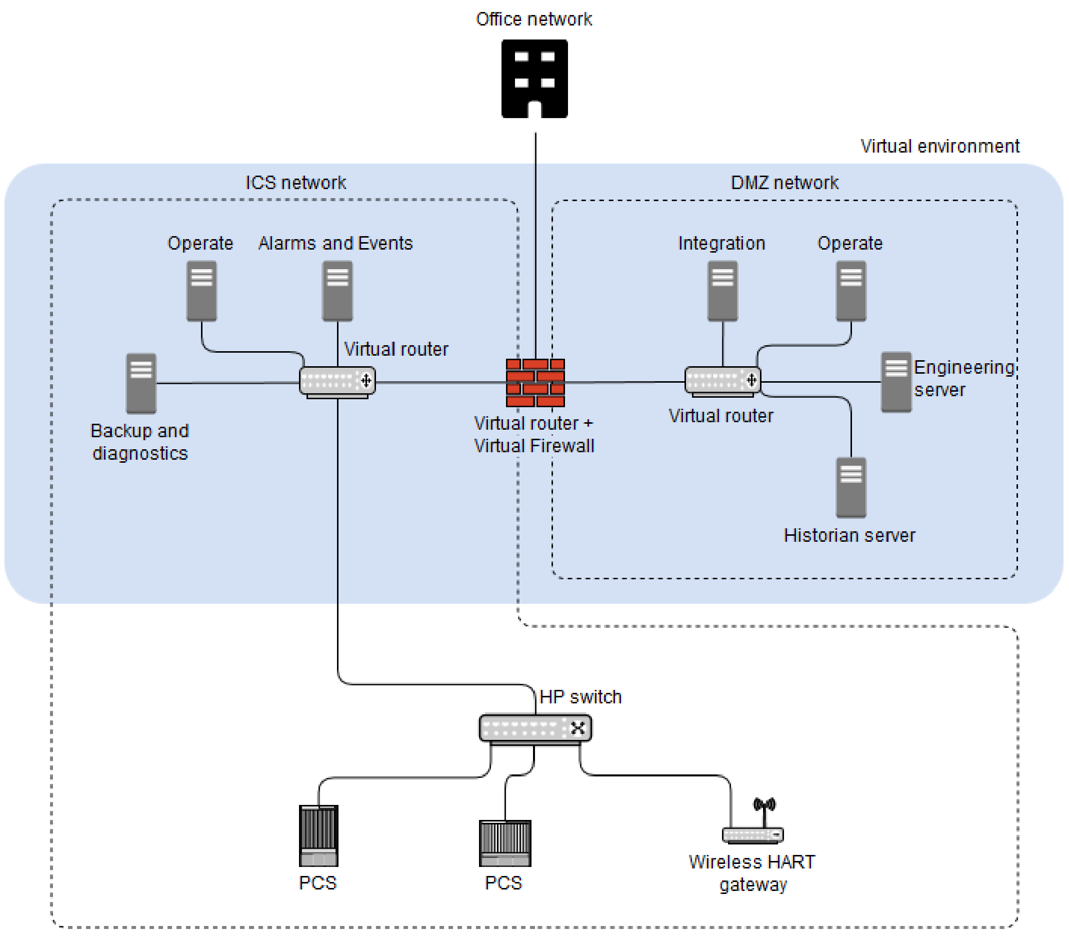

***Figure 12.*** *The distillation column ICS network.*

The hosts that are in focus of this thesis are those seven stations running in the virtualization environment in Figure 12. Virtualization does not seem to have any special OS hardening requirements, but when considering the whole system, it adds the need for hardening the virtual environment. The ICS network segment has three stations, which are running software for one control system role. These are the alarms and events server, the operation server and the backup and diagnostics server. The DMZ network segment includes four stations: The OPC integration server, the engineering server, the historian server and the other operation server, which also includes another alarms and events server. Each station has its own control system role which has a unique set of features and programs. Additionally, there are features, such as a printer client enabled in the operation server in the DMZ, and a print server in the historian server. These stations are currently using two different operating systems: Windows 7 and Windows Server 2008. However, these are planned to be updated to Windows 10 and Windows Server 2012 R2 operating systems during this thesis. For this reason, these operating systems were used for the development of hardening and auditing scripts in this thesis. The HP switch connects Valmet ACN process control stations to the ICS network. Those control stations are not using Windows operating systems and are not in scope of this thesis.

Because each operating system has at least partly different features, services and included automation software, each one also has its own OS hardening configuration baseline. This baseline should be as restrictive as it can without causing harm for the OS operations, but it can affect other software used in the station. To take this into account, some programs require adding exceptions to the configuration baseline. These are parameters that must not be hardened to ensure the operation. This means that, depending on the software used in the different stations, there are as many hardening baselines as there are different types of stations.

## 6.2 Windows operating systems

Windows is an operating system developed by Microsoft Corporation. Currently there are Windows versions for desktop, server and mobile use. [29] Windows server versions include memory, networking, and other capabilities needed in a server environment. Windows server operating systems also include server manager software for easier management, and they have multiple server roles and features included. Microsoft Windows Server versions are structured very similar to desktop operating systems. This means that the Windows Server 2008 R2 OS is closely reflecting Windows 7, Windows Server 2012 is similar to Windows 8, and Server 2012 R2 has a lot in common with Windows 8.1. The latest server version, Windows Server 2016 shares features with Windows 10. [30] This leads to the fact that the operating systems, Windows Server 2012 R2 and Windows 10, used in this thesis do not share as much similarity as Windows 10 and Server 2016 systems would. However, there are still a lot of common features in these versions, and both can use the latest PowerShell version [31].

From a hardening perspective, both client and server Windows operating systems include many features that must be disabled or removed in an ICS environment. One example is the Windows store that appeared in Windows 8 and is also found in later Windows client and server versions. The Windows Server 2016 OS even comes with two Xbox live related services if the desktop experience is installed. These are recommended to be disabled even by the Microsoft Security Guidance blog [32]. Those services are also found in Windows 10. The other significant concern of using Windows 10 in an ICS, could be the telemetry features found in Windows 10. This means that Microsoft is collecting information from the system for development purposes. [33] In ICSs, the hosts do not have an internet connection, so trying to collect and send information to Microsoft unnecessarily uses host's resources and adds unwanted network traffic in the ICS network.

