UNIVERSITY OF TARTU
Institute of Computer Science
Cyber Security Curriculum

Martin Jõgi

Establishing, Implementing and Auditing Linux Operating System Hardening Standard for Security Compliance

Master's Thesis (30 ECTS)

Supervisors: Truls Tuxen Ringkjob
Associate Prof. Raimundas Matulevičius

Tartu 2017

Establishing, Implementing and Auditing Linux Operating System Hardening Standard for Security Compliance

Abstract:

Regulations create challenges to the companies as they must meet tough security standards for security compliance every year. Not following security standards are making companies more vulnerable to cyber threats. This paper provides a proof-of-concept solution for being compliant with operating system hardening requirements of the company by establishing, implementing and auditing Linux (Debian) operating system hardening standard. This work will focus on building a hardened virtual machine image for Microsoft Azure platform that is compliant with the hardening standard that is established in this thesis. As it is not enough to just meet the company security compliance requirements, then there is a need to ensure auditability of it, therefore a proof-of-concept solution is built for automatically auditing the operating system hardening standard that continuously allows to validate the gap between the standard and actual configurations on virtual machines. To demonstrate the result, the author analysed virtual machines compliance state before and after implementing the new operating system hardening standard.

Keywords:

Security compliance, operating system hardening, security standard, auditing, automation

CERCS: P170 Computer science, numerical analysis, systems, control

Linux operatsioonisüsteemi turvalisuse tugevdamise standardi loomine, implementeerimine ja auditeerimine turvavastavuseks

Lühikokkuvõte:

Regulatsioonid loovad väljakutseid ettevõtetele, sest nad peavad igal aastal turvavastavuseks täitma turvastandardeid. Turvastandardite mitte järgimine teeb ettevõtted küberohtudele haavatavaks. Antud magistritöö pakub lahenduse, et täita ettevõtte operatsioonisüsteemi tugevdamise nõudeid: luues, implementeerides ja auditeerides Linux (Debian) operatsioonisüsteemi tugevdamise turvastandardit. Magistritöö käsitleb tugevdatud virtuaalmasina tõmmise ehitamist Microsoft Azure pilveplatvormile, mis vastab operatsioonisüsteemi tugevdamise standardile. Kuna lihtsalt ettevõtte turvavastavuse nõuete täitmine ei ole piisav, siis tuleb tagada selle auditeeritavus, et turvavastavuse täitmist audiitoritele tõestada. Antud magistritöös on koostatud lahendus, mis automaatselt auditeerib operatsioonisüsteemi tugevdamise standardit ja võimaldab kinnitada erinevusi standardi ja tegelike virtuaalmasinate konfiguratsioonide vahel. Autor on analüüsinud virtuaalmasinate turvavastavust enne ja pärast uue operatsioonisüsteemi tugevdamise standardi rakendamist.

Võtmesõnad:

Turvavastavus, operatsioonisüsteemi tugevdamine, turvastandard, auditeerimine, automatiseerimine

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Table of Contents

Acknowledgments 5
Table of Abbreviations 6
1 Introduction 7
1.1 Scope 7
1.2 Problem Statement 7
1.3 Methodology 9
1.4 Structure of the Thesis 10
2 Background 12
2.1 Security Compliance 12
2.2 Operating System Hardening 12
2.3 Security Standards and Hardening Guidelines 13
2.4 Security Technical Implementation Guide for Red Hat Linux 13
2.5 Related Work in this Field 14
2.6 Summary 14
3 Establishing the Operating System Hardening Standard 15
3.1 Approach 15
3.2 Analysing the Holding Company Linux Hardening Standard 15
3.3 Results 23
3.4 Threats to Validity 23
3.5 Establishing the Hardening Standard for the Parent Company 23
3.6 Summary 26
4 Implement the Operating System Hardening Standard 27
4.1 Approach 27
4.2 Benefits of Hardened Image 27
4.3 Compare Image Creation Tools 28
4.4 Choosing Hardening Requirements for Implementing 30
4.5 Preparing Development and Testing Environment 30
Preparing Test Kitchen 30
Developing Chef Cookbook 31
Preparing Packer template 33
4.6 Building Hardened Virtual Machine Image 34
4.7 Related Work in this Field 35
4.8 Summary 36
5 Auditing the New Hardening Standard 37

5.1 Approach 37
5.2 The Importance of Automated Auditing 37
5.3 Choosing the Tool for Automated Compliance Auditing 38
5.4 Compliance as Code and InSpec 38
5.5 Developing Security Compliance Auditing Tests 40
5.6 Related Work in this Field 41
5.7 Summary 41
6 Validation of Proof of Concept Solution 43
6.1 Methodology and Validation Question 43
6.2 Results 43
6.3 Discussion 45
6.4 Threats to Validity 45
6.5 Summary 45
7 Conclusions and Future Work 46
7.1 Limitations 47
7.2 Answer to the Research Questions 47
7.3 Conclusions 49
7.4 Future Work 49
8 References 50
Appendix 53
I. The Holding Company Linux Hardening Standard 53
II. Chef Cookbook for Implementing the New Hardening Standard 60
III. InSpec Profile for Auditing the New Hardening Standard 66
IV. Results of Validating the Proof of Concept Solution 70
V. License 79

Acknowledgments

My gratitude goes to my supervisors Truls Tuxen Ringkjob from Tallinn University of Technology and Raimundas Matulevičius from University of Tartu for providing support and feedback for the thesis.

I would also like to thank my employer and colleagues who supported me with this thesis.

Finally, I would like to thank my family. Special thanks to my wife – I would not have been able to make this without your patience, understanding and support. I would also like to thank my little daughter who inspired me while writing this thesis.

Table of Abbreviations

ARM - Azure Resource Manager

AWS - Amazon Web Services

CIS - Center for Internet Security

CLI - Command-Line Interface

DCCP - Datagram Congestion Control Protocol

DISA - Defense Information Systems Agency

DoD - Department of Defense

HIPAA - The Health Insurance Portability and Accountability Act

ICMP - Internet Control Message Protocol

ISO - International Organization for Standardization

LDAP - Lightweight Directory Access Protocol

NIST - National Institute of Standards and Technology

NSA - National Security Agency

OS - Operating System

PCI - Payment Card Industry

PCI DSS - Payment Card Industry Data Security Standard

RDS - Reliable Datagram Sockets

SCAP - The Security Content Automation Protocol

SCTP - Stream Control Transmission Protocol

SSH - Secure Shell

STIG - Security Technical Implementation Guide

TDD - Test-Driven Development

TIPC- Transparent Inter-Process Communication

VM - Virtual Machine

XCCDF - Extensible Configuration Checklist Description Format

1 Introduction

Nowadays companies are providing their services and products online more and more. Companies are processing sensitive information, therefore for protecting themselves and their customers they must meet several security standards for security compliance each year.

Research reports show that security breaches are increasing. For example, PwC research 2015 states that many companies have suffered due to security breaches (PwC, 2015). According to Verizon (2016) and Identity Theft Resource Center (2017) data breach reports, security breaches are still on the rise. Verizon 2015 PCI (Payment Card Industry) compliance report states that many companies are still failing PCI compliance due to lack of implemented controls (Oosten, Baritchi & Koten, 2015). These research results clearly show that security compliance is important but not easy to achieve - having clear security standards, implementing and auditing these standards should be vital for any industry for minimizing the risks against cyber threats.

Cloud computing is widely used these days, mostly because it allows companies deliver products much faster and with smaller effort. However, based on our experience, standard Linux distribution images in the cloud are not built for security by default nor they are hardened against any known security standards or guidelines. Therefore, this thesis explores possibility to build a hardened Debian Linux virtual machine image to ensure that any newly created virtual machine is compliant according to the operating system (OS) hardening standard which we are presenting in this thesis. As it is not enough to just implement the OS hardening standard requirements, we need to ensure auditability of it to provide evidence to the auditors. For that, an automated compliance tool is needed to verify the compliance level of the virtual machines. Therefore, this thesis concentrates on both – building an automated proof-of-concept solution for implementing and auditing the OS hardening standard using existing open source tools out there.

1.1 Scope

In this thesis, we call the list of OS hardening requirements of the holding company to "the holding company OS hardening standard". To support the work in thesis, we are listing these hardening requirements in Appendix 1. These hardening requirements are based on publicly known DISA STIG (Defense Information Systems Agency Security Technical Implementation Guide) for Red Hat (Defense Information Systems Agency, 2015). In the reality, the holding company standard contains several other hardening requirements, but in thesis we are taking only publicly available hardening requirements based on DISA STIG for Red Hat in to scope due to confidentiality reasons. Therefore, we acknowledge that the holding company standard in this thesis is identical to DISA STIG for Red Hat.

We are not going to implement all the hardening requirements in this thesis. The requirements that we are implementing for our proof-of-concept prototype are listed in the section 4.4. We are implementing the solution in Microsoft Azure platform with a help of configuration management tool Chef. Describing Chef and Microsoft Azure base fundamentals are not in scope of this thesis. We are implementing and auditing security controls which are technically possible, any procedural or administrative security controls are out of scope.

1.2 Problem Statement

The author of this master thesis is working at IT company which got acquired by the holding company. The author's team manages thousands of Linux Debian based servers.

The problem is that hardening and securing the operating system of these servers are based on experience of service engineers and results of internal penetration tests. There are no operating system hardening standards followed, while at the same time the parent company is under increasing pressure because of regulatory requirements of the holding company. The parent company must be compliant according to the holding company OS hardening standard which is based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015).

In addition, it is difficult to ensure the continuous compliance, because the company doesn't have an automated solution that lets, on regular basis, validate the gap between the standard and actual hardening configurations and practices on servers. This current process involves manual data collection which takes many resources and makes the author's team situation difficult during the holding company internal audits. There are risks that some servers are not configured properly from security compliance perspective. Therefore, there is a need for an automated compliance tool to verify the compliance level of the parent company Linux servers.

In this thesis, we aim to build a hardened Debian Linux virtual machine image which is compliant with the holding company regulatory requirements. This ensures that any newly created virtual machine is already compliant according to the standard. To accomplish this task, we have chosen Microsoft Azure as a platform and Chef configuration management tool for developing and implementing the OS hardening standard requirements. For hardened image creation, we have chosen three open source tools for analysis and suitable tool will be chosen for the company. Finally, we will implement a solution to automatically audit the OS hardening standard by developing security compliance auditing tests using open source compliance analysis tool and analyse the findings. Before we can start implementing the solution, we need to find answers to the research questions for satisfying the goals of this thesis.

This thesis is mainly focused on meeting security compliance of the holding company, with the following goals:

1. Analyse the holding company OS hardening standard by identifying only relevant requirements which can be implemented on the parent company Linux Debian servers. Based on the results, establish a new OS hardening standard for the parent company.
2. Build a proof-of-concept prototype for implementing the OS hardening standard by creating a hardened Debian Linux virtual machine image for Microsoft Azure platform.
3. Build a proof-of-concept prototype for automatically auditing the OS hardening standard.
4. Analyse the results before and after implementing the OS hardening standard.

The main research question of this thesis is: *How to be compliant with security compliance requirements of the holding company?* To answer to this question, we need to answer to the following research questions below.

RQ1: What are the security standards and hardening guidelines for security compliance in industry?

Why security compliance and OS hardening is important?

We have discovered that there are several security standards and hardening guidelines available such as PCI DSS, HIPAA, ISO (27000 series), NIST (SP800 series) and hardening guidelines provided by CIS, NSA, NIST and DISA.

RQ2: How to establish Linux operating system hardening standard for security compliance?

Is the holding company OS hardening standard based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015) suitable for the parent company? What hardening requirements are relevant for the parent company? What hardening requirements are already in place and implemented? What hardening requirements are not in place? How complicated it is to implement hardening requirements which are not in place on the parent company Debian systems?

We have identified all the relevant hardening requirements for the parent company Debian Linux systems from the holding company hardening standard. This allowed the author to establish the new hardening standard for the parent company that is used as a baseline for implementing the proof-of-concept solution in this thesis.

RQ3: How to implement Linux operating system hardening standard for security compliance?

What requirements of the OS hardening standard can be implemented on the parent company Debian Linux operating systems? As we can't implement everything, we need to answer to this question and concentrate on small part that we can implement in this project. What virtual machine image creation tool is suitable for the parent company for preparing the hardened base image? What it takes to integrate Microsoft Azure platform, Chef configuration management and chosen image creation tool for automatically implementing the hardening standard?

We have chosen Packer as suitable image creation tool. We have developed Chef *cookbook* (Appendix 2) that contains all the code for automatically implementing the hardening requirements from the new standard. We have integrated this Chef *cookbook* with image creation tool Packer and Microsoft Azure cloud platform for creating a hardened virtual machine image.

RQ4: How to audit Linux operating system hardening standard for security compliance?

What solutions are there that can be used for the automated security compliance auditing based on the company OS hardening standard? What tool is suitable for the parent company? What it takes to develop security compliance checks (tests) for the OS hardening requirements which we have implemented in this project?

We have chosen InSpec as suitable automated security compliance auditing tool. We have developed security compliance auditing tests for InSpec for automatically auditing the new OS hardening standard (Appendix 3).

RQ5: How many OS hardening requirements each chosen virtual machine (VM) meets from the new OS hardening standard?

For validating the proof-of-concept solution, we need to find answer to how many OS hardening requirements are met on a virtual machine that was created with hardened and non-hardened VM image.

According to automated security compliance auditing tool InSpec scan results, we discovered that 29 hardening requirements (~21%) of 138 were not met on virtual machine that was created with default (non-hardened) virtual machine image (Appendix 4).

1.3 Methodology

In order to understand the problem better, we have done background study of several data breach security reports. Based on these reports, we understand why security compliance is

important and needed. In addition, for understanding security compliance and operating system hardening, we are exploring several books and papers. Moreover, we have done additional background study of security standards and hardening guidelines.

To successfully address the problem and achieve our goals, we will find answers to our research questions. During this time, we will explore additional literature. Considering that there is not much research done in this field, the sources will be found mainly on internet: academic search engines, publications, white and grey literature, e-books, websites and technical documentations.

While analysing the holding company OS hardening standard and choosing relevant hardening requirements, we are using several Linux hardening books as a guideline. These books help us to understand the differences between Red Hat and Debian distributions and help the author during decision making while choosing relevant requirements. In addition, we are using Linux hardening books during implementation of the hardening standard.

Before implementing and auditing the OS hardening standard, we have done a research of relevant work in this field by exploring several papers, books and websites. After choosing a correct tooling for hardened virtual machine image creation and for security compliance auditing, we are installing and configuring these by following technical documentations. We are following Test-Driven Development (Erdogmus, Morisio, & Torchiano, 2005) software development approach when developing our proof-of-concept hardening and auditing solution. While developing OS hardening requirements into configuration management tool Chef and developing security compliance auditing tests, we are also following guidelines provided for each security control in DISA STIG for Red Hat (Defense Information Systems Agency, 2015).

1.4 Structure of the Thesis

To answer to the research questions, the thesis will be structured as follows:

- Chapter 1. Introduction - introduces the thesis and states the problem and research questions that the thesis handles.

- Chapter 2. Background – introduces the necessary information and background study for understanding the thesis project.

- Chapter 3. Establishing the Operating System Hardening Standard - the holding company Linux hardening standard based on DISA STIG for Red Hat Linux (Defense Information Systems Agency, 2015) is being analysed and based on the results, answer to the second research question is done.

- Chapter 4. Implementing the Operating System Hardening Standard - describes what it takes to create a hardened Linux Debian OS virtual machine image which is compliant with the OS hardening standard. Image creation tools are analysed and proof-of-concept solution is implemented. Answer to the third research questions is done.

- Chapter 5. Auditing the Operating System Hardening Standard – for answering to the fourth research question, we are looking briefly what solutions are there that can be used as automated security compliance auditing tool. After choosing suitable compliance auditing tool for the parent company, proof-of-concept prototype will be implemented by developing compliance tests for auditing the new OS hardening standard.

- Chapter 6. Validation of Proof of Concept Solution - to validate our solution, we analyse the security compliance auditing results before and after implementing the OS hardening standard.

- Chapter 7. Conclusions and Future Work - presents the summary, conclusions, answers to the research questions and limitations of the thesis. Additional discussion is done in regards of the future work.

- Appendix I. The Holding Company Linux Hardening Standard – contains a list of OS hardening requirements that are taken from DISA STIG for Red Hat (Defense Information Systems Agency, 2015).

- Appendix II. Chef Cookbook for Implementing the Hardening Standard – examples of code snippets for Chef configuration management tool for implementing the proof-of-concept virtual machine image.

- Appendix III. InSpec Profile for Auditing the New Hardening Standard – examples of security compliance auditing tool InSpec code snippets for auditing the new OS hardening standard.

- Appendix IV. Results of Validating the Proof-of-Concept Solution – detailed InSpec report of security compliance results before and after implementing the OS hardening standard on virtual machines.

2 Background

In this chapter, we will present the background information for understanding the thesis. As our goal in this thesis is meeting security compliance of the holding company, we will introduce what security compliance is and why it is important. Since our thesis concentrates also to operating system hardening, we will present the meaning and importance of OS hardening. In addition, we describe several security standards. Moreover, DISA Red Hat Linux OS hardening guideline is introduced as we are using it as our baseline in this thesis.

In this chapter, we will answer to *RQ1* which is: "What are the security standards and hardening guidelines for security compliance in industry?" To support answer to this question, we need to understand the importance of security compliance and OS hardening.

2.1 Security Compliance

There are several definitions for security compliance. Lustig (2015) states that "The term regulatory compliance refers to the adherence of an organization to the laws, specifications, regulations, and standards required for an industry" (p.11).

Julisch (2009) describes security compliance as follows: "Security compliance, in IT systems, is the state of conformance with externally imposed functional security requirements and of providing evidence (assurance) thereof" (p.72).

According to Lustig (2015), companies need to meet several requirements, otherwise there can be several negative consequences. We agree with Lustig (2015) statement: "Many regulatory standards exist to protect individuals' and companies' data" (p.11), therefore we will briefly describe several industry accepted security standards in the section 2.3.

Several research reports claim that security incidents are increasing. The Verizon PCI Compliant Report 2015 created by Oosten, Baritchi & Koten (2015), presents that companies who had data breach were less PCI DSS (Payment Card Industry Data Security Standard) compliant than companies who met PCI DSS requirements. Therefore, based on these results, we can clearly say that security compliance is important, because it helps companies to achieve more security.

2.2 Operating System Hardening

We have found several literatures that describes operating system hardening very well. Andress (2014) describes operating system hardening as follows: "one of the main goals of operating system hardening is to reduce the number of available avenues through which our operating system might be attacked" (p.132).

Bauer (2003) states that Linux operating system has historically used to be insecure by default - several Linux distributions have pre-installed applications that are not needed and which can be disabled. Bauer (2003) also phrases Marcus Ranum's sentence from "A Network firewall" paper: "That which is not explicitly permitted is forbidden" (Ranum, 1992, p. 7). Bauer (2003) agrees that this sentence fits well to OS hardening. Although, the book and research paper is old, we agree that these thoughts are still valid – nowadays there are still several not needed applications pre-installed on operating systems which increase the attack surface.

In addition, OS hardening is necessary to be compliant with security compliance requirements. For example, several industry accepted security standards require that OS hardening policy is followed and auditors require evidence that it is implemented accordingly. For example, PCI DSS requirement 2.2 states: "Develop configuration standards for all system

components. Assure that these standards address all known security vulnerabilities and are consistent with industry-accepted system hardening standards" (PCI Security Standards Council LLC, 2016, p. 31). We are briefly describing these standards and several other hardening guidelines as well in the section 2.3.

We believe that OS hardening is important for mitigating and minimizing the risks of security threats.

2.3 Security Standards and Hardening Guidelines

We have studied that data breaches are increasing these days. To protect companies and users, there are several industry accepted security standards available. For example, PCI DSS which is put together by credit card companies such as MasterCard, Visa and American Express (PCI Security Standards Council LLC, 2017). PCI DSS provides requirements to protect payment account details such as credit card numbers (PCI Security Standards Council LLC, 2016).

The Health Insurance Portability and Accountability Act (HIPAA), sets the standard for protecting personal health information (Health Information Privacy, 2017). There are several other known security standards such as ISO 27000 family standards (International Organization for Standardization, 2017) and NIST SP800 computer security standards (The National Institute of Standards and Technology, 2017a).

In addition, there are several industry-accepted hardening standards. The Center for Internet Security (CIS) is providing hardening guideline for Debian Linux systems (Center for Internet Security, 2017a). National Security Agency (NSA) provides detailed hardening guideline for Red Hat systems (National Security Agency, 2007). The same does National Institute of Standards and Technology (NIST) (The National Institute of Standards and Technology, 2017b) and DISA (Defense Information Systems Agency, 2015). Moreover, several Linux distributions have official hardening guidelines, such as Red Hat (Jahoda, et al., 2016) and Debian (Peña & Reelsen, 2012).

However, company's systems and requirements are different, therefore choosing a right standard is not always straightforward. Choosing a wrong guideline could break systems. Therefore, in this thesis in Chapter 3, we are analysing each requirement of the holding company hardening standard (that is based on DISA STIG for Red Hat) before implementing it in the parent company. Since we are using DISA STIG for Red Hat (Defense Information Systems Agency, 2015) as a baseline in thesis, we will do a brief introduction of it in the next section.

2.4 Security Technical Implementation Guide for Red Hat Linux

As the holding company hardening standard is based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015), we are briefly describing it in this section.

According to Defense Information Systems Agency (2015) document "Red Hat 6 STIG - Ver 1, Rel 7" ("U_RedHat_6_V1R7_Overview.pdf" file), the STIG for Red Hat is created

by NSA, Red Hat and DISA for increasing security of Department of Defense (DoD) (Defense Information Systems Agency, 2015).

Each requirement defined in this guideline has a Severity Category Code (CAT) I, II, or III (Defense Information Systems Agency, 2015). The Figure 1 presents vulnerability severity category code definitions of DISA STIG for Red Hat.

	DISA Category Code Guidelines
CAT I	Any vulnerability, the exploitation of which will **directly and immediately** result in loss of Confidentiality, Availability, or Integrity.
CAT II	Any vulnerability, the exploitation of which **has a potential** to result in loss of Confidentiality, Availability, or Integrity.
CAT III	Any vulnerability, the existence of which **degrades measures** to protect against loss of Confidentiality, Availability, or Integrity.

Figure 1. DISA Category Codes taken from the "Red Hat 6 STIG - Ver 1, Rel 7" document "U_RedHat_6_V1R7_Overview.pdf" (Defense Information Systems Agency, 2015)

The requirements from DISA STIG for Red Hat are published in XCCDF (Extensible Configuration Checklist Description Format) language which is published as XML file ("U_RedHat_6_V1R7_Manual-xccdf.xml") that can be used by SCAP (The Security Content Automation Protocol)-based scanning tools (Defense Information Systems Agency, 2015). There are tools and documentations available (Defense Information Systems Agency, 2017) for getting human readable output of these XML files. DISA STIG for Red Hat contains implementation and auditing guidelines for each security control (Defense Information Systems Agency, 2015).

In this thesis, we have listed the hardening requirements of the holding company based on DISA STIG for Red Hat in Appendix 1. In the next chapter, we are analysing each hardening requirement from the holding company standard for deciding whether they are suitable for the parent company Linux Debian systems.

2.5 Related Work in this Field

We have reviewed and took several Linux hardening books as a guideline (Turnbull, 2005; Bauer, 2003). These books help us to understand the importance of Linux OS hardening. In addition, they help us to understand the differences between Red Hat and Debian distributions and help the author during decision making while choosing the relevant requirements in the Chapter 3. Moreover, these books are helpful while implementing the requirements in the Chapter 4.

2.6 Summary

In this chapter, we have described the meaning and the importance of security compliance and operating system hardening. In addition, we have reviewed several industry accepted security standards such as PCI DSS, HIPAA, ISO, NIST and hardening guidelines provided by CIS, NSA, NIST and DISA STIG for Red Hat which is used as a baseline for implementing the proof-of-concept solution in this thesis.

3 Establishing the Operating System Hardening Standard

In this chapter, we are analysing the holding company hardening standard. Based on the analysis, the decision will be done if the parent company needs separate hardening standard. We will identify all relevant requirements for the parent company Debian Linux systems while analysing the holding company hardening standard.

In this chapter, we will answer to *RQ2* which is: "How to establish Linux operating system hardening standard for security compliance?" To find answer to this question, we will find answer to several sub-questions: "Is the holding company OS hardening standard based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015) suitable for the parent company? What hardening requirements are relevant for the parent company? What hardening requirements are already in place and implemented? What hardening requirements are not in place? How complicated it is to implement hardening requirements which are not in place on the parent company Debian systems?"

3.1 Approach

To achieve the first goal and answer to the *RQ2* that we have stated in the section 1.2, we are taking the holding company hardening standard (Appendix 1) as an input for analysis. The holding company is IT company that acquired the parent company where the author works at. The holding company requires that OS hardening requirements from DISA STIG for Red Hat (Defense Information Systems Agency, 2015) must be followed in the parent company.

As the parent company is using Debian Linux systems, then the author cannot be sure if Red Hat hardening standard is suitable for the parent company. There is also a chance that some hardening requirements are already in place, therefore the goal is to review and analyse the holding company Linux OS hardening standard (Appendix 1) and identify the relevant hardening requirements.

The author expects that there are three categories for relevant hardening requirements:

1. Requirements that are already in place;
2. Requirements that are not in place, but can be added easily;
3. Requirements that are not in place, but the solution is not straightforward and needs discussion;

To choose which hardening requirements go to which relevant category is based on years of work experience of the author. In some cases, the author was not sure which requirement belong to which category, therefore the author triggered specific scans over the parent company infrastructure using specific tools. Describing these tools are not in scope of this work. In addition, discussion was done with other teams for complex hardening requirements. To understand the differences between Debian and Red Hat systems and to support the analysis in this chapter, the author used hardening books as described in the section 2.5.

The result of the holding company standard analysis contains relevant hardening requirements for the parent company Debian systems. These requirements will be used for establishing the new hardening standard for the parent company. This new standard will be taken as a baseline for implementing the proof-of-concept solution.

3.2 Analysing the Holding Company Linux Hardening Standard

In this section, the author is analysing the holding company standard that is based on DISA STIG for Red Hat that is published as "U_RedHat_6_V1R7_Manual-xccdf.xml" file

(Defense Information Systems Agency, 2015). We are using the approach as described in the section 3.1.

To support the analysis, we have listed the holding company Linux hardening requirements based on DISA STIG for Red Hat (Defense Information *Systems Agency, 2015) in Appendix 1. Each requirement has a number to refer to.*

File integrity monitoring:

The requirement 1 states that a file integrity monitoring utility Aide is required (Defense Information Systems Agency, 2015). The parent company is using file integrity tool only on some servers and Samhain tool is used instead of Aide. The category for this requirement will be 2.

Login banner:

The requirement 2 states that a login banner must be present to warn attackers about legal actions (Defense Information Systems Agency, 2015). This requirement is already in place and implemented by configuration management tools. The category for this requirement will be 1.

Account lockout:

The requirement 3 states that "accounts must be locked upon 35 days of inactivity" (Defense Information Systems Agency, 2015). "Disabling inactive accounts ensures that accounts which may not have been responsibly removed are not available to attackers who may have compromised their credentials" (Defense Information Systems Agency, 2015). This requirement is not in place. The author doubts if it is necessary. These days' systems are managed by configuration management tools, logging manually in to systems is minimal. It conflicts with the parent company internal user management processes, therefore this requirement will be skipped. There is separate process and monitoring mechanism in place for detecting user accounts of people who left the company.

"The operating system must manage information system identifiers for users and devices by disabling the user identifier after an organization defined time period of inactivity" (Defense Information Systems Agency, 2015) states the requirement 96. As stated in the requirement 3 – the parent company has a specific monitoring mechanism in place for detecting users who left the company and who's account is still enabled on the servers, therefore this requirement is in place.

The requirements 133-134 protect against password brute force attacks. For instance, the requirement 133 states that the system must lock accounts after several login failures (Defense Information Systems Agency, 2015). The author's team won't use passwords for logging in to the servers, therefore these requirements are not relevant. As passwords are not used, then the requirements 147, 156-157, 159-163, 197-199, 167-169 are also not relevant.

Public directories:

The requirement 4 states "all public directories must be owned by a system account" (Defense Information Systems Agency, 2015). "Allowing a user account to own a world-writable directory is undesirable because it allows the owner of that directory to remove or replace any files that may be placed in the directory by other users" (Defense Information Systems Agency, 2015). This requirement is in place. The category for this requirement will be 1.

The requirement 118 states “the sticky bit must be set on all public directories” (Defense Information Systems Agency, 2015). “Failing to set the sticky bit on public directories allows unauthorized users to delete files in the directory structure” (Defense Information Systems Agency, 2015). As the author was not sure if this requirement is in place, quick scan was conducted. It came out that this requirement is in place on all servers. The category will be 1.

Syslog:

The requirement 5 mentions that “all Rsyslog generated log files must be owned by *root”* (Defense Information Systems Agency, 2015). This requirement is not in place. The parent company is using Syslog-ng instead of Rsyslog, but the same logic can be implemented. The category for this requirement will be 2.

System command files:

The requirement 6 states that “all system command files must be owned by *root*” (Defense Information Systems Agency, 2015). “System binaries are executed by privileged users as well as system services, and restrictive permissions are necessary to ensure that their execution of these programs cannot be co-opted” (Defense Information Systems Agency, 2015). This requirement is in place. The category for this requirement will be 1. The requirement 7 states “all system command files must have mode *0755* or less permissive” (Defense Information Systems Agency, 2015). Similarly, as in the requirement 6 - it is in place everywhere by default. The category will be same as in the requirement 6. The requirement 6 and 7 can be merged to one requirement as they are related.

Auditd service:

“Audit log files must be owned by *root*” (Defense Information Systems Agency, 2015) to fulfil the requirement 8. The requirement 9 states that “Audit log files must have mode *0640*” (Defense Information Systems Agency, 2015). Attacker could tamper the audit logs if file ownership and permissions are not set up correctly (Defense Information Systems Agency, 2015). These requirements are not in place, but can be easily added.

The requirement 10 describes that “auditing must be enabled at boot by setting a kernel parameter” (Defense Information Systems Agency, 2015). “Auditd takes care of enabling this for all processes which launch after it does, adding the kernel argument ensures it is set for every process during boot” (Defense Information Systems Agency, 2015). This requirement is not in place as Auditd utility is not in use. As the author’s team is not managing */etc/grub.conf* file with configuration management tools and as the Auditd might require a specific Linux kernel and some versions of Auditd might not be compatible with Linux Debian OS then it needs more investigation to implement it properly. The category for this requirement will be 3.

The requirements 51 – 75, 90-93, 164-166 are related to Auditd utility rulesets (Defense Information Systems Agency, 2015). None of the requirements are in place, because Auditd is not installed on any server. The category for all Auditd rulesets requirements will be 2 except for the requirement 71 which is not relevant as the author’s team is not using SELinux. All requirements of auditing system are listed in Appendix 1.

The requirement 94 states “The operating system must back up audit records on an organization defined frequency onto a different system or media than the system being audited” (Defense Information Systems Agency, 2015). Meeting this requirement covers also the requirement 98. It is critical that the logs are replicated to central log server. In case of server compromise, the attacker could tamper local logs integrity (Defense Information Systems

Agency, 2015) . Having logs on central logging server is useful for forensics examination and investigation. In addition, several automated alerting can be done based on logs, therefore fulfilling this request is a must. As the author's team, has already set up central logging server and it is just matter of syslog configuration change, then the category will be 2.

Automated file system mounting tools:

The requirement 11 states that "automated file system mounting tools must not be enabled" (Defense Information Systems Agency, 2015). "New filesystems should not be arbitrarily introduced via the automounter" (Defense Information Systems Agency, 2015). This requirement is in place – the author's team is not using automounters. The category will be 1.