A new way to build, deploy, and service Windows has been introduced in Windows 10. This is called Windows as a service. The purpose for this is to remove the need for new Windows versions in every few years, and to add features to Windows 10 with updates. New features are delivered to the Windows Insider community as soon as possible for testing purposes and to get feedback. Windows servicing now includes two types of updates instead of major revisions, service packs and monthly updates. These new types are feature updates, which add new functionalities twice a year, and quality updates for security and reliability fixes. Quality updates are released at least once a month. For special systems, there is the Long-Term Servicing Channel, which is designed to be used only with special devices that typically perform a single important task, such as those controlling medical equipment. These devices will get new feature releases only in about every three years, because it is more important to keep these devices stable, than up to date with user interface changes. [34] Windows Server 2016 is provided with two servicing models. Servers that install the Server with Desktop Experience or the Server Core versions use the Long-Term Servicing Branch that is also used in previous Windows Server operating systems. This provides five years of mainstream support and five years of extended support. However, Windows Server 2016 Nano servers use the Current Branch for Business servicing model, which is much like the Windows 10 experience. [35]

### 6.2.1 Windows AppLocker

Although there are many third-party application whitelisting tools, Windows AppLocker is viewed here for two reasons. Firstly, it is included in both Windows versions used in this thesis, though Windows 10 needs to be enterprise or education version. The second reason is that there is a PowerShell module for configuring AppLocker restrictions, so it could be added to scripts developed in this thesis. [36] [37]

Windows AppLocker was introduced in Windows 7, but with limitations. All current features in AppLocker have been available since the Windows 8, at least in enterprise and server Windows versions. [37] AppLocker can be managed with three management interfaces: Microsoft Management Console snap-in, Group Policy Management and Windows PowerShell. With AppLocker, it is possible to separately set rules for executables, Windows installers, scripts and packaged apps. Without allow rules, AppLocker restricts all applications or other supported file formats. AppLocker can also be used in audit mode, where it does not restrict any applications, but audits application usage. This can be helpful for designing AppLocker ruleset. Execution rules can be set by a path, a hash or a publisher which can be extracted from the signing certificate. [36] [38]

### 6.2.2 Windows Firewall

Windows Firewall is a host-based firewall included in Windows operating systems. Earlier versions of Windows Firewall, for example in Windows XP, were not very good, but this has been improved in later versions and Windows Firewall with Advanced Security is now a quite capable security tool. [30] [38]

Windows Firewall has two different management consoles. A simplified console is found in the Control Panel from the System and Security menu. This menu does not give advanced management, but at least it has some simple information and functions. The Windows Firewall with Advanced Security console can be opened from the previous firewall menu by clicking the Advanced settings link. This includes a lot more management for configuring firewall settings. The firewall can also be configured through the Group Policy or with PowerShell. Windows Firewall has three different firewall profiles. The domain profile is used if a domain-joined computer is connected to network where the domain controller is accessible. The private profile is set when the network is chosen as a “Home” or “Work” network. Finally, the public profile is used when the network is chosen as “Public”. Any network interface card (NIC) connected is assigned with one of these profiles. [30] [38] [39]

### 6.2.3 Windows PowerShell

Microsoft PowerShell is a command-line shell and an automating tool first created for Microsoft Windows operating systems. It was released in 2006 integrated in Windows 7 and Server 2008 R2, but it is also available for a few earlier Windows versions [40]. Since then, PowerShell has been in every desktop and server versions of Windows, and it has reached the version 5.1. PowerShell has also become an open source project and is now available for Linux, UNIX, and MacOS systems [31]. The purpose of PowerShell is providing powerful tools for administrators to manage and automate operations in Windows. The idea is that Microsoft would create GUI applications, which are using PowerShell commands, so that everything in the operating system could be done with PowerShell. [40]

PowerShell is used with shell commands running small .NET programs called commandlets, which can be used manually by typing commands into the command line shell, or automated with scrips. Commands can be run locally or remotely with the Windows Remote Management (WinRM) feature [41]. An easy way to write scripts is to use PowerShell ISE (Integrated Scripting Environment). It comes integrated in PowerShell and consists of a script editor, a console window, and other features for writing scripts. Third-party programs designed for PS scripting are also available. More PowerShell commands can be added with Snap-ins or, in version 2 or later, with modules. There is a huge number of modules available from Microsoft and third-party developers, or you can create your own modules. [40]

Windows PowerShell can be quite an effective tool for hardening, especially in a Windows environment. Settings in Windows can be configured easily with PS even though Microsoft doesn’t provide commands or modules for everything, at least not yet. However, it seems that there is still development of PowerShell going on in Microsoft. During this thesis, the version 5.1 of PS was released containing some useful new commands, such as tools for local account management.

### 6.2.4 Windows security baseline

Windows security baselines are provided by Microsoft to help customers with security configurations. Constantly evolving security threat landscape requires that IT professionals and policy makers need to keep up with security threats, and make required configuration changes to reduce the threats. Windows 10 includes over 3000 Group Policy settings and over 1800 Internet Explorer settings, but not all of them are security-related. Managing this much configuration parameters can be difficult without help from Microsoft. Windows security baselines contain a group of configuration settings that are recommended by Microsoft. Configuration is based on feedback from Microsoft security engineering teams, product groups, partners and customers. Different baselines can be found for each different Windows version, and even for different Windows 10 build versions. [42] [43]

Windows security baselines can be used with Microsoft Security Compliance Toolkit, which include tools for downloading, analyzing, testing, editing and storing Microsoft-recommended security configuration baselines [43]. Security Compliance Toolkit replaces older and more complex Security Compliance Manager. Unlike its predecessor it includes Desired State Configuration (DSC) tools, but some tools that were in Security Compliance Manager are still missing. [44]

### 6.2.5 Windows Server Update Services (WSUS)

Windows Server Update Services (WSUS) provides more control over Windows updates. For example, what updates are delivered and when. This can be very important especially in control systems, where patches must be implemented only during maintenance windows. WSUS is natively available in Windows Server operating systems, but it can be used with all Windows versions that are used in this thesis. [45]

WSUS enables more management of distribution of the latest Microsoft product updates for administrators. Updates are delivered to target computers from a WSUS server. This is managed through a management console. The WSUS server can also provide updates for other WSUS servers in the organization, but at least one server must be able to connect to the Microsoft Update service to get the latest available update information. [46]

## 6.3 Securing PowerShell

Because the PowerShell can be a quite powerful tool for administration, its security should be carefully considered. Security features in PowerShell mainly focuses on preventing a user to run commands and scripts unintentionally. It must also be noted that PowerShell commands and scripts have only the same permissions than the user executing them. Security can be increased with script restrictions and logging. Script restrictions are done with a script execution policy. There are 5 different execution policy options.