IP forwarding:

The requirement 22 states that "IP forwarding for IPv4 must not be enabled, unless the system is a router" (Defense Information Systems Agency, 2015). "IP forwarding permits the kernel to forward packets from one network interface to another" (Defense Information Systems Agency, 2015). This requirement is in place on all servers by default, therefore the category will be 1.

Library files:

To fulfil the requirement 23 "library files must be owned by *root*" (Defense Information Systems Agency, 2015). "Files from shared library directories are loaded into the address space of processes (including privileged ones) or of the kernel itself at runtime" (Defense Information Systems Agency, 2015). This requirement is in place. The category will be 1.

Mail relay:

The requirement 25 mentions that "mail relaying must be restricted" (Defense Information Systems Agency, 2015). "This ensures Postfix accepts mail messages (such as Cron job reports) from the local system only, and not from the network, which protects it from network attack" (Defense Information Systems Agency, 2015). The author's team is using mail relays only on some servers, but there is ongoing project to decommission these, therefore this requirement is not relevant.

Process core dumps:

The requirement 28 states that "process core dumps must be disabled unless needed" (Defense Information Systems Agency, 2015). "A core dump includes a memory image taken at the time the operating system terminates an application" (Defense Information Systems Agency, 2015). The author thinks that core dumps are essential in debugging C application crashes, therefore this requirement cannot and should not be completely disabled. This needs a discussion, therefore the category will be 3.

***Password, shadow* and *group* files:**

The requirements 37-49 are about hardening *password*, *shadow* and *group* files ownership and permissions (Defense Information Systems Agency, 2015). Most of them are already in place, except the requirement 42 and 49 which forces setting file permissions rights to *0000* on */etc/gshadow* and */etc/shadow* files (Defense Information Systems Agency, 2015). "Failure to give ownership of this file to *root* provides the designated owner with access to sensitive information which could weaken the system security posture" (Defense Information Systems Agency, 2015). The author proposes file permission *640* or less, because these files are owned by *root* by default. To mitigate the risk of password cracking – passwords won't be used in the system. The category will be 2 for the requirement 42, 49 and the category 1 for the rest (37-41, 43-48).

Atd service:

Using the Atd service is not allowed to fulfil the requirement 50 (Defense Information Systems Agency, 2015). "The Atd service could be used by an unsophisticated insider to carry out activities outside of a normal login session, which could complicate accountability" (Defense Information Systems Agency, 2015). This requirement is in place – Atd is disabled on all servers by default.

DCCP Protocol and DHCP Client:

The requirement 80 states that "the Datagram Congestion Control Protocol (DCCP) must be disabled unless required" (Defense Information Systems Agency, 2015). "Disabling DCCP protects the system against exploitation of any flaws in its implementation" (Defense Information Systems Agency, 2015). This requirement is not in place, but can be easily added, therefore the category will be 2. In addition, using the DHCP (Dynamic Host Configuration Protocol) client is not allowed according to the requirement 81 (Defense Information Systems Agency, 2015). The author's team is not using DHCP for setting up network connection on servers. The category for this requirement will be 1.

Graphical desktop:

The requirements 82-84, 135 are about graphical desktop environment (Defense Information Systems Agency, 2015). These are not relevant, because the author's team is not using graphical desktop on any servers.

LDAP Server:

According to the requirement 89 "The openldap-servers package must not be installed unless required" (Defense Information Systems Agency, 2015). This requirement is in place – the author's team is not using LDAP (Lightweight Directory Access Protocol) where it shouldn't be used. The author notes that this package is named Slapd in Debian Linux.

External devices:

The requirement 95 is about disabling access to external devices such as USB drives (Defense Information Systems Agency, 2015). This requirement is partially in place – some servers still allow external devices. Although the risk is low, as the attacker should have physical access to the servers for abusing this. There are multiple guards and other security processes in place to access the datacentres physically. The category will be 2.

Firewalls:

The requirement 97 states that "the operating system must prevent public IPv4 access into an organizations internal networks" (Defense Information Systems Agency, 2015). The parent company has several internal and external firewalls for protecting the internal networks. The requirement 97 is more about host based firewall. As the firewall devices are not managed by author's team then the author thinks it is good idea to have host based firewall as well – this gives better control for the team. Host based firewall can be useful for protecting against several Distributed Denial-of-Service attacks, isolating the server from specific networks in case of incidents and so on. The author is considering using utility Iptables. Fulfilling this requirement covers also the requirement 136. The category for this requirement will be 2.

To fulfil the requirement 183 "The systems local IPv4 firewall must implement a deny-all, allow-by-exception policy for inbound packets" (Defense Information Systems Agency, 2015). There are several external and internal firewalls in place as described in the section when reviewing the requirement 97. In the author's opinion, having additional local firewall

on each server gives more security, but "deny-all" rule might cause other problems and risks. For instance, if the author's team accidently forgets to add valid firewall rule, it could trigger an incident with huge end user impact. In addition, this means that someone should manage and keep the firewall rules up to date. Meeting this requirement might be complicated, therefore discussion is needed. The category will be 3.

Display unsuccessful login attempts:

To meet the requirement 99, users must be informed about failed logins made with their accounts (Defense Information Systems Agency, 2015). This requirement is not in place, but can be easily added.

Enable Postfix service:

The requirement 100 states the Postfix service is required (Defense Information Systems Agency, 2015). The author disagrees with this requirement. The author's team manages over thousand Linux servers, receiving hundreds of e-mails of system events doesn't provide any value. In addition, installing Postfix on every server introduces a new attack vector. If necessary – aggregated automated alerts can be created based on logs. This requirement is not relevant.

Other services and requirements:

To minimize the attack surface, then using Abrtd, Avahi and Bluetooth service is not allowed to fulfil the requirements 76-78 (Defense Information Systems Agency, 2015). In contrast to Avahi, Cron service is required (Defense Information Systems Agency, 2015). This is needed for fulfilling the requirement 79. The requirements 77-79 are already implemented, therefore the category will be 1. The requirement 76 is not relevant, because Abrtd service is running only on Red Hat systems. The requirements 86-88 are also about disabling unnecessary services like Netconsole, Ntpdate and Oddjobd (Defense Information Systems Agency, 2015). The requirement 86 and 88 is in place, but the author was not sure about the requirement 87, therefore a quick scan has been conducted on all servers. It came out that Ntpdate is installed on some servers. Service Ntpdate should be replaced with Ntpd service. The category for the requirement 87 will be 2.

Next, the author points out all requirements which are in place. These requirements are 101-102, 104-111, 120-121, 124-125, 149, 158, 184-185, 192-193, 181-182, 187, 189-191 and 204, therefore the category for these will be 1. The details of these requirements are in Appendix 1.

RDS, SCTP, TIPC protocol:

The requirement 103 states "the Reliable Datagram Sockets (RDS) protocol must not be used unless it is needed" (Defense Information Systems Agency, 2015). This rare network protocol is not automatically loaded to the kernel, but the attacker could change the kernel module configuration and reboot the system (Defense Information Systems Agency, 2015). Therefore, the author thinks that disabling RDS kernel module should be done in the kernel module configuration file as well for extra security and for easier auditing. Same should be done with the requirement 120 for disabling the Transparent Inter-Process Communication (SCTP) protocol (Defense Information Systems Agency, 2015) and requirement 188 for disabling the Transparent Inter-Process (TIPC) protocol (Defense Information Systems Agency, 2015). The category for these will be 2.

SSH service:

The requirements 112-117 are about hardening SSH (Secure Shell) service. “The SSH daemon must ignore *.rhosts* files” (Defense Information Systems Agency, 2015) to fulfil the requirement 112. “SSH trust relationships mean a compromise on one host can allow an attacker to move trivially to other hosts” (Defense Information Systems Agency, 2015). This requirement is in place on most servers, but some legacy servers do not have this requirement implemented. In addition, to meet the requirement 113 logon through SSH without password must be not allowed (Defense Information Systems Agency, 2015). Even though the author’s team is not using passwords for logging in via SSH, this requirement should be implemented for additional layer of security. The requirements 114 and 115 are not in place. The requirements 116 and 117 are about hardening SSH sessions idle timeout that are not in place (Defense Information Systems Agency, 2015). The author proposes that idle timeout should be 15 minutes. The category for all SSH requirements will be 2.

System boot loader configuration:

According to the requirement 122, “the system boot loader configuration files must have mode *0600* or less” (Defense Information Systems Agency, 2015). In addition, the requirement 123 states that “the system boot loader must require authentication” (Defense Information Systems Agency, 2015). These requirements are not in place. The author decided that the requirement 123 is not relevant, because password protecting bootloader configuration complicates management of thousands of servers. The category for the requirement 122 will be 2.

Correct *umask*:

The requirements 126-130 are about setting the correct system default *umask* (Defense Information Systems Agency, 2015). Unauthorized users could read or even write in to files if *umask* is misconfigured (Defense Information Systems Agency, 2015). “The system default *umask* for daemons must be *027* or *022*” (Defense Information Systems Agency, 2015) to fulfil the requirement 126. Fulfilling this requirement is not straightforward, because under */etc/init.d/* there are several daemons which doesn’t have the correct *umask* setting. To mitigate this, Linux Debian daemons configuration file */etc/rcS* can be modified accordingly, but as the author is not managing it with configuration management tools it might require some planning. Also, setting the *umask* to too restrictive can cause issues (Defense Information Systems Agency, 2015). The category for this will requirement will be 3. The requirement 127 and 128 is about hardening Bash and Csh shell (Defense Information Systems Agency, 2015). Shell Csh is disabled on the servers the author is managing, therefore the requirement 128 is not relevant. The requirement 127 is not in place and the category for this will be 2. To fulfil the requirement 129 and 130, *umask* in */etc/login.defs* and */etc/profile* must be set to *077* (Defense Information Systems Agency, 2015). The requirement 130 is not relevant, because in Linux Debian OS, PAM's (Linux Pluggable Authentication Modules) *pam_umask.so* module controls reading the *umask* setting from */etc/login.defs*. The requirement 129 is not in place, because by default the *umask* value in */etc/login.defs* is *022* in Linux Debian OS. The category for this requirement will be 2.

Screen service:

The requirement 131 states “the system must allow locking of the console screen in text mode” (Defense Information Systems Agency, 2015) and that utility Screen must be installed (Defense Information Systems Agency, 2015). The author doesn’t see any benefit of this requirement. The author and his team is not using Screen utility and installing additional package might create additional attack vector, therefore this requirement is not relevant.

Kernel parameters:

The requirements 132, 137-139, 141-146, 150-152, 173-174 are about hardening kernel parameters (Defense Information Systems Agency, 2015). The author scanned all their servers and discovered that none of these requirements are implemented, therefore the category for these requirements will be 2. The details of these requirement can be found in Appendix 1.

The requirement 85 states that "The IPv6 protocol handler must not be bound to the network stack unless needed" (Defense Information Systems Agency, 2015). This requirement is in place. The category for this requirement will be 1.

Simultaneous system logins:

"System must limit users to 10 simultaneous system logins, or a site-defined number, in accordance with operational requirements as limiting simultaneous user logins can insulate the system from denial of service problems caused by excessive logins" (Defense Information Systems Agency, 2015) states the requirement 140. This requirement is not in place and the category will be 2.

Interactive boot:

"The system must not permit interactive boot" (Defense Information Systems Agency, 2015) to fulfil the requirement 148. This requirement is not relevant.

Virtual and serial consoles:

The requirement 153 and 154 is about not allowing *root* user for accessing the system from virtual and serial consoles (Defense Information Systems Agency, 2015). Accessing the servers through console using *root* account might be needed in disaster recovery cases. The category for these requirements will be 3 – it needs further discussion.

Communication over VPN:

To meet the requirement 155, VPN (Virtual Private Network) must be used (Defense Information Systems Agency, 2015). This requirement is not relevant, because none of the author's team servers are placed in untrusted networks. Accessing these servers remotely is already done through VPN.

Linux security module:

The requirements 170-172 are about configuring Linux Security Module such as SELinux. As the author's team is not using SELinux or Grsec, then these requirements are not relevant.

Filesystem:

The requirements 175-179 state that the operating system must use an independent file system for */var*, */var/log*, */tmp*, */home* and */var/log/audit* to make sure that auditing continues working after specific partition is running out of space (Defense Information Systems Agency, 2015). These requirements are not in place. The category will be 3, because fulfilling these requirements need reviewing the logic how virtual machine creation and provisioning is done.

Samba client and TFTP protocol:

The requirement 180 which states "the system must use SMB (Samba) client signing for connecting to Samba servers using Smbclient" (Defense Information Systems Agency, 2015) is not relevant, because the author's team is not using Samba on any servers. Similarly, the requirement 194 "There must be no *.rhosts* or *hosts.equiv* files on the system" (Defense Information Systems Agency, 2015) is not relevant, because the author's team is already disabling *rhosts* in SSHD service configuration as stated in the requirement 112. In

addition, the requirement 186 is not relevant as the author's team is not using TFTP (Trivial File Transfer Protocol) (Defense Information Systems Agency, 2015).

3.3 Results

The goal of the section 3.2 was to identify relevant requirements of the holding company hardening standard and analyse how complicated it is for the parent company to meet these requirements on Debian Linux operating system. According to analysis, it came out that 37 (21%) requirements of 177 are not relevant, 56 (32%) requirements are in place, 73 (41%) requirements are not in place, but can be added easily. For the rest 11 (6%) requirements the solution is not straightforward and needs discussion with other teams. Figure 2 visualizes the results of the holding company hardening standard analysis.

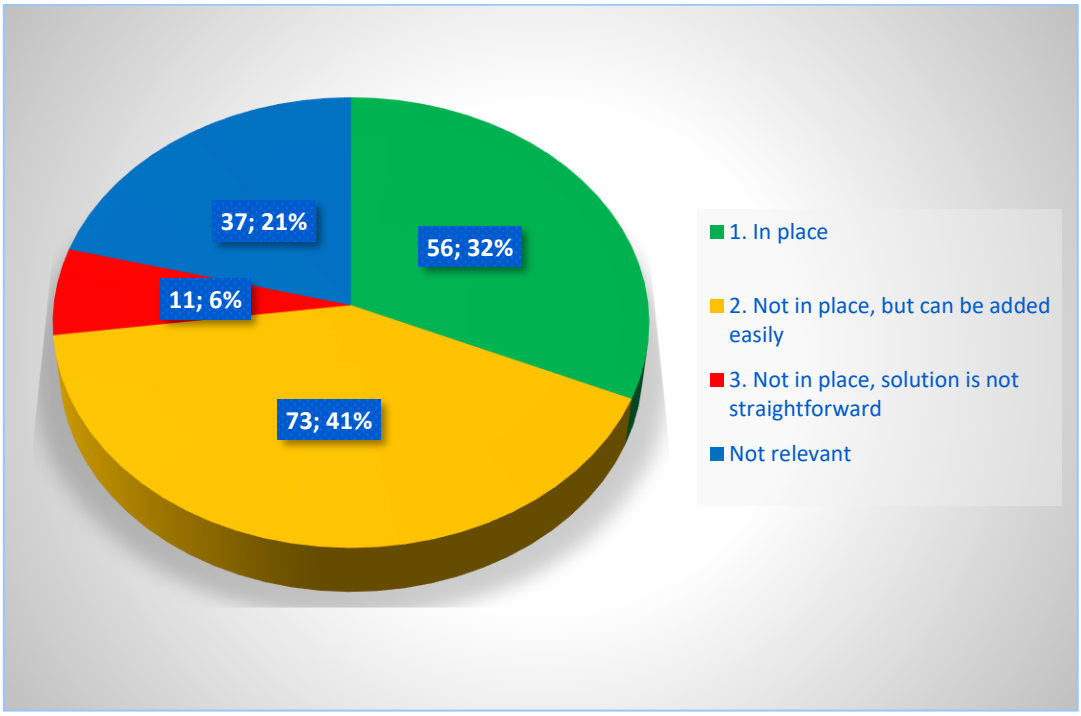

Figure 2. Results of the holding company hardening standard analysis

Based on the results, it was decided that the parent company needs separate Linux hardening standard for Debian Linux operating system, mainly because the holding company hardening standard which is based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015) has too many not relevant requirements for Debian systems. The holding company hardening standard will be taken as a baseline when establishing the new standard.

3.4 Threats to Validity

The main threat to validity is that the holding company hardening standard analysis result is mostly based on work experience of the author. The result could be different if the holding company standard was analysed by different person in the parent company.

3.5 Establishing the Hardening Standard for the Parent Company

In the section 3.2 there are 11 requirements which implementation is not straightforward, therefore discussion was done with other stakeholders and teams as well. The outcome of

the meeting was that, the requirements 153-154 and 183 will not be part of the new hardening standard. The requirements 10, 28, 126 and 175-179 will be added to the new hardening standard as well.

New operating system (Debian Linux) hardening standard for the parent company will be established based on the section 3.2 relevant requirements. The goal is to protect the parent company from cyber threats and to meet the base security compliance requirements from the holding company.

Based on the analysis in section 3.2, the author will categorize the relevant hardening requirements to 9 groups. *The hardening requirement numbers in the brackets refer to the actual requirement in the holding company Linux hardening standard (Appendix 1) which is based on DISA STIG for Red Hat Linux* (Defense Information Systems Agency, 2015).

1. **Secure file and directory permissions and ownership**

The hardening requirements below are based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015)*:*

- Public directories owned by a system account (req. 4)
- All Syslog-ng generated log files owned by *root* (req. 5)
- System command files owned by *root*, have mode *0755* or less permissive (req. 6,7)
- Audit log files owned by *root*, have mode *0640* or less permissive (req. 8,9)
- Shared libraries owned by *root* (req. 23)
- Stricter mode for *password*, *shadow* and *group* files - owned by *root*, mode *0640* or less (req. 37-49)
- Sticky bits for public writable directories (req. 118)
- Bootloader configuration files owned by *root*, mode *0600* (req. 120-122)
- Secure *umask* is set by default for users and daemons (req. 126-127,129)

2. **Security related to auditing**

The hardening requirements below are based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015)*:*

- Auditing must be enabled at boot (req. 10)
- Alert when the audit storage volume approaches capacity (req. 51, 75)
- Audit attempts to change system time (req. 52-56)
- Audit *chmod* (and *fchmod*, *fchmodat*) and *chown* (also *fchown*, *fchownat*, *lchown*) events (req. 57-62, 65)
- Audit *syscalls* to set/change/remove extended attributes (**exattr*) (req. 63-64, 66-69)
- Audit changes to sensitive config files: *sudoers*, *password*, *shadow* (req. 70, 90-93)
- Audit file system mounts (req. 72)
- Audit loading and unloading of kernel modules (req. 73)
- Audit user deletions of files and programs (req. 74)
- Audit logs are forwarded remotely and rotated (req. 94, 98, 164-166)

3. **Secure SSHD configuration**

The hardening requirements below are based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015)*:*

- Display login banner (req. 2, 111)

- Only protocol version 2 is allowed (req. 110)
- Ignore *.rhosts* files (req. 112)
- Do not not allow host-based authentication and authentication using an empty password (req. 113-114)
- Do not permit user environment settings (req. 115)
- Idle timeout is enforced, set for 15 minutes (req. 116-117)
- *Root* login is disabled (req. 149)
- Password based authentication is disabled (only SSH key authentication allowed)

4. Secure kernel (Sysctls) configuration

The hardening requirements below are based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015)*:*

- IP forwarding is disabled (req. 22)
- Enable TCP *syncookies* (req. 132)
- Ignore ICMP (Internet Control Message Protocol) redirects and bogus error responses, don't send ICMP redirects (req. 137-139, 142-144, 151-152)
- Source routed packets are not accepted (req. 145-146)
- ICMP broadcasts are ignored (req. 150)
- Reverse path filter is enabled (req. 173-174)
- Martian packets are logged (req. 141)

5. Security related to packages

The hardening requirements below are based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015)*:*

- Samhain must be installed (req. 1)
- Slapd, Rsh-server, Sendmail, Telnetd, Tftpd-hpa, Xinetd, Openbsd-inetd, Xserver-xorg, Nis packages must not be installed (req. 89, 108-109, 185, 187, 190-191, 193)

6. Security related to services

The hardening requirements below are based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015)*:*

- Autofs, Atd, Avahi-daemon, Bluetooth, Netconsole, Ntpdate, Oddjobd, Qpidd, Rdisc, Rexecd, Rlogind, Rshd, Telnet, Xinetd, Ypbind, Openbsd-inetd, Nis services must be disabled (req. 11, 50, 76-78, 86-88, 101-102, 104-105, 107, 184, 189, 192)
- Cron service must be running (req. 79)

7. Security related to protocols

The hardening requirements below are based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015)*:*

- Disable DCCP, RDS, SCTP, TIPC protocols (req. 80, 103, 119, 188)
- Disable IPv6 protocol handler (req. 85)
- Disable *netconsole* kernel module (req. 86)

8. Secure partitioning layout

The hardening requirements below are based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015)*:*

- Independent filesystem for */tmp*, */var*, */var/log* (req. 175-177)
- Independent filesystem for the Auditd log files and user home directories (req. 178-179)

9. Other requirements that minimize the attack surface

The hardening requirements below are based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015)*:*

- Disable process core dumps (req. 28)
- Disable *usb-storage* kernel module (req. 95)
- Install Iptables (req. 97, 136)
- Show unsuccessful login attempts (req. 99)
- Only *root* account has UID 0 (req. 106). Disable *Root* interactive login
- Time is synchronized using Ntpd. Only internal NTP (Network Time Protocol) servers must be used (req. 124-125)
- Simultaneous system logins limit enforced per user (req. 140)
- All the installed packages are authenticated during installation. Installing unsigned packages are not allowed (req. 181-182)
- Disable X Windows (req. 204)

Linux Debian OS hardened virtual machine image for Microsoft Azure platform will be built to make sure that these requirements are implemented. How this is done, will be described in the next Chapter 4. To ensure the continuous compliance, there will be a solution which allows the parent company to validate the gap between the new standard and actual hardening configurations on servers. This will be described in the Chapter 5.

3.6 Summary

In this chapter, we have analysed whether the holding company hardening standard based on DISA STIG for Red Hat is suitable for the parent company. As per our analysis, it was decided that the parent company needs separate Linux hardening standard for Debian Linux operating system. We have identified all relevant requirements for the parent company Debian Linux systems while analysing the holding company hardening standard. This allowed us to establish the new hardening standard for the parent company which will be used as the baseline in the next chapters.

4 Implement the Operating System Hardening Standard

In this chapter, we are analysing virtual machine image creation tools and choosing suitable tool for the parent company. We are integrating chosen tool with Microsoft Azure platform and configuration management tool Chef to automatically prepare a hardened virtual machine image for proof-of-concept. The hardened image will be prepared based on the new hardening standard which we have established in the section 3.5. As we can't implement all the requirements from the new standard, then in this chapter we are choosing the most critical one's for our proof-of-concept solution. We are also reviewing the related work in this field and describing briefly the benefits of hardened virtual machine image.

In this chapter, we will answer to *RQ3* which is: "How to implement Linux operating system hardening standard for security compliance?" To find answer to this question, we will answer to several sub-questions: "What requirements of the OS hardening standard can be implemented on the parent company Debian Linux operating systems? What virtual machine image creation tool is suitable for the parent company for preparing the hardened base image? What it takes to integrate Microsoft Azure platform, Chef configuration management and chosen image creation tool for automatically implementing the hardening standard?"

4.1 Approach

To achieve the second goal and answer to the *RQ3* that we have stated in the section 1.2, we are first describing the purpose and benefits of the hardened virtual machine image in the section 4.2. To not reinvent the wheel for implementing the proof-of-concept solution, we are reviewing related work in this field and analysing existing open source tools out there (in the section 4.3) that can be used for hardened VM image creation in Microsoft Azure platform. These tools are Azure CLI, Azure Manage and Packer. The author tested these tools on Linux Debian workstation and compared whether they meet the requirements of the parent company. The result of image creation tools analysis is prerequisite for building the hardened VM image. After we have chosen hardening requirements for implementing, we will set up development environment which allows to follow TDD methodology. As Test-Driven Development methodology (Erdogmus, Morisio, & Torchiano, 2005) requires that tests should be prepared first before developing the solution, thus we have prepared the tests based on the hardening requirements which we have chosen in the section 4.4. We have developed these tests in InSpec testing framework – the tool that we are using for automated security compliance auditing in the Chapter 5. These tests are described in the section 5.5 and in Appendix 3. For developing the solution, the Chef *cookbook* project has been created (Appendix 2). This *cookbook* contains all necessary code for automatically implementing the new hardening standard requirements based on the section 4.4. Finally, we have integrated Chef *cookbook* with chosen VM image creation tool and with Microsoft Azure platform. As a result, we will build the new hardened VM image which is based on the new hardening standard that we have established in this thesis.

4.2 Benefits of Hardened Image

According to Azure documentation created by Nottingham, Peterson, Foulds, & Squillace (2016), images in Microsoft Azure are required to provision a new virtual machine with preinstalled operating system and installed software. Based on the author's experience, the standard Microsoft Azure Linux images are not built for security by default nor they are hardened against any security guidelines. Overall, they are made to cater the largest customer base, therefore the author will create their own hardened custom base image for Linux

Debian OS based on the new hardening standard for the parent company as described in the section 3.5.

Creating a custom hardened image is needed for the parent company for several other reasons. Firstly, when using only configuration management tools (Puppet, Chef, Ansible etc.) for bootstrapping new virtual machines directly then there are possibilities that first bootstrap will fail (temporary network issues, packages repository might be unavailable etc.), this leaves new instance as "half secure" and non-compliant because some security features have not been implemented. Secondly, the author's team provides infrastructure for several development teams. Developers might not always take security into account during the build process, therefore having the hardened image gives confidence to the developers that the OS where their code is running is compliant. Thirdly, provisioning a new virtual machine will be much faster if all needed features are already implemented in the custom image. To reduce these risks, pre-hardened custom base image is needed. This increases the parent company confidence that all their Linux virtual machines are compliant - have identical security hardening configurations implemented. This is step forward for passing the holding company internal audit.

4.3 Compare Image Creation Tools

There are several possibilities for creating custom Microsoft Azure images. In this master's thesis, the author is analysing tools named Azure CLI, Azure Manage and Packer. In this section, the goal is to choose suitable image creation tooling for the parent company. The author expects that these tools meet the following requirements:

1. Is open source and works on Linux operating system;
2. Azure ARM (Azure Resource Manager) deployment model is supported;
3. Supports configuration management tool Chef to install software onto the image;
4. Automatically uploads image to Azure Blob storage account;
5. Possible to create Debian Wheezy and Debian Jessie images;
6. Easy to install and good documentation;

Azure Command-Line Interface

According to Kshirsagar, et al. (2017), Azure Command-Line Interface (Azure CLI) is a Windows and Linux tool that provides shell commands for managing resources in Microsoft Azure platform. As described in Microsoft Azure documentation created by Kshirsagar, et al. (2017), installing Azure CLI is not complicated – using Npm package, Windows installer or Docker container. The author is using Npm package installed on Debian Linux. According to Foulds, et al. (2017), with Azure CLI, it is possible to create a new Linux virtual machines, generalize and capture the whole virtual machine and prepare it to be used as an image while using *azure vm generalize* and *azure vm capture* shell commands. As a result, new virtual machines can be created using the custom image. There is a good documentation available how to do it step-by-step (Foulds, et al., 2017).

Although installing Azure CLI is easy, it works on Debian Linux, supports Azure ARM mode, supports Chef and it has a great documentation, then still many additional commands should be executed to create and upload the custom image to Azure storage account. In author's opinion, this takes too much time and requires extra resources for building the complete solution which meets the parent company requirements.

Azure Manage

According to Blank (2017), Azure Manage is a set of scripts for managing Debian images within the Microsoft Azure platform. There is a good documentation available how to install

the dependencies and the tool itself on Debian Jessie (Zarkos, et al., 2017), but in author's opinion how to operate (which commands and parameters should be used) with this tool is poorly documented. This tool supports Azure ARM mode, Debian Wheezy and Jessie image creation, image uploading to Azure storage account (Blank, 2017), but separate tooling is needed for bootstrapping Chef cookbooks and recipes onto the specific image. Overall, it seems to be excellent tool for creating images from scratch, but it requires extra resources for building the solution which meets the parent company requirements.

Packer

According to HashiCorp (2017a), Packer is a tool that can be used for building custom virtual machine images from a configuration file that is defined by a user. It is possible to integrate Packer with multiple configuration management tools such as Chef and integrate it with several cloud platforms such as Azure and AWS (HashiCorp, 2017a). According to HashiCorp (2017b,c) documentation, installing Packer on Linux is very easy, because it's distributed as a binary package. It supports Azure ARM deployment model, Debian Wheezy and Jessie image creation, it automatically uploads ready images to Azure storage account and has an excellent documentation.

Summarized overview of virtual machine image creation tools is listed in Table 1.

Table 1. Results of comparing virtual machine image creation tools

Requirement	Azure CLI	Azure Manage	Packer
Open Source	+	+	+
Azure ARM support	+	+	+
Chef support	+	-	+
Automatic image upload to Azure storage account	-	-	+
Debian Wheezy and Jessie image creation support	+	+	+
Easy to install	+	+	+
Good documentation	+	-	+

Results:

Based on the short analysis, Azure CLI and Azure Manage is not suitable tool for the parent company as they don't support automatic image upload to Azure storage account. Moreover, Azure Manage has bad documentation and it does not have any built-in Chef features for bootstrapping Chef *cookbooks* onto the specific image. Thus, Packer meets all these requirements, therefore it will be chosen as virtual machine image creation tool for the parent company.

4.4 Choosing Hardening Requirements for Implementing

Before we can start with implementing the solution, we need to choose requirements from the new hardening standard what can be applied on our proof-of-concept hardened image. As we can't implement all the requirements in this thesis due to limited resources, then we are choosing the most critical one's and requirements that are technically easier to implement for our proof-of-concept solution.

The following hardening requirement groups from the Chapter 3 section 3.5 will be implemented:

1. Group 1 - Secure file and directory permissions and ownership
2. Group 3 - Secure SSHD configuration
3. Group 4 - Secure kernel (Sysctls) configuration
4. Group 5 - Security related to packages
5. Group 6 - Security related to services
6. Group 7 - Security related to protocols

The exact hardening requirements from these groups are also listed in the section 3.5. As in this thesis we are implementing a proof-of-concept prototype in the testing environment, then we can safely implement any hardening requirement from the above groups.

4.5 Preparing Development and Testing Environment

In this section, we are preparing a new Chef *cookbook* project *hardened-infra* (Appendix 2) which is prerequisite for building the hardened image for the parent company. For Chef *cookbook* development, we will set up development environment using Test Kitchen and integrate it with Microsoft Azure platform, testing and compliance analysis tool InSpec and with Chef. Finally, we will prepare Packer template and build the new hardened image based on the new hardening standard requirements which we have implemented in Chef *cookbook hardened-infra*.

Preparing Test Kitchen

For following the TDD methodology during Chef *cookbook* development, then the author decided to use Test Kitchen tool. "Test Kitchen is an integration tool for developing and testing infrastructure code on isolated target platforms" (Nichol, 2017). This tool allows the author to create test machines, apply configuration management tool changes and test the changes afterwards. After finishing the development, the test machine can be easily destroyed.

In this thesis, we are using a plugin that is suitable for Test Kitchen and Azure ARM (Preston, 2017). We are also using a driver for integrating Test Kitchen with testing framework InSpec (Hartmann, 2017). While using these drivers, we will create test virtual machines to Azure, bootstrap it with Chef *cookbook* after each new feature development and test the changes after.