The default option is *Restricted*, which does not allow any script execution. The *AllSigned* option allows only digitally signed scripts to run. The *RemoteSigned* allows executing unsigned scripts locally, but remote scripts must be signed. The *Unrestricted* option allows all scripts to run much like *Bypass,* which is special setting intended for application developer use. If *Bypass* is used, execution policy is bypassed altogether. Additionally, there is also the *Undefined* option which resets the policy to default. The PowerShell execution policy can be configured through the Group Policy for systems in an Active Directory environment. PowerShell remoting uses Windows Remote Management (WinRM) implementation which includes authentication and encryption features. The WinRM feature can be disabled, but it is enabled at least in Windows Server 2012 by default. [40]

Securing PowerShell completely is difficult, so logging all activities is important. First PowerShell versions provided only few useful audit settings, but in PowerShell version 3.0, this shortcoming was addressed [47]. Later versions, especially version 5.0, have also increased new superior logging capabilities. These logging capabilities include engine lifecycle logging, module/pipeline logging, script block tracking and transcripting. Engine lifecycle logging is enabled by default, and includes logging of startup and termination of PowerShell hosts and history of command-line arguments, including code passed to PowerShell.exe. Module or pipeline logging has existed from version 3.0 and contains logging of pipeline events into Windows Event Logs. Script block tracking logs information about all code that was run. This information is stored into Windows Event Logs. PowerShell 5.0 also allows logging of all PowerShell commands used and the outputs they generated. This logging method is called by the term transcription. Transcription data is outputted into a file on the disk and can be difficult to parse. Logging policies can be configured locally or via Group Policy. [41]

## 6.4 Configuration management in Windows

Because this thesis focuses on managing hardening in Windows operating systems, it is important to know some usual configuration management tools used in Windows. Most settings in Windows are designed to be manually configured with a graphical interface, but there are ways to make configuration easier, both locally or remotely, for multiple computer devices at once. This section introduces two methods for Windows configuration management to give some basic knowledge on how Windows systems are managed.

### 6.4.1 Group Policy

Group Policy provides means to configure computer and user settings in Active Directory (AD) Domain Services based networks. This requires at least one AD server installed in the network [48]. Computers that are managed also need to be joined in the domain, and users whose configurations are managed, must use domain credentials in those computers. Without Active Directory available, there is only one Group Policy available. This is called the *local policy.* Group Policy settings are managed with Group Policy Objects. These objects contain two halves, which are the user half and the computer half. Each half contains policy settings in hierarchical structure, which groups similar settings under different branches. Group policy includes password, audit, group, AppLocker, Firewall settings, among others. [49]

Using the Group Policy locally is done with the Local Group Policy Editor. This is a graphical editor where policies can be explored and set to desired values. In an Active Directory environment, it is possible to remotely configure multiple user and computer configurations in domain by creating Group Policy objects using the Group Policy Management Console. [49]

### 6.4.2 PowerShell DSC

PowerShell DSC, or Desired State Configuration, was introduced in Windows Management Framework 4.0. The base idea in DSC is to create specially formatted text files to describe how s system should be configured. These are called MOF (Managed Object Format) files. These files are deployed to target computers, and configurations are implemented automatically with PowerShell. MOF is a cross-platform standard defined by Distributed Management Task Force. This means that MOF files can also be created with other software than PowerShell, and PowerShell made MOF files can be used for configuring non-Windows computers. [50]

There are two methods for delivering MOF files into target machines. The push method uses Windows Remote Management (WinRM) protocol to push files to their destination computers. The pull method, on the other hand, works more like Group Policy, though without a domain controller. The pull method can also be configured to use the Server Message Blocks (SMB) protocol to pull MOF files from the server. Once the file is delivered to the computer (only one MOF file is allowed in one device), the configuration is read and implemented with DSC resources. A MOF file is divided into sections. Each section uses a DSC resource for implementing the configuration. Each section in the configuration file has its own DSC resource, which is used to implement the configuration parameters. Microsoft provides resources for many useful configuration items, such as registry settings and Windows features. More resources can also be implemented, and with script resource, it is possible to run custom scripts. PowerShell DSC can also be used to monitor that devices will remain in their desired configurations. [30] [50]

## 6.5 Hardening Windows operating systems

Windows operating systems are designed for a great number of purposes. For this reason, they include a lot of possible hardening objects. While developing an all new hardening configuration was not done in this thesis, a lot of examination of possible Windows hardening parameters was done. Some security testing and AppLocker auditing was also done to review the current level of hardening. This section presents some of those parameters which includes a few settings and removable features that should be considered to be removed when hardening Windows operating systems. Operating systems in focus are Windows 10 and Windows Server 2012 R2 that were used for the script implementation in this thesis. The following hardening objects do not give a fully comprehensive hardening configuration. The section only gives some ideas of what to look for when hardening Windows operating systems, and how hardening could be improved. Unlike in a traditional ICT environment, restrictions in an ICS environment can prevent classic ways to implement the hardening configuration. Remote connections can be very limited and internet connection is not available. For these reasons, the configuration is usually implemented locally. Configuring manually all required parameters is time consuming and so the hardening is done automatically with scripts.

Windows 10 is quite aggressive in collecting telemetry information and error reporting. Having these features in system that does not have an internet connection is quite pointless. Disabling all settings, services and scheduled tasks related to these, can be recommended for all Windows devices in ICSs. Even if this would not increase the security very much, it can decrease unnecessary network traffic. Other Windows features that require an internet connection can also be removed in most cases. These include the Windows Store, social features, such as Messaging or Skype, OneDrive, an e-mail program, the Weather application and other features depending on the Windows version. In addition to these there are games and gaming related services and tasks running in Windows 10, some of which are found even in server versions of Windows. These include Microsoft Solitaire Collection, Xbox live services and tasks, which should be removed, and disabling the Windows 10 gaming mode. After that, there are still media content and other programs that should be removed. The server versions are a little better in this way, because server computers rarely need those features, and so Microsoft does not include them. However, this has changed a little bit with the latest Server 2016 which seems to include two Xbox live related services if the desktop experience is installed. These are recommended to be disabled even by Microsoft Security Guidance blog [32]. Additionally, the local group policy should be configured in a secure way. The Windows secure baseline configuration can be most helpful in this part of hardening. It has a list of policy objects with descriptions, suggested values of these objects, and registry keys. [51] The baseline includes settings for user accounts, audit and logging, security options, firewall and a lot of other settings.

Microsoft seems to have recognized the need for a more customizable Windows OS for computers running specialized tasks, such as many devices in manufacturing, healthcare or governmental use. For this kind of usage Microsoft provides Windows 10 IOT Enterprise version, which includes locking down capabilities. This means that unnecessary system features can be removed from the installation to run applications and specialized functions in a secure, reliable and streamlined way on mission critical systems. [52] However, even removing all unnecessary features from the installation does not mean that those features can be left out from hardening and auditing. At least auditing the features must still be done to notice if a feature is restored, for example, with Windows updates.