In a Chef cookbook project (Appendix 2), *kitchen.yml* configuration file should be defined accordingly (See Figure 3).

```
---
driver:
  name: azurerm

driver_config:
  subscription_id: <%= ENV['AZURE_SUBSCRIPTION_ID'] %>
  location: 'West Europe'
  machine_size: 'Standard_A0'

transport:
  ssh_key: /home/martin/.ssh/dummychefsolokey

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: debian-7
    driver_config:
      image_urn: Credativ:Debian:7:latest
      vm_name: martin-testkitchen-wheezy

suites:
  - name: default
    run_list:
      - recipe[hardened-infra::default]
```

Figure 3. Test Kitchen configuration file that was prepared with the help of documentation of Test Kitchen plugin for Azure (Preston, 2017)

Test Kitchen configuration file in Figure 3 is used for creating a test virtual machine running on Linux Debian Wheezy with minimum *Standard_A0* size (1 CPU, 0.768 GB RAM, 20GiB HDD) to Azure cloud platform. This test machine is bootstrapped with Chef *cookbook* named *hardened-infra* (Appendix 2). In this thesis, we are using this test machine for implementing the hardening requirements from the section 4.4.

Developing Chef Cookbook

For preparing a new hardened Debian virtual machine image, the author has developed a new Chef *cookbook* named *hardened-infra* (Appendix 2). This *cookbook* contains all necessary code for automatically implementing the new hardening standard requirements based on the section 4.4. Separate Chef *cookbook* is needed mostly for two reasons. Firstly, it is used to apply needed changes onto the virtual machine image in Azure with help of Packer tool (using *chef-solo* or *chef-client* provisioner). How this is done will be described in the next section. Secondly, the author's team has servers outside of Azure platform (for various compliance reasons), where Azure images are not compatible. These servers are bootstrapped with Chef directly. Basically, having a separate Chef *cookbook* gives us the confidence that applied configuration is the same on all servers – on virtual machines in Azure and outside of Azure.

Each OS hardening requirement group (listed in the section 4.4) has a separate Chef *recipe* (Ruby code snippet) in *hardened-infra* Chef *cookbook*. For example, the Chef *recipe packages.rb* removes and installs packages (See Figure 4).

```
# 5. Security related to packages
# Packages to remove
blacklist = %w(
  slapd
  rsh-server
  sendmail
  telnetd
  tftpd-hpa
  xinetd
  openbsd-inetd
  xserver-xorg
  nis)
blacklist.each do |removepackage|
  package removepackage.to_s do
    action :purge
  end
end
# Packages to install
whitelist = %w(samhain)
whitelist.each do |installpackage|
  package installpackage.to_s do
    action :install
  end
end
```

Figure 4. Chef *recipe* for removing and installing packages

The Chef *recipe services.rb* stops and starts needed services according to the new hardening standard (See Figure 5).

```
# 6. Security related to services
# Services to stop
blacklist = %w(
  autofs
  atd
  avahi-daemon
  bluetooth
  ntpdate
  oddjobd
  qpidd
  xinetd
  openbsd-inetd
  nis)
blacklist.each do |stopservice|
  service stopservice.to_s do
    action [:stop, :disable]
    # Error is triggered in Wheezy if there is no such service, we ignore it
    ignore_failure true
  end
end
# Services to start
whitelist = %w(cron)
whitelist.each do |startservice|
  service startservice.to_s do
    action :start
  end
end
```

Figure 5. Chef recipe for stopping and starting services

All the rest of the Chef *recipes* for implementing the hardening requirements based on the section 4.4 are available in Appendix 2.

Preparing Packer template

In this section, the author is preparing a Packer template for building the hardened Linux Debian Wheezy virtual machine image for proof-of-concept solution.

For preparing the hardened virtual machine image, the author installed Packer on Linux Debian workstation. After that, we have followed instructions from Packer documentation (HashiCorp, 2017d) to prepare configuration (template) that describes how to build custom VM images (HashiCorp, 2017d). According to HashiCorp (2017c,e), several configuration parameters should be defined before building VM image for Azure. The following configuration parameters are required for authenticating against Azure API as described in Figure 6: *subscription_id*, *client_id*, *client_secret*, *resource_group_name* and *storage_account* (HashiCorp, 2017c,e). The author used a setup script (Bednarski, 2017) which automatically sets up specific Azure resources and exports necessary JSON configuration parameters and values for Packer. According to HashiCorp (2017e), the author had to set up *capture_container_name*, *capture_name_prefix*, *image_publisher*, *image_offer*, *image_sku*, *location*, *vm_size* and *os_type* parameters and values for ARM builder as described in Figure 6 under "builders" section.

```
1   {
2       "variables": {
3           "client_id": "{{env `ARM_CLIENT_ID`}}",
4           "client_secret": "{{env `ARM_CLIENT_SECRET`}}",
5           "resource_group": "{{env `ARM_RESOURCE_GROUP`}}",
6           "storage_account": "{{env `ARM_STORAGE_ACCOUNT`}}",
7           "subscription_id": "{{env `ARM_SUBSCRIPTION_ID`}}",
8           "tenant_id": "{{env `ARM_TENANT_ID`}}",
9           "ssh_user": "packer",
10          "ssh_pass": "{{env `PACKER_TEMP_PW`}}"
11      },
12      "builders": [{
13          "type": "azure-arm",
14          "client_id": "{{user `client_id`}}",
15          "client_secret": "{{user `client_secret`}}",
16          "resource_group_name": "{{user `resource_group`}}",
17          "storage_account": "{{user `storage_account`}}",
18          "subscription_id": "{{user `subscription_id`}}",
19          "tenant_id": "{{user `tenant_id`}}",
20          "capture_container_name": "images",
21          "capture_name_prefix": "packer",
22          "ssh_username": "{{user `ssh_user`}}",
23          "ssh_password": "{{user `ssh_pass`}}",
24          "os_type": "Linux",
25          "image_publisher": "credativ",
26          "image_offer": "Debian",
27          "image_sku": "7",
28          "ssh_pty": "true",
29          "location": "West Europe",
30          "vm_size": "Standard_A1"
31      }],
```

Figure 6. Packer template – variables and builders section. Template has been prepared with the help of Packer documentations (HashiCorp, 2017c,d,e)

Figure 7 describes *shell* and *chef-solo* provisioner sections. In Figure 7 the Packer *shell* provisioner (HashiCorp, 2017g) sets up prerequisites for Chef and upgrades all system packages to latest version. The *chef-solo* Packer provisioner (HashiCorp, 2017f) installs *hard-*

ened-infra Chef *cookbook* which applies necessary hardening changes from section 4.4. Finally, the Packer *shell* provisioner (HashiCorp, 2017g) is used for executing *waagent deprovision* command that does necessary changes to the system for reprovisioning (HashiCorp, 2017e).

```
    "provisioners": [{
        "type": "shell",
        "execute_command": "echo '{{user `ssh_pass`}}' | {{ .Vars }} sudo -S -E sh '{{ .Path }}'",
        "inline": [
            "echo 'packer ALL=NOPASSWD:ALL' > /etc/sudoers.d/waagent",
            "apt-get update",
            "apt-get upgrade -y",
            "apt-get install -y curl"
        ],
        "inline_shebang": "/bin/sh -x"
    }, {
        "type": "chef-solo",
        "cookbook_paths": "/home/martin/devel/chef/cookbooks",
        "data_bags_path": "/home/martin/devel/chef/infrastructure_chef_web/data_bags",
        "run_list": ["recipe[users::users]", "recipe[hardened-infra::default]"],
        "json": {
            "users": {
                "databags": ["operations"]
            }
        },
        "skip_install": false
    }, {
        "type": "shell",
        "skip_clean": true,
        "execute_command": "echo '{{user `ssh_pass`}}' | {{ .Vars }} sudo -S -E sh '{{ .Path }}'",
        "inline": [
            "/usr/sbin/waagent -force -deprovision+user && export HISTSIZE=0 && sync",
            "apt-get remove -y --purge curl",
            "apt-get autoremove -y --purge",
            "apt-get clean"
        ],
        "inline_shebang": "/bin/sh -x"
    }]
}

```

Figure 7. Packer template – Chef and Shell provisioners. Template has been prepared with the help of Packer documentations (HashiCorp, 2017c,d,e,f,g)

In the next section 4.6 we are building the hardened virtual machine image using the Packer template which we have prepared in this section.

4.6 Building Hardened Virtual Machine Image

Before building the hardened virtual machine image with Packer, the *packer validate* command should be used for verifying configuration and the syntax of a template (HashiCorp, 2017d). For triggering the build, we are executing *packer build* command (HashiCorp, 2017d). Packer build prepares an Azure template (JSON file) and a VHD (Virtual Hard Disk) image file (HashiCorp, 2017e). Figure 8 shows that the hardened Debian Wheezy image was built and uploaded to Azure storage account.

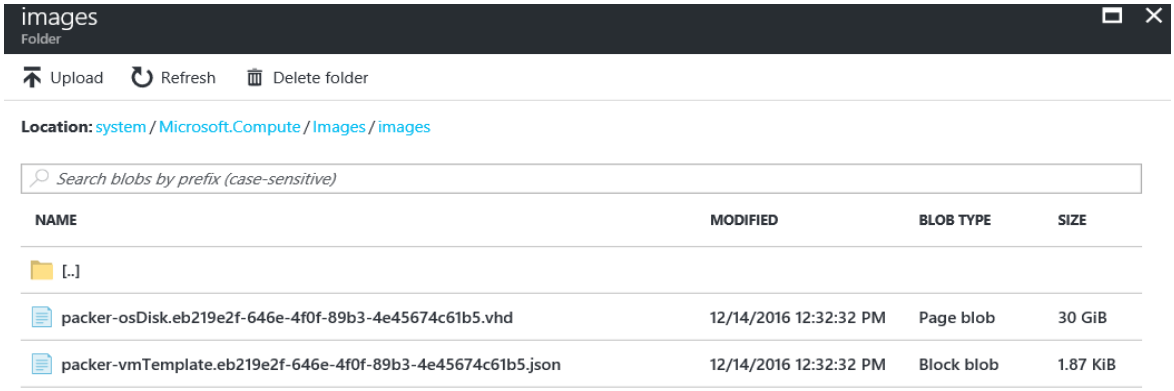

Figure 8. The hardened Debian Wheezy image built by Packer (view from Azure Portal)

For testing the new custom hardened Linux Debian Wheezy image created by Packer, the author has created a new virtual machine based on the new image using Azure CLI utility. See the command below:

```
azure vm create westeurope-martin-packer2 -l "westeurope" --resource-group packerbuild1337 --image-urn https://packerbuild1337.blob.core.windows.net/system/Microsoft.Compute/Images/images/packer-osDisk.eb219e2f-646e-4f0f-89b3-4e45674c61b5.vhd -u martinj -M ~/.ssh/id_rsa.pub -y Linux --storage-account-name packerbuild1337 -f packernic1
```

After logging in to this virtual machine, the author confirmed that necessary hardening changes have been implemented. In the next Chapter 5, we have implemented a solution that automatically verifies if the hardening changes are implemented to ensure that the parent company is compliant with security compliance requirements of the holding company.

4.7 Related Work in this Field

Despite the importance of security compliance and OS hardening, the author has discovered that little research has been done in this field, especially when researching automation possibilities for implementing hundreds of security controls from specific OS hardening standard.

Montesino & Fenz (2011a,b) have published an interesting research paper that analyses how many security controls can be automated based on ISO 27001 and NIST SP800-53 standards. It concentrates on automating technical and procedural security controls as well. The researchers have concluded that 30% of the security controls included in these standards can be automated by using existing tools (Montesino & Fenz, 2011a,b). The main differences between these papers and this thesis are that they are not implementing any automated solution. Instead, they are listing potential tools that can be used for automatically implementing security controls and they are analysing how many security controls can be automated. Moreover, they are taking different security standard as a baseline.

According to Bird (2016), there are examples available for harden Linux systems using configuration management tools like Puppet, Chef and Ansible. One of the example is Chef *cookbook* (Richter, 2017a) that is based on NSA Red Hat standard (National Security Agency, 2007). Similar Puppet module (Richter, 2017b) and Ansible *playbook* is available as well (Gumprich, 2017). After reviewing these solutions, we have decided that these are not suitable for the parent company, because these solutions have limited requirements implemented and they are not fully based on DISA STIG for Red Hat.

In addition, we have investigated if there are pre-built hardened virtual images available for provisioning instances in the cloud. CIS is providing hardened virtual images for the Amazon Web Services (AWS) cloud platform (Center For Internet Security, 2017b). These images are hardened according to CIS standards (Center for Internet Security, 2017a). This is not suitable for the author, because the author's team is using Microsoft Azure cloud. Besides that, CIS hardened image for AWS is not free and we are using different OS hardening standard as a baseline.

4.8 Summary

In this chapter, we have reviewed related work in this field. We discovered that not much academic research has been done in this field. For better understanding for the readers, we briefly described why hardened image is needed for the parent company. Before we could start with implementing the proof-of-concept solution, we have analysed three open source tools which can be used for image creation – Azure CLI, Azure Manage and Packer. As per our analysis, it was decided that Packer is the most suitable image creation tool as it met all the requirements of the parent company. We have chosen the most critical requirements from the new hardening standard which we have applied in the new hardened image for Linux Debian Wheezy. Before building the hardened image, we have prepared a Chef *cookbook* which contains all the code for automatically implementing the hardening requirements. For that, we have prepared development and testing environment using Test Kitchen which allowed us to integrate Microsoft Azure platform and configuration management tool Chef during Chef *cookbook* development. Finally, we have prepared Packer template where we integrated Packer with Microsoft Azure platform and Chef configuration management tool. As a result, we have built the new hardened image which is based on the new hardening standard.

5 Auditing the New Hardening Standard

In this chapter, we are reviewing related work in security compliance auditing field. We are reviewing several research papers and potential compliance tools for automated auditing. In addition, we are briefly describing the importance of automated compliance auditing for the parent company. Finally, we are choosing automated compliance auditing tool for the parent company and proof-of-concept solution will be implemented by developing security compliance auditing tests which are based on the new hardening standard.

In this chapter, we will answer to *RQ4* which is: "How to audit Linux operating system hardening standard for security compliance?" To find answer to this question, we will answer to several sub-questions: "What solutions are there that can be used for the automated security compliance auditing based on the company OS hardening standard? What tool is suitable for the parent company? What it takes to develop security compliance checks (tests) for the OS hardening requirements which we have implemented in this project?"

5.1 Approach

To achieve the third goal and answer to the *RQ4* that we have stated in the section 1.2, we are first describing the purpose and the importance of the automated security compliance auditing in the section 5.2. To not reinvent the wheel for implementing the proof-of-concept auditing solution, we are reviewing related work in this field and analysing existing open source tools out there (in the section 5.3) that can be used for automated security compliance auditing. We have reviewed tools such as OpenScap, ServerSpec, InSpec, Rspec and audit mode in Chef Client. Due to limited resources and time, we had no possibility to test hands-on experience of all tools. The suitable tool was mostly chosen based on the existing literature, related work, documentations, requirements of the parent company and work experience of the author. Automated security compliance auditing tool is chosen for the parent company and security compliance tests (controls) are developed for each OS hardening requirement from the section 4.4 using the same development environment that we have set up in the section 4.5. As a result, the proof-of-concept solution is implemented for automatically auditing the new hardening standard that we have established in this thesis.

5.2 The Importance of Automated Auditing

Even though the new hardening standard is now established and implemented, then there is still a gap how to prove to the auditors that the parent company meets the new hardening standard and is compliant from security perspective. It is not enough to just meet the holding company regulatory requirements, we need to ensure auditability of it.

The parent company has over thousand Linux servers and there is no automated solution for security compliance auditing, therefore it requires many human resources to collect data for the auditors which is not cost effective for the company. Besides saving time and resources, automated and continuous security compliance auditing gives the confidence that all the servers are hardened as per standard – any misconfiguration will be detected and fixed before providing evidence to the auditor.

Moreover, the parent company is doing hundreds of production changes every month. Automated security compliance auditing makes sure that these changes are also compliant with the new hardening standard.