Increasing hardening further than just using available guides and baselines can be difficult. One possible method to find additional hardening objects in Windows operating systems could be using Windows AppLocker in audit mode to reveal programs and scripts that start up in the background. This was used during this thesis, and some processes were found that seemed unnecessary, but with given time it was not possible to do enough functional testing to add these into the hardening baseline. Some security testing, such as network monitoring and vulnerability scanning, was also done to find out what it could reveal about hardening. With network monitoring, it was found that unhardened Windows 10 tries frequently to connect to Microsoft servers, and this was reduced with hardening. Windows Server 2012 R2 on the other hand, did not cause any unwanted traffic in one-hour testing, even when it was not hardened. Vulnerability scanning on the other hand revealed that the tested device needed system patches. No additional removable features were found in this case.

# 7. HARDENING AND AUDITING SCRIPTS

Hardening management includes hardening and auditing tasks in multiple phases of the ICS lifecycle. These tasks must be performed for all host devices, and can include setting and verifying possibly hundreds of settings in each device. This kind of work done manually is slow and change of misconfigurations is great, so it is an ideal job for using scripts. By using scripts, hardening or auditing can be done in seconds, and more consistently. This chapter focuses on the implementation of hardening and auditing scripts that are needed for hardening management. The implementation is presented by describing requirements, the hardening template, and the scripts. In addition, difficulties, testing, and possible future development are also considered in this chapter.

Because the focus of this thesis was on Windows systems, the scripts created for this thesis are made for Windows hosts in mind. Windows operating systems are very common in ICSs. However, there are many other types of host devices in control systems, such as network firewalls, routers, and Linux computers, that must also be hardened and audited. The scripts are done using Microsoft PowerShell, which comes included in Windows 7 and later desktop versions, and Windows Server 2008 and later server versions. PowerShell can also be installed into Windows XP and Windows Server 2003 operating systems. The reason for choosing PowerShell was, because it was included in all Windows operating systems used in the distillation column. Microsoft is also recommending it for configuration management. It is a powerful tool especially in later Windows versions and possibly even more powerful in the future thanks to the support from Microsoft, which is also important.

Some ICS programs had resource needs that conflicted with the hardening baseline. Conflicts are solved by adding setting exceptions to the baseline configuration. This means that depending on installed programs, each host has its own hardening baseline. The scripts must be able to adapt to all baselines if they are used for re-hardening or auditing. The idea how this could be done was to create a new hardening template, which could include hardening baselines for each used operating system and exceptions that different programs require. The scripts could then use this information to create a device specific hardening baseline. Information about the operating system and installed programs of each host can be requested from the user, or the scripts can try to access this information from the host computer.

## 7.1 Requirements

At the beginning, some requirements were set for the hardening scripts. The requirements were set partly by nature of the ICS environment, and partly by security reasons. Additionally, there were requirements to make the management of hardening a little easier. The first environment related requirement was that no internet access would be needed. This meant that everything the scripts needed had to be included in Windows, or could be easily delivered with the scripts. Furthermore, additional software shouldn't be necessary to be installed. Partly for the environmental and partly for the security reasons, the scripts needed to be only locally used. The last environmental requirement was that scripts needed to be able to manage hardening and auditing in different devices with different programs and with different Windows versions. The idea was to automatically identify what configuration baseline to use, depending on the Windows version and the installed software.

The hardening template, that was intended to manage all configuration baselines, needed to be extensible and easy to read and manage manually or with the scripts. Expandability was required because new OS baselines and hardening exceptions must be added in the future. The reason for the template to be manually used is that while the hardening template is designed for scripts to use, updating new baselines and exceptions is done manually.

## 7.2 The hardening template

The hardening template is basically a list of all hardening configuration parameters in a useful format. Because of the requirements for the hardening template, two options were considered in this thesis. The first option was CSV (Comma Separated Values) which can be edited with Microsoft Excel and used with PowerShell scripts. The second was XML (Extensible Markup Language) which has editors like Microsoft XML Notepad to allow easy manual management, and which is ideal for automatic data processing. XML was chosen for providing not only flexibility and expandability, but also some structure for the template. The tree structure of XML makes reading the template easier, which is an important aspect for the template, because it is supposed to be mainly manually configured.

The structure of the hardening template is divided into two main branches. The first branch is the *WindowsTemplates* element, which includes the baseline hardening configuration for each operating system. The second branch is for different control system roles and features, that include all the hardening exceptions for each feature. Figure 13 illustrates the structure of the template with two operating system configurations included. One for Windows 10 and the other for Windows Server 2012 R2. Operating system baseline configurations include different categories for different setting types. These are included to make management easier. The scripts would work even if all the settings were located under the same category element. Each *Setting* element has three attributes: *Type*, *Name* and *Value*. The *Type* attribute identifies the type of the setting. This is needed, because there are a lot of ways how to manage settings in Windows. Every type has a set function and a get function in the scripts used for configuring and auditing. The *Name* parameter identifies the setting. It must be unique in the same types of settings, but not in all settings. The last parameter *Value* defines the value of the setting. The scripts are included with functions for configuring the following setting types: Windows security policies, services, file associations, NetBIOS, Network adapter bindings, and scheduled tasks.

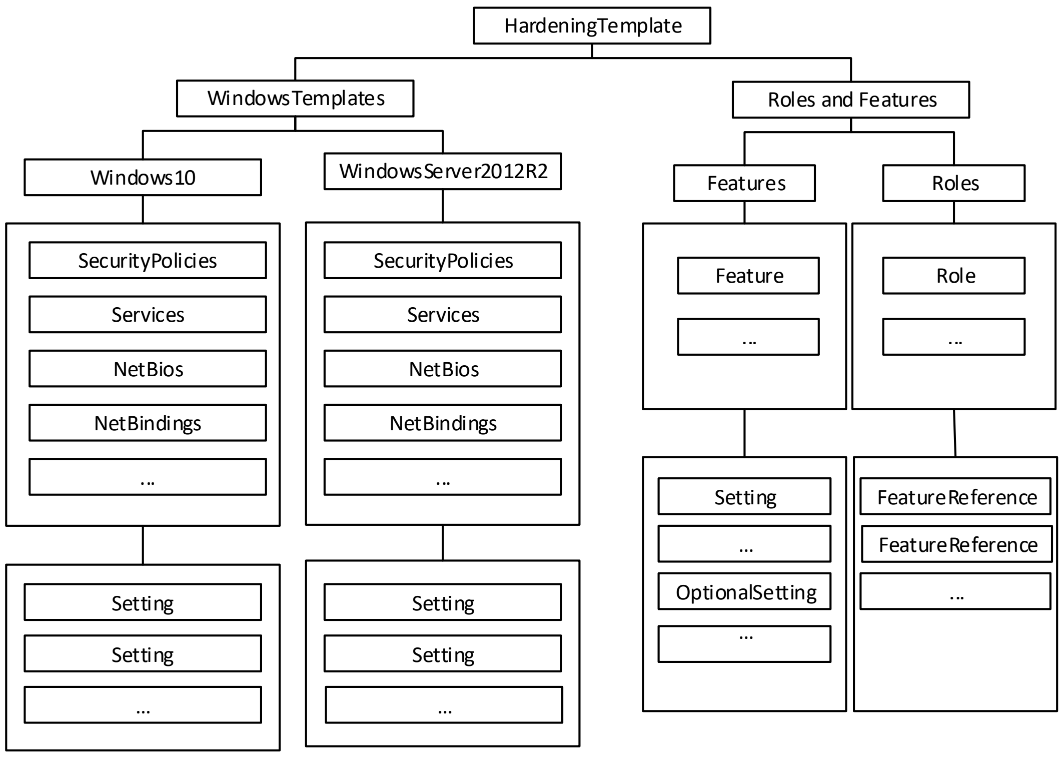

***Figure 13.*** *The structure of the hardening template.*

The control system consists of different roles, such as an engineering server and an operation server. These roles are combinations of different features, which are the required programs. The Roles and Features XML element combines information about all the available roles and features. The control system roles are found under the *Roles* element and features in the *Features* element. Each *Role* element includes a unique *Name* attribute and a list of feature references to indicate features that the role has. The *FeatureReference* elements only have a *Name* attribute to indicate a feature, and no child elements. All features, programs and other functions, that can be installed in a control system device, are included under *Feature* elements. A *Feature* element is identified by a *Name* attribute, and it can include multiple *Setting* elements. These settings are the exceptions that the feature needs. *Setting* elements in features are similar to S*etting* elements in Windows templates. A *Feature* element can also have an *OptionalSettings* element which consists of settings that are optional for the feature. The Roles and features element is structured like this because roles can have common features. Also, this way if a feature setting is changed it must only be done in one place. Another solution could have been that the feature elements include which roles they belong to. Then the role elements could be removed. This might be a bit easier for designing the scripts, but it makes the template harder to manage, for example, if its needed to check what features a role has, or features of a role change.

The *HardeningTemplate* element includes a version attribute to give some version control. This attribute could be then verified with the scripts to ensure that no unsuitable template version is used with the scripts. However, this feature is not included in the scripts developed in this thesis.

## 7.3 Hardening and auditing scripts

Two scripts were made in this thesis. A hardening script, which hardens a computer with a given hardening template, and an auditing script, which can compare a current configuration of the system against a hardening baseline in the template. These scripts use multiple common functions, especially functions related to using of the hardening template. The scripts could have been integrated easily into one, but were kept separate to simplify the use. The use cases for the scripts are found in Figure 14. The scripts are used in a very similar way. Both ask for the Windows version, the automation role, and the path of the hardening template. This information and the hardening template is then used to create a machine specific hardening baseline. The baseline is then used for either hardening or auditing depending on the script.

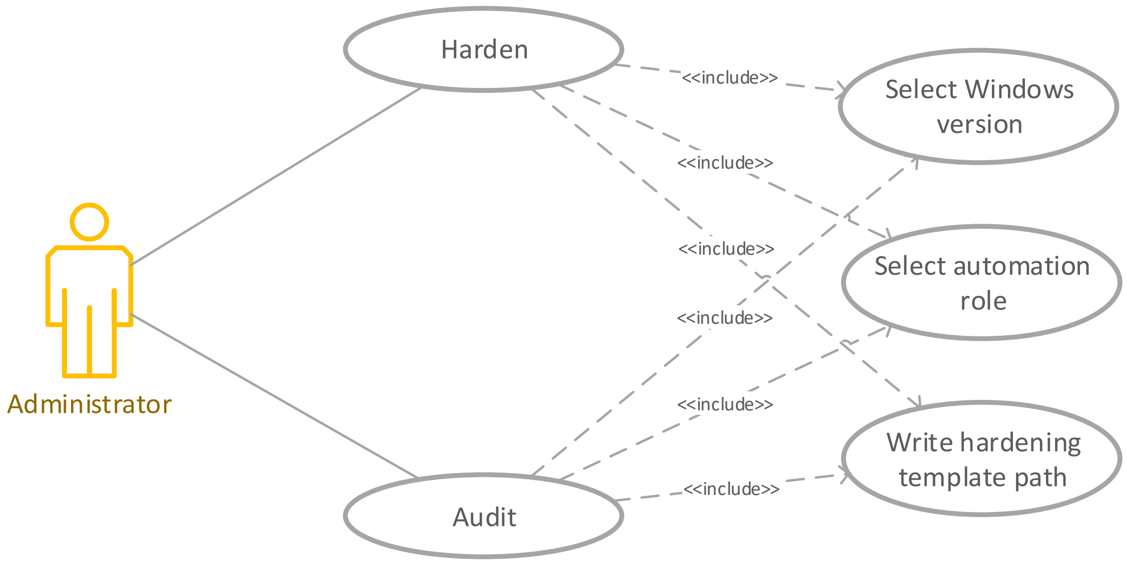

***Figure 14.*** *Use cases for the scripts.*

Two things define the hardening baseline of a system. The operating system and all the control system features installed in the device. Selecting the operating system, or in this case the right Windows version, was simple. The script generates a list of available Windows templates from the hardening template. These options are then presented for the user. An option for automatic selection is also included. This option will check the operating system and select the right Windows template. If a matching template is not found, user must select again. The feature selection was supposed to be able to accept multiple roles and to have an option for the user to make a new combination of features from features available in the template. Additionally, an automatic option which could detect the installed features was considered. However, at the current stage the scripts are limited only to accept one predetermined role. No automated selection is available. If the selected role has features that include optional settings, the user is asked to decide if the settings are needed in the device.

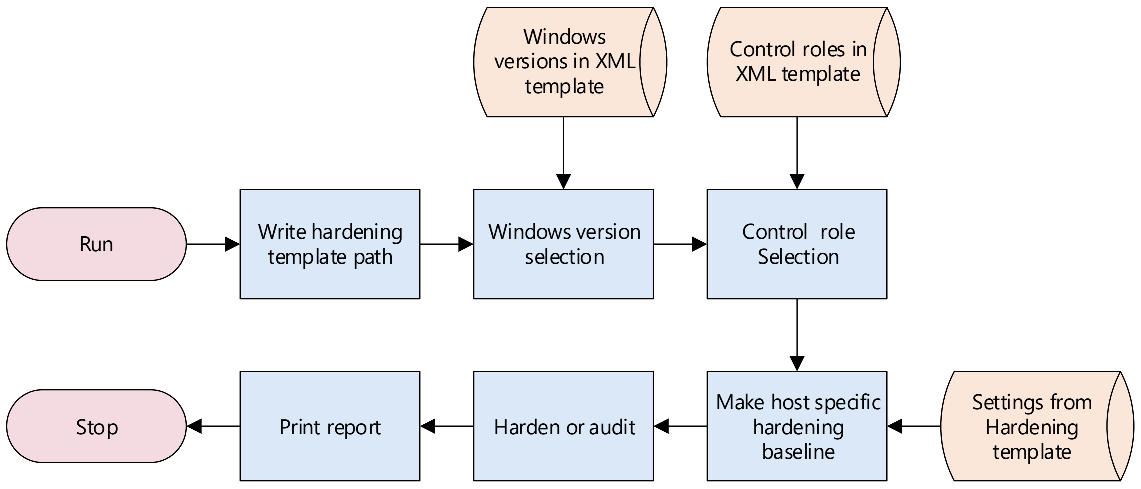
Windows versions in XML template
Control roles in XML template
Run
Write hardening template path
Windows version selection
Control role Selection
Stop
Print report
Harden or audit
Make host specific hardening baseline
Settings from Hardening template