The parent company has several annual audits, where auditor comes on-site and requires real time evidence of specific OS configuration. Automated real time security compliance

auditing will save time for the all stakeholders – it is just matter of triggering the automation and auditors will get the compliance state results automatically.

To solve this cap, the author will choose the suitable tool for automated security compliance auditing in the next section 5.3.

5.3 Choosing the Tool for Automated Compliance Auditing

The author expects that the new security compliance auditing tool meets the following requirements of the parent company:

1. It is open source and it can be easily installed;
2. It requires no configuration overhead during set up;
3. It supports Linux Debian OS;
4. It supports automated security compliance auditing on local servers (running as an agent);
5. It supports automated security compliance auditing on remote servers via SSH;
6. It has human readable language, so that auditors and management can understand the security compliance auditing tests;
7. It can be used as testing tool for integration testing;
8. Test development should be straightforward;
9. It has a good documentation;
10. It should provide human readable report of the scanning results;
11. It should provide report in JSON, XML or other format for automation purposes;
12. It can be triggered any time during evidence providing for auditors;

As the security compliance auditing tool must be open source, then we are not reviewing tools such as QualysGuard, Nessus, Nexpose, Chef Compliance and other enterprise tools.

Although we have discovered in the section 5.6 that several researches have been done using SCAP based tools, then the author decided that SCAP based solutions do not meet the requirements of the parent company. Especially it does not meet the requirement 6 and 8 as the XCCDF is not user friendly to read and to develop in the author's opinion.

During our project, we have analysed several open source tools that can be used for automated security compliance auditing such as SCAP based solution OpenScap that we discovered while reviewing related work in the section 5.6. We have also analysed testing frameworks and tools such as ServerSpec, InSpec and Rspec. In addition, we have tried audit mode feature in Chef Client. After comparing these with each other, we have chosen InSpec testing framework as suitable tool for the parent company. The reason behind choosing this tool is that it met all the requirements of the parent company. Especially that its language is easy to read and develop, it's built for automated security compliance auditing and continuous testing and allows to write security compliance requirements in to code (Chef Software Inc, 2017a). As InSpec is the tool we are using for security compliance auditing, then we are briefly describing it in the next section 5.4.

5.4 Compliance as Code and InSpec

In this section, we are describing the meaning of „Compliance as Code" and presenting the open source "Compliance as Code" testing framework and security compliance auditing tool InSpec.

Bird (2016) states that "DevOps can be followed to achieve what Justin Arbuckle at Chef calls "Compliance as Code": building compliance into development and operations, and

wiring compliance policies and checks and auditing into Continuous Delivery so that regulatory compliance becomes an integral part of how DevOps teams work on a day-today basis." (p. 69). There is an interesting interview article where the term "compliance as code" is explained by Justin Arbuckle (Arbuckle, 2014). According to Arbuckle (2014), it is advised to address security compliance requirements from the beginning of new product development. We agree with this statement, therefore we are following this DevOps approach in this thesis and using the tool InSpec for converting OS hardening requirements into code. We have used this approach while developing Chef *cookbook* for implementing the OS hardening requirements (the section 4.5) by writing tests first for any OS hardening requirement. These tests will be also used for continuous security compliance auditing. We have developed these tests in the next section 5.5.

"InSpec is an open-source testing framework for infrastructure with a human-readable language for specifying compliance, security and other policy requirements" (Chef Software Inc, 2017a). During our research, we discovered that it meets all the requirements of the parent company that we have listed in the section 5.3. Besides that, InSpec has multiple built-in resources that simplifies tests creation and it is possible to add additional information to any security compliance auditing test, so that everyone can understand the requirements (auditors, management etc.) " (Chef Software Inc, 2017a). For better understanding, the author has created InSpec profile named *inspec-hardened-infra*. The author describes the default example test (also called as control) that is created automatically by default after any InSpec profile creation. This example (*example.rb*) is presented in Figure 9.

```
1  control 'tmp-1.0' do
2    impact 0.7
3    title 'Create /tmp directory'
4    desc 'An optional description...'
5    describe file('/tmp') do
6      it { should be_directory }
7    end
8  end
```

Figure 9. Example of InSpec test and its language (Chef Software Inc, 2017b)

Line 1 is a unique ID for the control. Line 2 defines the criticality if the control fails. Line 3 and 4 is a human readable title and description of the control. Line 5-7 is the actual test block. InSpec built-in resource named *file* is used in line 5. File location is set to */tmp* directory. Line 6 contains the test which expects that the file in location */tmp* is a directory (Chef Software Inc, 2017b). In Figure 10, successful test result is presented after executing InSpec profile *inspec-hardened-infra* on Linux workstation.

```
Profile: InSpec Profile (inspec-hardened-infra)
Version: 0.1.0
Target:  local://

  ✔  tmp-1.0: Create /tmp directory
     ✔  File /tmp should be directory

Profile Summary: 1 successful, 0 failures, 0 skipped
Test Summary: 1 successful, 0 failures, 0 skipped
```

Figure 10. Results after executing example InSpec profile

InSpec has other useful features which we are not describing in this thesis. Based on our hands-on experience, we can say that InSpec is a very simple testing framework which can be used for security compliance auditing. In the next section 5.5, we are developing InSpec tests for each OS hardening requirement from the section 4.4.

5.5 Developing Security Compliance Auditing Tests

We have developed InSpec tests for each OS hardening requirement from the section 4.4. For example, Figure 11 describes security compliance auditing tests for verifying if the new OS hardening standard requirements from the group 5 (Security related to packages) are met.

```
control '5. Security related to packages' do
  impact 1.0
  title 'Only approved packages are allowed'
  blacklist = %w(
    slapd
    rsh-server
    sendmail
    telnetd
    tftpd-hpa
    xinetd
    openbsd-inetd
    xserver-xorg
    nis)
  whitelist = %w(samhain)
  # Not allowed packages
  blacklist.each do |removepackage|
    describe package(removepackage.to_s) do
      it { should_not be_installed }
    end
  end
  # Allowed packages
  whitelist.each do |installpackage|
    describe package(installpackage.to_s) do
      it { should be_installed }
    end
  end
end
```

Figure 11. InSpec tests – 5. Security related to packages

Figure 12 presents the tests for verifying if SSHD configuration is hardened according to the new standard. All the rest of the tests are available in Appendix 3.

```
control '3. Secure SSHD configuration' do
  impact 1.0
  title 'Only approved SSH parameters are allowed'
  describe sshd_config do
    its('Protocol') { should eq '2' }
    its('Banner') { should eq '/etc/banner' }
    its('IgnoreRhosts') { should eq('yes') }
    its('PermitEmptyPasswords') { should eq('no') }
    its('HostbasedAuthentication') { should eq('no') }
    its('PermitUserEnvironment') { should eq('no') }
    its('ClientAliveCountMax') { should eq('0') }
    its('ClientAliveInterval') { should eq('900') }
    its('PermitRootLogin') { should eq('no') }
    its('PasswordAuthentication') { should eq('no') }
  end

  describe file('/etc/banner') do
    it { should exist }
  end
end
```

Figure 12. InSpec tests – 3. Secure SSHD configuration

These tests allowed the author to follow TDD methodology as well while implementing the OS hardening standard in the Chapter 4. Moreover, we are using these tests for automatically and continuously auditing the new OS hardening standard of the parent company. For developing security compliance auditing tests, the author used the same development and testing environment using Test Kitchen as described in the section 4.5. *While developing security compliance auditing tests, we have used DISA STIG for Red Hat* (Defense Information Systems Agency, 2015) *auditing guidelines where possible, such as Linux commands what to execute for verifying if specific security requirement has been implemented*.

To validate the gap between the new OS hardening standard and actual hardening configuration on servers on regular basis, we have reused the InSpec profile *inspec-hardened-infra* that we created in the section 5.4. After we have finished the development and testing using Test Kitchen, we added InSpec tests into InSpec profile *inspec-hardened-infra*. Later, we discovered that it is not needed as Test Kitchen can be integrated with remote InSpec profiles, so that it is not needed to keep the same tests in two separate places. This profile is used to execute InSpec controls (tests) on any remote server via SSH. After execution, results are provided automatically whether the server is compliant according the new OS hardening standard. We will analyse the results in the next Chapter 6.

5.6 Related Work in this Field

The author has discovered that there are several researches done in security compliance auditing field, especially in cloud computing area. For example, there is a research done about building automated security compliance tool using CloudAudit frameworks and OpenVAS vulnerability scanning tool on OpenStack cloud platform (Ullah, Ahmed, & Ylitalo, 2013). Doelitzscher (2014) and Bleikertz (2010) came up with a solution for auditing cloud platforms. The main difference between these papers and this thesis is that they concentrate on auditing the cloud platform, not the virtual machines running on the cloud platform with relevant standards as we do in this thesis.

Koschorreck (2011) did a research about the OVAL (Open Vulnerability and Assessment Language) and XCCDF languages and how they can be used for SCAP-based solution for automated security compliance auditing. Koschorreck (2011) used a tool UPW Compliance Guard for implementing security compliance auditing tests for specific security controls.

Montesino (2011b) and Fenz (2011b) also concluded that SCAP can be used for automating security compliance auditing (Montesino & Fenz, 2011b). These research papers are useful sources, because the holding company hardening standard is based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015) which is published as XCCDF as well. Therefore, the author considered using SCAP-based solution as a potential security compliance auditing tool and reuse the XCCDF XML file published by DISA. There are several SCAP based products available validated by NIST (The National Institute of Standards and Technology, 2017c). However, after analysing security compliance analysing tools in the section 5.3, the author decided that SCAP-based tools do not meet the parent company requirements, therefore we are not using SCAP-based tool for automated security compliance auditing in this thesis. Instead, we decided that testing framework and security compliance auditing tool InSpec is suitable for the parent company.

5.7 Summary

In this chapter, we have reviewed related work in this field. We have discovered that there are several research papers available related to automated security compliance auditing. We

have briefly described why automated security auditing is important and needed for the parent company. We have reviewed several open source tools that can be used for automated security compliance auditing such as SCAP-based tool OpenScap, that we discovered while reviewing related work in this field, testing frameworks ServerSpec, InSpec, Rspec and audit mode in Chef Client. InSpec "Compliance as Code" testing framework was chosen as it met all the requirements of the parent company. Moreover, we have briefly described the meaning of DevOps approach "Compliance as Code" which we have followed while implementing the proof-of-concept solution. In addition, InSpec testing framework and its use cases have been described. Finally, we have developed security compliance auditing tests using InSpec for automatically auditing the new OS hardening standard.

6 Validation of Proof of Concept Solution

In this chapter, we are validating the proof-of-concept solution that we have implemented in this thesis. The author is describing the methodology and environment that is used for validating. Finally, we will analyse and write summary of the findings.

In this chapter, we will answer to *RQ5* for achieving the fourth goal in the section 1.2. The *RQ5* is: "How many OS hardening requirements each chosen virtual machine (VM) meets from the new OS hardening standard?" For validating the proof-of-concept solution, we will find answer to how many OS hardening requirements are met on a virtual machine that was created with hardened and non-hardened VM image.

6.1 Methodology and Validation Question

For validating the proof-of-concept solution – Linux Debian VM (virtual machine) image that is hardened according to the new OS hardening standard, the author will choose two virtual machines from the parent company server park in Azure. One server is created with default (non-hardened) VM image and the second one is created with hardened VM image that was built in the section 4.6. For validation, the author decided to measure *how many OS hardening requirements each chosen VM meets from the new OS hardening standard?* Security compliance tool InSpec and its profile *inspec-hardened-infra* that we have developed in the section 5.5 is used for comparing and analysing the results automatically. This allows the author to validate the functionality of chosen automated security compliance auditing tool as well. Figure 13 visualizes the proof-of-concept solution validation steps.

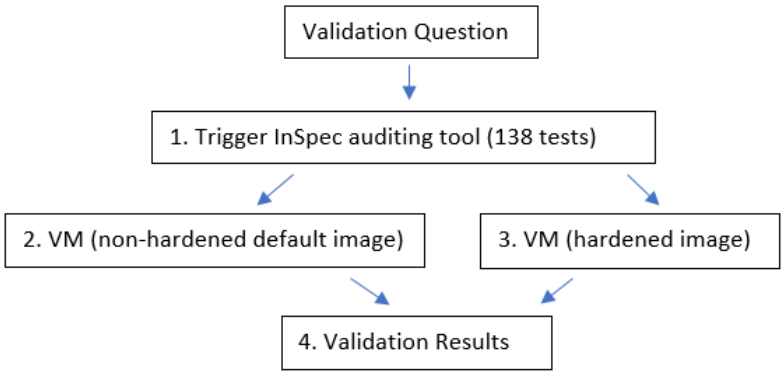

Figure 13. Proof-of-concept solution validation steps

The environment for validating is simple – the author installed InSpec on his Linux workstation. InSpec profile (Appendix 3) is used to execute InSpec auditing controls (tests) on chosen remote servers via SSH in Azure. There is InSpec control for each OS hardening requirement group that we have listed the section 4.4. In this case, it means that 138 tests are executed on each VM. We are analysing the results of InSpec scan results in the next section 6.2.

6.2 Results

The automated security compliance auditing tool InSpec detailed scan results are available in Appendix 4. We have validated if hardening requirements in the section 4.4 are implemented and in place on each VM. Figure 14 presents the InSpec scan results based on a VM that was created with default (non-hardened) VM image. Figure 15 presents the InSpec scan results based on a VM that was created with hardened VM image that we have built in this thesis.

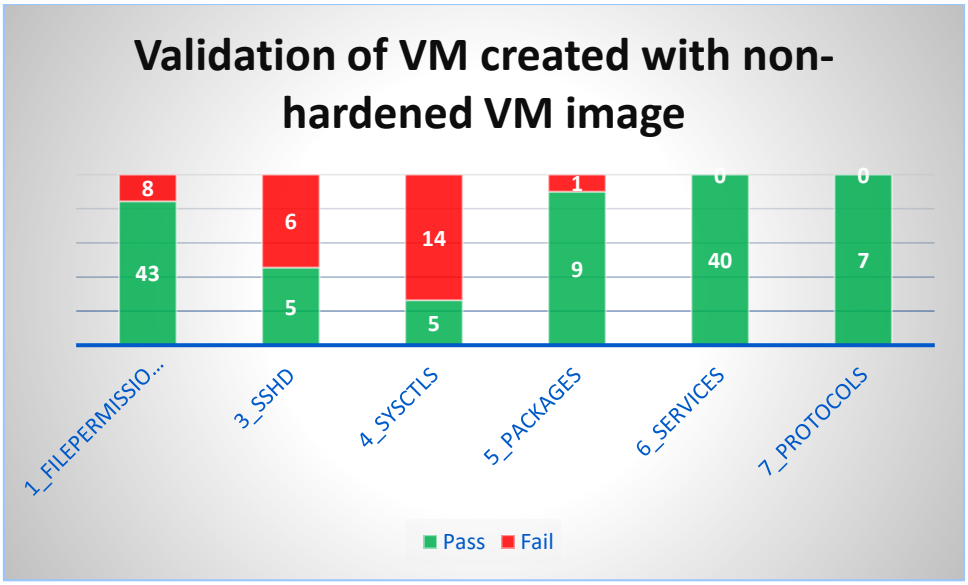

Figure 14. InSpec scan results based on non-hardened virtual machine

It came out that 29 tests of 138 (~21%) did not pass and 109 tests (~79 %) did pass on non-hardened VM. This means that this VM is not compliant with 29 OS hardening requirements from the new OS hardening standard. To be more detailed, 8 requirements from the group „Group 1 - Secure file and directory permissions and ownership" are not in place, such as correct file and owner permissions on */etc/group*, */etc/gshadow*, */etc/passwd* files and correct *umask* value in *login.defs* configuration file. The SSHD configuration file (Group 3 - Secure SSHD configuration) is not compliant with 6 OS hardening requirements. Configuration values such as *PermitUserEnvironment*, *ClientAliveCountMax*, *ClientAliveInterval*, *PermitRootLogin* are not compliant with the new OS hardening standard. In addition, SSH login banner is missing. Linux Kernel configuration (Group 4 - Secure kernel Sysctls configuration) is not compliant with 14 requirements such as *net.ipv4.conf.all.log_martians*, *net.ipv6.conf.all.disable_ipv6*, *net.ipv6.conf.default.disable_ipv6* and other parameters that are listed in Appendix 4. There are no packages installed that are not compliant (Group 6 - Security related to services), although file integrity monitoring tool Samhain is not installed, therefore 1 requirement in this group is not compliant with the new standard. The OS hardening requirements from the other groups (Group 6 - Security related to services and Group 7 - Security related to protocols) are all met.