***Figure 15.*** *The flow chart of hardening and auditing.*

To run scripts locally, the PowerShell script execution policy must be set to *Unrestricted* or *RemoteSigned*. However, by using the right mouse click to a script and selecting *Run with PowerShell*, it runs the script even when the execution policy is set to *Restricted* or *AllSigned*. The reason for this seems to be that when a script is run this way, it runs with the *Bypass* policy [53]. When either the hardening or the auditing script is executed, the user is asked to insert a path for the hardening template. When the template is found and read, the script generates a Windows version selection from the Windows templates included in the hardening template. The control system role selection is also generated from the template. The host specific hardening baseline is then generated from the Windows template with the setting exceptions in features included in the selected role. The baseline is then used for either hardening or auditing, depending on the script used. While the hardening script only prints the configured parameters directly into the PowerShell window, the audit script prints more useful reports in html and csv formats. After the computer is hardened, the auditing script should be used to verify the configuration, and to get a better report.

Auditing reports, which are printed in html and csv formats, include all the settings in the baseline. Each setting is printed on its own row. Columns indicate *Type*, *Name*, *TargetValue*, *CurrentValue*, and *Verdict* parameters. *Type* indicates the type of the setting, *TargetValue* is the value in the baseline, *CurrentValue* is the value in the computer, and *Verdict* indicates if the setting is properly configurated. This is done by comparing target and current values. If the values match, *Verdict* indicates *OK* and if not, it indicates *Failed*. Also, if the current value cannot be accessed from the system, the *Verdict* is *Unknown*.

## 7.4 Difficulties faced

Making the scripts was more difficult than was previously thought. This was because, although PowerShell was hyped as an all-powerful management tool for Windows, it seemed a bit unfinished, at least for this kind of local hardening and auditing. Many configuration parameters needed did not have commands included in PowerShell. Not even in PS version 5. Third-party modules that provided some of those tools existed, but because of the requirements, these could not be used. Commands for local user and group management did appear in PowerShell version 5.1, but too late for the scripts. However, it seems promising that Microsoft is investing in PowerShell. The lack of tools was solved by using old legacy tools found in the operating systems. For example, tools like Secedit, and assoc command in the Windows command prompt, that are still available in the latest Windows versions. These tools introduced some new problems, such as how to manage errors, and how the security templates of Secedit should be managed.

Inconsistency in tools also caused a lot of difficulties. For example, while the *Set-Service* command did have a *StartupType* handle, the *Get-Service* command did not. This meant, that auditing of services could not be done with same tools as hardening. Service startup types were found by using the *Get-WmiObject* command for the Win32_Service WMI class. This class included the method *ChangeStartMode* and a string *StartMode* that were used in the scripts. These also had some inconsistency included. For example, when setting a service startup type to automatic, the method used was *ChangeStartMode("Automatic"),* the *StartMode* string is shortened to *"Auto".* Another problem was with using Secedit, which could not export all same security policies that it could import into the system. Using some other means than Secedit could be better. At the beginning, the Windows Registry was intended to be used, but finding required Registry keys and values seemed to be quite difficult. This was partly solved by using Windows Security Templates, that included keys for those settings. Registry keys were still not used, because all the needed configuration parameters did not have Registry keys. Also, the Windows Security Baseline did not have example values for keys, so all those keys would have to be manually examined. Another problem with Registry keys was noticed with Windows 10 Settings menu items. It was found that if these settings were not configured, or were only manually changed in GUI, the registry keys did not exist. This meant that if those settings are audited by using Registry, they must first be configured with Registry keys.

The automatic checking option for installed features was not done because the lack of a reliable way to do it. At one point the idea was to accomplish it by using the Win32_product WMI class, which gives a list of installed products. This was found to reconfigure all installed applications, which might cause issues for the hardening configuration. Microsoft support suggests using the Win32red_AddRemovePrograms class instead. [54] This kind of functioning caused some concern about all other tools used in the scripts.

## 7.5 Testing the scripts

Testing was done to verify that the hardening script sets, and that the audit script gets all the settings correctly. The auditing script was used for testing the hardening script, and the hardening script was used to set different configuration scenarios to test the auditing script. A lot of manual changing and verifying of parameters was also done. Because there were so many settings involved with many different value options to test, not all of them were tested. Most tests focused on hardening values of the settings. Some testing was also done for the default parameters of Windows to remove hardening from the system. This was done by adding a default configuration template into the *WindowsTemplates* element in the hardening template. This allowed to take a step back to the default configuration, and in this way to make new hardening tests faster than if resuming back to a virtual snapshot. A few problems were found that were not fixed in time, such as not getting all necessary security policy parameters with Secedit. To notify this, the *Unknown* parameter was added into the Verdict column in the audit report.

Testing was done in Windows 10 and Windows Server 2012 R2, but without the ICS environment. This means that proper tests in the control system environment are needed in future, to ensure that the control system is not affected if the scripts are used.

## 7.6 Future development

Even though the scripts meet their purpose for the most part, they do not include all functionality that was planned. Also, a lot more exception and error handling should be included, as well as more testing of the different configuration templates. Planned features that were not yet included are an automatic checking option for installed automation features, ability to include multiple different roles in a host computer, and creating a user specified feature list. Also, functions for the registry type parameters did not get finished in time, even though some registry hardening settings were included in the security template. These functions can be created with *Set-ItemProperty* and *Get-ItemProperty* commands, which are found in the Microsoft.PowerShell.Management PowerShell module. With the included functions for registry parameters, it should be figured out what security policy settings in the hardening template could be replaced with registry settings. This way the need for Secedit could be decreased or removed altogether. Including the Windows Settings menu items in the hardening configuration could also be very useful, because currently these settings must be configured and audited manually, and the Windows updates sometimes change these settings. Additionally, more settings from the Windows security baselines should be added into the hardening configuration baseline. However, including new parameters into hardening needs comprehensive testing before they can be applied into the production environment.

The method used in the scripts to allow setting exceptions for each feature, could be used for configuring application whitelisting or firewall exceptions. This would allow automated configuration of application whitelist or firewall exceptions for installed applications. Tools like Windows AppLocker and Firewall are included in Windows operating systems, and already include management commands in PowerShell.

Remote hardening features can be quite easily added to the PowerShell scripts. For example, the way that the script uses a list of IP addresses of available host devices in an accessed network segment to harden or audit all of those. This needs Windows Remote Management to be turned on, and might cause misconfigurations that are hard to notice. If this kind of a function is needed, the PowerShell DSC should be reviewed first, because it functions very similarly.

Two methods for securing the scripts against unwanted changes were considered in this thesis. The scripts could be digitally signed. This means that if the scripts are changed and the PowerShell script execution policy set as *AllSigned,* PowerShell should not allow execution of the scripts. However, this seemed to be very easily bypassed. Another idea was to convert the scripts to executives, though hardening and auditing could also be made, for example in C#, with the use of the PowerShell class, if PowerShell tools are required.

# 8. CONCLUSION

In this thesis, the goal was to study how hardening should be managed over the ICS lifecycle, and how to deal with the problems met in maintaining the OS hardening configuration, especially in the Windows environment. To make the hardening management possible, automated hardening and auditing scripts were implemented with Windows PowerShell. Additionally, some hardening configuration improvements were researched to better meet the demands of newer Windows operating systems. The review showed that hardening is underrated in many security guides. This might be because it does not add fancy new security software or features, but vice versa. Although hardening is usually used as a security feature in ICSs, a more systematic hardening approach can improve security, reduce unnecessary network traffic, and free computer resources. The need for hardening management also increases with the use of more modern operating systems.

To make hardening management possible in build and operation phases, two new scripts are developed in this thesis. One for hardening the system and other for auditing it. The scripts are implemented with Windows PowerShell to review its capabilities for hardening and auditing of the Windows environment. Windows PowerShell is feasible for local hardening and auditing script implementation, but with some difficulties. While PowerShell operates well, it seems a bit unfinished when considering the tools available. Tools, such as local user and group management commands, appeared only in PowerShell 5.1 during this thesis, and were not used in the scripts. Managing of network and service settings needed multiple different methods, and some settings needed to be configured with old command line tools. In addition, there were tools that executed unwanted configuration restoring when used. However, if Microsoft keeps pushing PowerShell in the right way, it can be a very powerful tool to implement hardening and auditing to ICS devices. Even if PowerShell is not directly used, it can be implemented, for example, into C# program with PowerShell class. Additionally, remote functions for hardening and auditing scripts can be easily added with PowerShell to harden or audit multiple devices at once. Though, if remoting is needed, PowerShell DSC should be reviewed.

When considering the use of Windows operating systems in ICSs, it was clear that new Windows versions have more features and automatic functions that can be harmful in ICS. This leads to the conclusion that hardening is even more important in new Windows systems. However, using a modified Windows 10 IOT Enterprise version can make it a little easier. When it comes to Windows updates, the long-term servicing channel should be used in Windows 10 to reduce feature updates. Overall Windows 10 caused a lot of problems in the target environment. Its automated functions and updates restored security features that conflicted with the control system. Disabling those was difficult, because the features enabled themselves sometime after disabling. This kind of functioning can be justified to keep an unskilled user from harming their security, but is not acceptable in an ICS. This means that when using later Windows operating systems in an ICS, the need for hardening increases. Also, the need of management features, such as well-designed configuration audition is necessary. When considering both operating systems used in this thesis, the Windows Server 2012 R2 is better suited for ICS use. This might be because it is a server version or because problematic features are included in Windows 10, which can raise some concern about Windows Server 2016.

# 9. REFERENCES

[1] Google Project Zero, "Reading privileged memory with a side-channel," 3 1 2018. [Online]. Available: https://googleprojectzero.blogspot.fi/2018/01/reading-privileged-memory-with-side.html. [Accessed 15 1 2018].

[2] Pasi Ahonen *et al.*, "Kyber-Teo - tuloksia 2014-2016," VTT, 2016.

[3] Keith Stouffer, *et al.*, Guide to Industrial Control Systems Security, National Institute of Standards and Technology, 2015, p. 247.

[4] Industrial Control Systems Cyber Security Emergency Response Team, Recommended Practice: Improvinf ICS Cybersecurity with Defense-in-Depth Strategies, U.S. Department of Homeland Security, 2016, p. 49.

[5] L. Piètre-Cambacédès, M. Tritschler and G. Ericsson, "Cybersecurity Myths on Power Control Systems: 21 Misconcepts and False Beliefs," *IEEE Transactions on Power Delivery,* vol. 26, pp. 161-172, 2011.

[6] Jari Seppälä, Mikko Salmenperä, "Towards Dependable Automation," in *Cyber Security: Analytics, Technology and Automation*, 2015, s. 229-249.

[7] ANSI/ISA-99.00.01-2007, The Instrumentation, Systems and Automation Society, 2007, p. 92.

[8] Centre for the Protection of National Infrasturcure, Manage industrial control systems lifecycle, Centre for the Protection of National Infrasturcure, 2015, p. 24.

[9] Suomen Automaatioseura ry, Automaatiosuunnittelun prosessimalli, Suomen Automaatioseura ry, 2007, s. 13-17.

[10] Risto Ajo *et al.*, Laatu automaatiossa, SAS, s. 245.

[11] S. Mandfield-Devine, "Ransomware: taking businesses hostage," *Network Security,* vol. 2016, no. 10, pp. 8-17, 2016.

[12] K. Scarfone and P. Mell, Guide to Intrusion Detection and Prevention Systems, National Institute of Standards and Technology, 2007, p. 127.

[13] Suomen Automaatioseura ry, Teollisuusautomaation tietoturva, Suomen Automaatioseura ry, 2005, s. 160.

[14] E. Knapp, Industrial Network Security, Elsevier, 2011, p. 341.

[15] U.S. Department of Homeland Security, Cyber Security Procurement Language for Control Systems, U.S. Department of Homeland Security, 2009, pp. 1-2, 34-35.

[16] J. Vacca, Computer and Information Security Handbook, Elsevier, 2013, pp. 97-181.

[17] European Network and Information Security Agency, Protecting Industrial Control Systems, European Network and Information Security Agency, 2011, p. 70.

[18] K. Scarfone and P. Hoffman, Guidelines on Firewalls and Firewall Policy, National Institute of Standards and Technology, 2009, p. 48.

[19] K. S. Murugiah Souppaya, Guide to Enterprise Patch Management Technologies, National Institute of Standards and Technology, 2013, pp. 1-7.

[20] Tom Steven *et al.*, Recommended Practice for Patch Management of Control Systems, U.S. Department of Homeland Security, 2008, p. 23.

[21] Arnold Johnson *et al.*, Guide for Security-Focused Configuration Management of Information Systems, National Institute of Standards and Technology, 2011, p. 88.