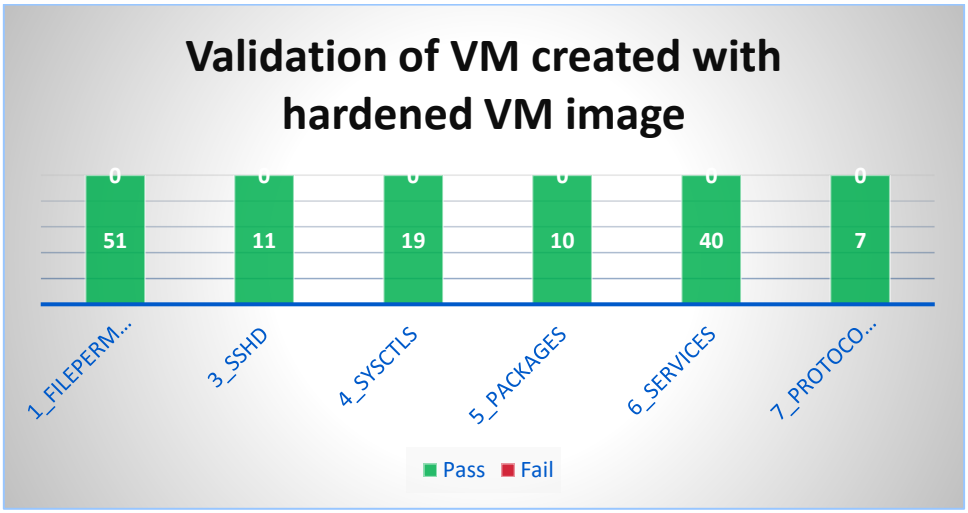

Figure 15. InSpec scan results based on hardened virtual machine

We can see that VM created with the new hardened image meets all the hardening requirements that we have taken in to scope. All OS hardening requirements from the chosen control groups (138 InSpec tests in total) are applied on hardened VM.

6.3 Discussion

Based on the results from the previous section, it is visible that our proof-of-concept solution improved the security compliance percentage of the parent company by ~21 % (based on example of one virtual machine). We discovered that virtual machine that was created with a default VM image did not comply with 29 hardening requirements from the new OS hardening standard. In contrast to VM with default VM image, VM with hardened image is compliant with all the requirements from the new OS hardening standard. We have validated the proof-of-concept solution that improves the situation for being compliant with the holding company security compliance requirements. In addition, we have also confirmed that standard Microsoft Azure Linux Debian images are not built for security by default nor they are hardened against any security guidelines.

6.4 Threats to Validity

One of the main threats to validity is the count of virtual machines that we are validating. Due to limited resources and time, we will not measure the results of the whole server park of the parent company, instead we are taking into scope two virtual machines – one created with a new hardened VM image and the second one created with the default (non-hardened) VM image. The results could be different while auditing the whole server park.

The second threat is that we are taking only the hardening requirements in to scope that were chosen in the section 4.4. Therefore, the parent company might not be compliant with other hardening requirements from the new standard.

Third threat is that we have implemented proof-of-concept solution in this thesis which means that this solution has not been actively tested, therefore some issues could pop up that we are not aware of and affect the automated security compliance auditing results (during implementation we haven't noticed any).

Last but not least threat to validity is a cloud vendor. As the virtual machines are hosted in Azure cloud, then the parent company has no control over Azure cloud availability and reliability. Incidents caused by cloud vendor, could also affect the validation results.

6.5 Summary

In this chapter, we have validated our proof-of-concept solution - a hardened Linux Debian VM image and security compliance auditing tool InSpec. For that, we have created two virtual machines – one with default (non-hardened) VM image and one with hardened VM image that we have built in the Chapter 4. We have described the methodology for validation and answered to the validation question: *how many OS hardening requirements each chosen VM meets from the new OS hardening standard?* According to InSpec scan results, we discovered that 29 hardening requirements (~21%) of 138 are not met on VM that was created with non-hardened image. Virtual machine that was created with hardened image meets all the OS hardening requirements and passes all 138 security compliance auditing tests.

7 Conclusions and Future Work

In this thesis, we have implemented a proof-of-concept solution for building a hardened Debian Linux virtual machine image for Azure cloud that is compliant with the holding company security compliance requirements. To support the work in thesis, we have done background study for understanding the importance of security compliance and operating system hardening, we have studied several security standards and hardening guidelines such as PCI DSS, HIPAA, ISO (27000 series), NIST (SP800 series) and hardening guidelines provided by CIS, NSA, NIST and DISA. Before we could start implementing the solution, we have analysed the holding company Linux hardening standard that is based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015). As the holding company hardening standard is based on DISA STIG for Red Hat, but the virtual machines that the parent company is managing are running on Linux Debian operating system, therefore the author decided to analyse the holding company standard and identify the relevant OS hardening requirements for Debian Linux systems. Based on the results, the holding company standard has too many not relevant hardening requirements for Debian systems, therefore new standard has been established for the parent company that is based on DISA STIG for Red Hat. This new standard is used as a baseline for implementing the proof-of-concept solution in this thesis.

In this paper, we have analysed three open source tools which can be used for hardened virtual machine image creation in Microsoft Azure cloud platform – Azure CLI, Azure Manage and Packer. As per our analysis, it was decided that Packer is the most suitable image creation tool as it met all the requirements of the parent company. Before building the hardened image, we have prepared a Chef configuration management tool *cookbook* which contains all the code for automatically implementing the hardening requirements from the new standard. Later, we integrated it with Packer and with Microsoft Azure platform and new hardened image has been built. Finally, we have compared our work with related work in security compliance implementing field.

Even though the new hardening standard was established and implemented, then there was still a gap how to prove to the auditors that the parent company meets the new hardening standard and is compliant from security perspective. It is not enough to just meet the holding company security compliance requirements, therefore we need to ensure auditability of it. To find solution for this gap, we have reviewed related work and tools in this field. We have studied several open source tools that can be used for automated security compliance auditing such as OpenScap, testing frameworks ServerSpec, InSpec, Rspec and audit mode in Chef Client. InSpec - security compliance auditing tool and testing framework that allows to follow DevOps approach "Compliance as Code" was chosen as it met all the requirements of the parent company. We have developed security compliance auditing tests for InSpec for automatically auditing the new OS hardening standard. Finally, we have compared our work with related work in security compliance auditing field.

To validate the proof-of-concept solution, we have created two virtual machines to Azure – one with default (non-hardened) VM image and second with hardened VM image that we have built in this thesis. For validating, we have used InSpec automated security compliance auditing tool and InSpec controls (tests) that we have developed in this work. According to InSpec scan results, we discovered that 29 hardening requirements (~21%) of 138 are not met on VM that was created with non-hardened image.

In this chapter, we are discussing the limitations of this work, we are answering to research questions and presenting the conclusions. Last but not least, the recommendations for future work will be presented.

7.1 Limitations

The work done in this thesis has several limitations:

1. The relevant hardening requirements that we have chosen for the new OS hardening standard is based on our understanding, experience and requirements of the parent company.
2. The implemented hardening requirements are based only on one hardening guideline – DISA STIG for Red Hat. Validation results of other security guidelines implementation might be different.
3. Due to limited time and resources, for implementing the proof-of-concept solution, we have taken only the hardening requirements in to scope that were chosen in the section 4.4. Therefore, we don't have visibility whether the parent company is compliant with the rest of hardening requirements.
4. The hardened VM image that we have built in this thesis is compatible only with Microsoft Azure cloud. We are not introducing how to make Packer template compatible with other cloud providers.
5. In this thesis, Packer build was triggered manually by human for building the hardened image, we have not introduced any continuous automated build process. We have implemented only proof-of-concept solution for building the hardened VM image.
6. We discovered that hardened VM image building using Packer is sometimes slow (~10min). We had no resources to investigate the root cause of this in this thesis.
7. Size of the hardened VM image that we have created in this thesis is 30GiB, the bigger the image is, the more money should be paid for storage in Azure cloud. Due to limited time, we did not investigate how to reduce this size.
8. In this thesis, InSpec profile for auditing the new hardening standard was triggered manually by human. We have not introduced any solution for triggering it automatically for continuous auditing.
9. For validating the proof-of-concept solution, we have used only two virtual machines – one with default VM image and second with hardened VM image. Results could be different if validating whole server park.
10. We have built proof-of-concept solution in this thesis, therefore it needs active testing before moving to production environment. Proof-of-concept solution might have unnoticed issues.

7.2 Answer to the Research Questions

RQ1: What are the security standards and hardening guidelines for security compliance in industry?

To understand security compliance and support our work in thesis, we have discovered that there are several security standards and hardening guidelines available such as PCI DSS, HIPAA, ISO (27000 series), NIST (SP800 series) and hardening guidelines provided by CIS, NSA, NIST and DISA.

RQ2: How to establish Linux operating system hardening standard for security compliance?

To establish the new Linux (Debian) operating system hardening standard for the parent company, we have analysed the holding company hardening standard (Appendix 1) that is based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015). After the

analysis, we have identified all the relevant hardening requirements from the holding company standard for the parent company Linux Debian operating systems. We discovered that 37 (21%) requirements of 177 were not relevant, 56 (32%) requirements were in place, 73 (41%) requirements were not in place, but could be added easily. For the rest 11 (6%) requirements the solution was not straightforward. These results allowed the author to establish the new hardening standard for the parent company that is used as a baseline for implementing the proof-of-concept solution in this thesis.

RQ3: How to implement Linux operating system hardening standard for security compliance?

To implement the hardening requirements from the new hardening standard for the parent company, we have decided to prepare a hardened virtual machine image for Microsoft Azure platform that has all these hardening requirements implemented. To not reinvent the wheel, we have studied related work in this field and analysed three open source tools out there that can be used for hardened VM image creation – Azure CLI, Azure Manage and Packer. As per our analysis, it was decided that Packer is the most suitable image creation tool as it met all the requirements of the parent company. Due to limited time and resources, we have not taken all hardening requirements in to scope for implementing in our proof-of-concept solution. Before building the hardened image, we have prepared configuration management tool Chef *cookbook* named *hardened-infra* (Appendix 2) that contains all the code for automatically implementing the hardening requirements from the new standard. We have used Test-Driven Development (Erdogmus, Morisio, & Torchiano, 2005) software development methodology while developing the proof-of-concept solution. For that, we have prepared development and testing environment using Test Kitchen (Nichol, 2017) which allowed us to integrate Microsoft Azure platform and configuration management tool Chef during Chef *cookbook* development. Finally, we have prepared Packer template where we integrated Packer with Microsoft Azure platform and Chef. As a result, we have built the new hardened image which is based on the new hardening standard.

RQ4: How to audit Linux operating system hardening standard for security compliance?

To audit the new operating system hardening standard, we have studied related work in this field and discovered several open source tools that can be used for automated security compliance auditing – OpenScap, testing frameworks ServerSpec, InSpec, Rspec and audit mode in Chef Client. InSpec (Chef Software Inc, 2017a) – security compliance auditing tool and testing framework that allows to follow DevOps approach "Compliance as Code" was chosen as it met all the requirements of the parent company. We have developed security compliance auditing tests for InSpec for automatically auditing the new OS hardening standard (Appendix 3).

RQ5: How many OS hardening requirements each chosen virtual machine (VM) meets from the new OS hardening standard?

We have answered to this research question to validate the proof-of-concept solution that we have implemented in this thesis. According to InSpec scan results, we discovered that 29 hardening requirements (~21%) of 138 were not met on virtual machine that was created with default (non-hardened) VM image. The hardened VM image that we have built in this thesis was compliant with all the hardening requirements from the new standard. Due to limited resources and time, these results were based only on comparing two virtual machines. Moreover, we have not taken all hardening requirements in to scope for validating the proof-of-concept solution.

7.3 Conclusions

In this thesis, we have found answers to all research questions that we stated in the beginning of this work, this allowed us to achieve all our goals. We have established a new operating system hardening standard for the parent company Linux Debian operating systems that is based on DISA STIG for Red Hat (Defense Information Systems Agency, 2015). This new standard was used as a baseline for implementing the proof-of-concept solution – a hardened Linux Debian virtual machine image for Microsoft Azure platform that is based on the new hardening standard. The hardened virtual machine image was built with help of open source image creation tool Packer and configuration management tool Chef. Even though the new hardening standard was established and implemented, then there was still a gap how to prove to the auditors that the parent company meets the new hardening standard and is compliant with security requirements. For solving this gap, we have implemented a proof-of-concept solution using InSpec for automatically auditing the new hardening standard. To validate the proof-of-concept solution, we have analysed virtual machines compliance state before and after implementing the new operating system hardening standard.

We have successfully built a proof-of-concept solution for the parent company that can be used for being compliant with the holding company security compliance requirements. By using this solution, we can ensure that the parent company virtual machines have identical hardened configurations implemented and that they are compliant with the new hardening standard. This is step forward that the parent company will pass internal audit of the holding company.

We have learned (while studying several data breach reports and security standards in this thesis) that security compliances helps for achieving more security, therefore the hardened virtual machine image solution that we have implemented in this thesis is useful for mitigating and minimizing the risks of several security threats. For example, after we have validated the proof-of-concept solution, we have confirmed that standard Microsoft Azure Linux Debian images are not built for security by default nor they are hardened against any security guidelines, therefore the solution that we have built in this thesis can be used as alternative virtual machine image for achieving more security.

The automated continuous security compliance auditing solution that we have introduced in this thesis ensures that all virtual machines are hardened according to the new standard – any misconfiguration will be detected and fixed as soon as possible. As the parent company does hundreds of production changes every month then the automated security compliance auditing makes sure that these changes are also compliant with the new hardening standard. Moreover, this solution is cost effective for the company as it requires no human resources for collecting data for the auditors. It will also boost on-site auditing process – if evidence providing to the auditors of specific operating system configuration was provided manually before, then now it is just matter of triggering the automation (InSpec profile) and the auditors will get the results automatically.

7.4 Future Work

For future work, it is possible to address all the limitations that we have identified in this thesis. In addition, it is possible to take different security standard or operating system hardening guideline as a baseline and use similar solution that we have built in this thesis. Moreover, it would be interesting to see same solution for other cloud providers such as AWS, OpenStack and Google Cloud. Last but not least, it would be interesting to see similar solution that is implemented with using other configuration management tools such as Puppet, Ansible and SaltStack.

8 References

Andress, J. (2014). *The Basics of Information Security (Second Edition).* Waltham: Syngress.

Arbuckle, J. (2014). Automating Away the Regulatory Compliance Myth. (G. V. Hulme, Interviewer) Retrieved May 11, 2017, from https://devops.com/automating-away-regulatory-compliance-myth/

Bauer, M. D. (2003). *Building secure servers with Linux.* Sebastopol, California: O'Reilly & Associates, Inc.

Bednarski, C. (2017). *Packer*. Retrieved May 11, 2017, from github.com: https://github.com/mitchellh/packer/blob/master/contrib/azure-setup.sh

Bird, J. (2016). *DevOpsSec.* Sebastopol, California: O'Reilly Media, Inc.

Blank, B. (2017). *Azure management for Debian*. Retrieved May 11, 2017, from github.com: https://github.com/credativ/azure-manage/

Bleikertz, S. (2010). Automated Security Analysis of Infrastructure Clouds. Lyngby, Denmark: Technical University of Denmark.