[22] National Institute of Standards and Technology, "The United States Government Configuration Baseline," [Online]. Available: https://usgcb.nist.gov/usgcb/microsoft_content.html. [Accessed 10 2017].

[23] K. Scarfone, M. Soppaya, A. Cody and A. Orebaugh, Technigal Guide to Information Security Testing and Assessment, National Institute of Standards and Technology, 2008, p. 80.

[24] P. Ahonen, TITAN-käsikirja, VTT, 2010, pp. 57-113.

[25] J. Wack, M. Tracey and M. Souppaya, Guideline on Network Security Testing, National Institute of Standards and Technology, 2003, p. 92.

[26] D. Duggan, "Penetration Testing of Industrial Control Systems," Sandia National Laboratories, 2005.

[27] P. Bowen, J. Hash and M. Wilson, Information Security Handbook: A Guide for Managers, National Institute of Standards and Technology, 2006, pp. 19-22.

[28] Richard Kissel *et al.*, Security Considerations in the System Development Life Cycle, National Institute of Standards and Technology, 2008, pp. 1-36.

[29] Microsoft Corporation, "Microsoft products," [Online]. Available: https://support.microsoft.com/en-us/allproducts. [Accessed 6 11 2017].

[30] J. Krause, Mastering Windows Server 2016, Packt Publishing, 2016.

[31] Microsoft Corporation, "Installing Windows PowerShell," [Online]. Available: https://docs.microsoft.com/en-us/powershell/scripting/setup/installing-windows-powershell?view=powershell-5.1. [Accessed 7 11 2017].

[32] A. Margosis, "Guidance on Disabling System Services on Windows Server 2016 with Desktop Experience," 29 5 2017. [Online]. Available: https://blogs.technet.microsoft.com/secguide/2017/05/29/guidance-on-disabling-system-services-on-windows-server-2016-with-desktop-experience/. [Accessed 6 11 2017].

[33] Microsoft Corporation, "Configure Windows telemetry in your organization," 17 10 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows/configuration/configure-windows-telemetry-in-your-organization. [Accessed 18 12 2017].

[34] Microsoft Corporation, "Overview of Windows as a service," 27 7 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows/deployment/update/waas-overview. [Accessed 14 11 2017].

[35] Microsoft Corporation, "Windows Server 2016 new Current Branch for Business servicing option," 12 7 2016. [Online]. Available: https://blogs.technet.microsoft.com/windowsserver/2016/07/12/windows-server-2016-new-current-branch-for-business-servicing-option/. [Accessed 14 11 2017].

[36] Microsoft Corporation, "Applocker overview," 4 5 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows/device-security/applocker/applocker-overview. [Accessed 8 11 2017].

[37] Microsoft Corporation, "Requirements to use AppLocker," 5 4 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows/device-security/applocker/requirements-to-use-applocker. [Accessed 8 11 2017].

[38] Thomas Shinder *et al.*, Windows Server 2012 Security from End to Edge and Beyond, Elsevier, 2013, pp. 267-339.

[39] E. Goad, Windows Server 2012 Automation with PowerShell Cookbook, Packt Publishing Ltd., 2013.

[40] J. Jones and J. Hicks, Learn Windows PowerShell 3 in a month of lunches, New York: Manning Publications Co., 2013, p. 343.

[41] Australian Cyber Security Centre, Securing PowerShell in the Enterprise, Australian Cyber Security Centre, 2016, p. 17.

[42] Microsoft Corporation, "Windows Security Baselines," 17 10 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows/device-security/windows-security-baselines. [Accessed 9 11 2017].

[43] Microsoft Corporation, "Microsoft Security Compliance Toolkit 1.0," 17 10 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows/device-security/security-compliance-toolkit-10. [Accessed 10 11 2017].

[44] A. Margosis, "Security Compliance Manager (SCM) retired; new tools and procedures," 15 6 2017. [Online]. Available: https://blogs.technet.microsoft.com/secguide/2017/06/15/security-compliance-manager-scm-retired-new-tools-and-procedures/. [Accessed 10 11 2017].

[45] Microsoft Corporation, "WSUS 3.0 SP2 System Requirements," 19 1 2016. [Online]. Available: https://technet.microsoft.com/en-us/library/dd939928(v=ws.10).aspx. [Accessed 14 11 2017].

[46] Microsoft Corporation, "Windows Server Update Services (WSUS)," 22 5 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows-server/administration/windows-server-update-services/get-started/windows-server-update-services-wsus. [Accessed 14 11 2017].

[47] R. Kazanciyan and M. Hastings, Investigating PowerShell attacks, Black Hat, 2014, p. 25.

[48] Microsoft Corporation, "Group Policy for Beginners," 27 4 2011. [Online]. Available: https://technet.microsoft.com/en-us/library/hh147307%28v=ws.10%29.aspx?f=255&MSPPError=-2147217396. [Accessed 14 11 2017].

[49] J. MoskoWitz, Group Policy Fundamentals, Security, and the Managed Desktop, Wiley, pp. 1-72.

[50] D. Jones and S. Murawski, The DSC Book, PowerShell.org, 2017, p. 12.

[51] Microsoft Corporation, "Windows 10 RS1 and Server 2016 Security Baseline," [Online]. Available: https://msdnshared.blob.core.windows.net/media/2016/10/Windows-10-RS1-and-Server-2016-Security-Baseline.zip. [Accessed 10 5 2017].

[52] Microsoft Corporation, "Windows 10 IoT Platform Overview," [Online]. Available: http://wincom.blob.core.windows.net/documents/Windows_10_IoT_Platform_Overview.pdf. [Accessed 22 11 2017].

[53] E. Atac, "What's behind 'Run with PowerShell' context menu?," 19 7 2016. [Online]. Available: https://p0w3rsh3ll.wordpress.com/2016/07/19/whats-behind-run-with-powershell-context-menu/. [Accessed 18 12 2017].

[54] Microsoft Support, "Event log message indicates that the Windows Installer reconfigured all installed applications," 1 8 2017. [Online]. Available: https://support.microsoft.com/en-us/help/974524/event-log-message-indicates-that-the-windows-installer-reconfigured-al. [Accessed 24 10 2017].

[55] Microsoft Corporation, "Comparison of Standard and Datacenter editions of Windows Server 2016," 3 1 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows-server/get-started/2016-edition-comparison. [Accessed 6 11 2017].

[56] Microsoft Corporation, "Windows Server 2012 Security Baseline," 28 1 2013. [Online]. Available: https://technet.microsoft.com/en-us/library/jj898542.aspx. [Accessed 10 11 2017].

[57] U.S. Department of Homeland Security, "Common Cybersecurity Vulnerabilities in Industrial Control Systems," U.S. Department of Homeland Security, 2011.