Center for Internet Security. (2017a). *CIS Benchmarks*. Retrieved May 11, 2017, from cisecurity.org: https://learn.cisecurity.org/benchmarks

Center For Internet Security. (2017b). *CIS Hardened Virtual Images*. Retrieved May 11, 2017, from cisecurity.org: https://www.cisecurity.org/services/hardened-virtual-images/

Chef Software Inc. (2017a). *InSpec - Audit and Test Framework*. Retrieved May 11, 2017, from inspec.io: http://inspec.io/

Chef Software Inc. (2017b). *inspec/example.rb*. Retrieved May 11, 2017, from github.com: https://github.com/chef/inspec/blob/master/examples/profile/controls/example.rb

Ciske van Oosten, A. B. (2015). *PCI Compliance report*. Retrieved May 11, 2017, from http://www.verizonenterprise.com/placeholder/resources/reports/rp_pci-report-2015_en_xg.pdf

Defense Information Systems Agency. (2015). *Red Hat Enterprise Linux 6 Security Technical Implementation Guide.* Retrieved May 11, 2017, from iasecontent.disa.mil: http://iasecontent.disa.mil/stigs/zip/Apr2015/U_RedHat_6_V1R7_STIG.zip

Defense Information Systems Agency. (2017). *STIG Viewing Guidance*. Retrieved May 11, 2017, from http://iase.disa.mil/: http://iase.disa.mil/stigs/Pages/stig-viewing-guidance.aspx

Doelitzscher, F. (2014). Security Audit Compliance for Cloud Computing. Furtwangen, Germany: Plymouth University.

Erdogmus, H., Morisio, M., & Torchiano, M. (2005). On the Effectiveness of the Test-First Approach to Programming. *IEEE Transactions on Software Engineering*, 226 - 237.

Foulds, I., Squillace, R., Lepow, D., Wong, H., Mansfield, C., & Kshirsagar, D. (2017). *How to generalize and capture a Linux virtual machine using the Azure CLI 2.0*. Retrieved May 11, 2017, from Microsoft Docs: https://docs.microsoft.com/en-us/azure/virtual-machines/virtual-machines-linux-capture-image

Gumprich, S. (2017). *os-hardening (Ansible Role)*. Retrieved May 11, 2017, from github.com: https://github.com/dev-sec/ansible-os-hardening

Hartmann, C. (2017). *Test-Kitchen Plugin for InSpec*. Retrieved May 11, 2017, from github.com: https://github.com/chef/kitchen-inspec

HashiCorp. (2017a). *Introduction to Packer*. Retrieved May 11, 2017, from Packer.io: https://www.packer.io/intro/
HashiCorp. (2017b). *Install Packer*. Retrieved May 11, 2017, from Packer.io: https://www.packer.io/intro/getting-started/install.html
HashiCorp. (2017c). *Authorizing Packer builds in Azure*. Retrieved May 11, 2017, from Packer.io: https://www.packer.io/docs/builders/azure-setup.html
HashiCorp. (2017d). *Build an Image*. Retrieved May 11, 2017, from Packer.io: https://www.packer.io/intro/getting-started/build-image.html
HashiCorp. (2017e). *Azure Resource Manager Builder*. Retrieved May 11, 2017, from Packer.io: https://www.packer.io/docs/builders/azure-arm.html
HashiCorp. (2017f). *Chef Solo Provisioner*. Retrieved May 11, 2017, from Packer.io: https://www.packer.io/docs/provisioners/chef-solo.html
HashiCorp. (2017g). *Shell Provisioner*. Retrieved May 11, 2017, from Packer.io: https://www.packer.io/docs/provisioners/shell.html
Health Information Privacy. (2017). Retrieved May 11, 2017, from hhs.gov: https://www.hhs.gov/hipaa/for-professionals/security/index.html
Identity Theft Resource Center. (2017). *Data Breach reports - 2016 End of Year Rport.* Retrieved May 11, 2017, from http://www.idtheftcenter.org/images/breach/2016/DataBreachReport_2016.pdf
International Organization for Standardization. (2017). *ISO/IEC 27001 Information security management*. Retrieved May 11, 2017, from ISO.org: https://www.iso.org/isoiec-27001-information-security.html
Jahoda, M., Krátký, R., Prpič, M., Capek, T., Wadeley, S., Ruseva, Y., & Svoboda, M. (2016). *A Guide to Securing Red Hat Enterprise Linux.* Retrieved May 11, 2017, from redhat.com: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/pdf/Security_Guide/Red_Hat_Enterprise_Linux-6-Security_Guide-en-US.pdf
Julisch, K. (2009). Security Compliance: The Next Frontier in Security Research. *Proceedings of the 2008 workshop on New security paradigms*, p71-74. Rüschlikon, Switzerland.
Koschorreck, G. (2011). Automated Audit of Compliance and Security Controls. *2011 Sixth International Conference on IT Security Incident Management and IT Forensics,* 137 - 148.
Kshirsagar, D., Kmouss, K., Lepow, D., Squillace, R., Wheeler, S., & Newill, T. (2017). *Install the Azure CLI*. Retrieved May 11, 2017, from Microsoft Docs: https://docs.microsoft.com/en-us/azure/cli-install-nodejs
Lustig, M. (2015). *Compliance at Speed.* Sebastopol, California: O'Reilly Media, Inc.
Montesino, R., & Fenz, S. (2011a). Information Security Automation: How Far Can We Go? *2011 Sixth International Conference on Availability, Reliability and Security*, 280 - 285.
Montesino, R., & Fenz, S. (2011b). Automation Possibilities in Information Security Management. *2011 European Intelligence and Security Informatics Conference,* 259 - 262.
National Security Agency. (2007). *Redhat Enterprise Linux 5 Hardening Guide.* Retrieved May 11, 2017, from NSA.gov: http://www.nsa.gov/ia/_files/os/redhat/rhel5-guide-i731.pdf
Nichol, F. (2017). *Test Kitchen*. Retrieved May 11, 2017, from github.com: https://github.com/test-kitchen/test-kitchen

Nottingham, C., Peterson, N., Foulds, I., & Squillace, R. (2016). *About images for Linux virtual machines*. Retrieved May 11, 2017, from Microsoft Docs: https://docs.microsoft.com/en-us/azure/virtual-machines/virtual-machines-linux-classic-about-images

PCI Security Standards Council LLC. (2016). *Data Security Standard.* Retrieved May 11, 2017, from Official PCI Security Standards Council Site: https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2.pdf

PCI Security Standards Council LLC. (2017). *About us*. Retrieved May 11, 2017, from Official PCI Security Standards Council Site: https://www.pcisecuritystandards.org/about_us/

Peña, J. F.-S., & Reelsen, A. (2012). *Securing Debian Manual.* Retrieved May 11, 2017, from debian.org: https://www.debian.org/doc/manuals/securing-debian-howto/

Preston, S. (2017). *kitchen-azurerm*. Retrieved May 11, 2017, from github.com: https://github.com/test-kitchen/kitchen-azurerm

PwC. (2015). *2015 Information Security Breaches Survey.* PwC. Retrieved May 11, 2017, from http://www.pwc.co.uk/assets/pdf/2015-isbs-technical-report-blue-digital.pdf

Ranum, M. J. (1992). A Network Firewall. Greenbelt.

Richter, D. (2017a). *os-hardening (Chef cookbook)*. Retrieved May 11, 2017, from github.com: https://github.com/dev-sec/chef-os-hardening

Richter, D. (2017b). *os_hardening (Puppet module)*. Retrieved May 11, 2017, from forge.puppet.com: https://forge.puppet.com/hardening/os_hardening

Zarkos, S. A., Peterson, N., Squillace, R., Nottingham, C., Lepow, D., & Pasic, A. (2017). *Prepare a Debian VHD for Azure*. Retrieved May 11, 2017, from Microsoft Docs: https://docs.microsoft.com/en-us/azure/virtual-machines/linux/debian-create-upload-vhd

The National Institute of Standards and Technology. (2017a). *NIST Special Publications*. Retrieved May 11, 2017, from csrc.nist.gov: http://csrc.nist.gov/publications/PubsSPs.html#SP%20800

The National Institute of Standards and Technology. (2017b). *NIST Computer Security Publications*. Retrieved May 11, 2017, from csrc.nist.gov: http://csrc.nist.gov/publications/PubsDrafts.html

The National Institute of Standards and Technology. (2017c). *Security Content Automation Protocol Validated Products*. Retrieved May 11, 2017, from nvd.nist.gov: https://nvd.nist.gov/scap/validated-tools

Turnbull, J. (2005). *Hardening Linux.* Berkeley: Apress.

Ullah, K. W., Ahmed, A. S., & Ylitalo, J. (2013). Towards Building an Automated Security Compliance Tool for the Cloud. *IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 1587 - 1593.

Verizon. (2016). *Verizon's 2016 Data Breach Investigations Report.* Verizon. Retrieved May 11, 2017, from http://www.verizonenterprise.com/resources/reports/rp_DBIR_2016_Report_en_xg.pdf

Appendix

I. The Holding Company Linux Hardening Standard

The holding company requires that the hardening requirements from DISA STIG for Red Hat are applied (Defense Information Systems Agency, 2015).

The hardening requirements titles (1-204) below are taken from (Defense Information Systems Agency, 2015) *that is published as "U_RedHat_6_V1R7_Manual-xccdf.xml" file.*

The full description of the hardening requirements and their auditing guidelines are available from (Defense Information Systems Agency, 2015) *in "U_RedHat_6_V1R7_Manual-xccdf.xml" file. We have described DISA STIG for Red Hat in the section 2.4.*

In the reality, the holding company standard contains several other hardening requirements, but in thesis we are taking only publicly available hardening requirements based on DISA STIG for Red Hat in to scope.

1. A file integrity tool must be installed.
2. A login banner must be displayed immediately prior to, or as part of, graphical desktop environment login prompts.
3. Accounts must be locked upon 35 days of inactivity.
4. All public directories must be owned by a system account.
5. All rsyslog-generated log files must be owned by root.
6. All system command files must be owned by root.
7. All system command files must have mode 0755 or less permissive.
8. Audit log files must be owned by root.
9. Audit log files must have mode 0640 or less permissive.
10. Auditing must be enabled at boot by setting a kernel parameter.
11. Automated file system mounting tools must not be enabled unless needed.
22. IP forwarding for IPv4 must not be enabled, unless the system is a router.
23. Library files must be owned by root.
25. Mail relaying must be restricted.
28. Process core dumps must be disabled unless needed.
37. The /etc/group file must be group-owned by root.
38. The /etc/group file must be owned by root.
39. The /etc/group file must have mode 0644 or less permissive.
40. The /etc/gshadow file must be group-owned by root.
41. The /etc/gshadow file must be owned by root.
42. The /etc/gshadow file must have mode 0000.
43. The /etc/passwd file must be group-owned by root.
44. The /etc/passwd file must be owned by root.

45. The /etc/passwd file must have mode 0644 or less permissive.

46. The /etc/passwd file must not contain password hashes.

47. The /etc/shadow file must be group-owned by root.

48. The /etc/shadow file must be owned by root.

49. The /etc/shadow file must have mode 0000.

50. The atd service must be disabled.

51. The audit system must alert designated staff members when the audit storage volume approaches capacity.

52. The audit system must be configured to audit all attempts to alter system time through /etc/localtime.

53. The audit system must be configured to audit all attempts to alter system time through adjtimex.

54. The audit system must be configured to audit all attempts to alter system time through clock_settime.

55. The audit system must be configured to audit all attempts to alter system time through settimeofday.

56. The audit system must be configured to audit all attempts to alter system time through stime.

57. The audit system must be configured to audit all discretionary access control permission modifications using chmod.

58. The audit system must be configured to audit all discretionary access control permission modifications using chown.

59. The audit system must be configured to audit all discretionary access control permission modifications using fchmod.

60. The audit system must be configured to audit all discretionary access control permission modifications using fchmodat.

61. The audit system must be configured to audit all discretionary access control permission modifications using fchown.

62. The audit system must be configured to audit all discretionary access control permission modifications using fchownat.

63. The audit system must be configured to audit all discretionary access control permission modifications using fremovexattr.

64. The audit system must be configured to audit all discretionary access control permission modifications using fsetxattr.

65. The audit system must be configured to audit all discretionary access control permission modifications using lchown.

66. The audit system must be configured to audit all discretionary access control permission modifications using lremovexattr.

67. The audit system must be configured to audit all discretionary access control permission modifications using lsetxattr.

68. The audit system must be configured to audit all discretionary access control permission modifications using removexattr.

69. The audit system must be configured to audit all discretionary access control permission modifications using setxattr.

70. The audit system must be configured to audit changes to the /etc/sudoers file.

71. The audit system must be configured to audit modifications to the systems Mandatory Access Control (MAC) configuration (SELinux).

72. The audit system must be configured to audit successful file system mounts.

73. The audit system must be configured to audit the loading and unloading of dynamic kernel modules.

74. The audit system must be configured to audit user deletions of files and programs.

75. The audit system must identify staff members to receive notifications of audit log storage volume capacity issues.

76. The Automatic Bug Reporting Tool (abrtd) service must not be running.

77. The avahi service must be disabled.

78. The Bluetooth service must be disabled.

79. The cron service must be running.

80. The Datagram Congestion Control Protocol (DCCP) must be disabled unless required.

81. The DHCP client must be disabled if not needed.

82. The graphical desktop environment must automatically lock after 15 minutes of inactivity and the system must require user reauthentication to unlock the environment.

83. The graphical desktop environment must have automatic lock enabled.

84. The graphical desktop environment must set the idle timeout to no more than 15 minutes.

85. The IPv6 protocol handler must not be bound to the network stack unless needed.

86. The netconsole service must be disabled unless required.

87. The ntpdate service must not be running.

88. The oddjobd service must not be running.

89. The openldap-servers package must not be installed unless required.

90. The operating system must automatically audit account creation.

91. The operating system must automatically audit account disabling actions.

92. The operating system must automatically audit account modification.

93. The operating system must automatically audit account termination.

94. The operating system must back up audit records on an organization defined frequency onto a different system or media than the system being audited.

95. The operating system must enforce requirements for the connection of mobile devices to operating systems.

96. The operating system must manage information system identifiers for users and devices by disabling the user identifier after an organization defined time period of inactivity.

97. The operating system must prevent public IPv4 access into an organizations internal networks, except as appropriately mediated by managed interfaces employing boundary protection devices.

98. The operating system must support the requirement to centrally manage the content of audit records generated by organization defined information system components.

99. The operating system, upon successful logon/access, must display to the user the number of unsuccessful logon/access attempts since the last successful logon/access.

100. The postfix service must be enabled for mail delivery.

101. The qpidd service must not be running.

102. The rdisc service must not be running.

103. The Reliable Datagram Sockets (RDS) protocol must be disabled unless required.

104. The rexecd service must not be running.

105. The rlogind service must not be running.

106. The root account must be the only account having a UID of 0.

107. The rshd service must not be running.

108. The rsh-server package must not be installed.

109. The sendmail package must be removed.

110. The SSH daemon must be configured to use only the SSHv2 protocol.

111. The SSH daemon must be configured with the Department of Defense (DoD) login banner.

112. The SSH daemon must ignore .rhosts files.

113. The SSH daemon must not allow authentication using an empty password.

114. The SSH daemon must not allow host-based authentication.

115. The SSH daemon must not permit user environment settings.

116. The SSH daemon must set a timeout count on idle sessions.

117. The SSH daemon must set a timeout interval on idle sessions.

118. The sticky bit must be set on all public directories.

119. The Stream Control Transmission Protocol (SCTP) must be disabled unless required.

120. The system boot loader configuration file(s) must be group-owned by root.

121. The system boot loader configuration file(s) must be owned by root.

122. The system boot loader configuration file(s) must have mode 0600 or less permissive.

123. The system boot loader must require authentication.

124. The system clock must be synchronized continuously, or at least daily.

125. The system clock must be synchronized to an authoritative DoD time source.

126. The system default umask for daemons must be 027 or 022.

127. The system default umask for the bash shell must be 077.

128. The system default umask for the csh shell must be 077.

129. The system default umask in /etc/login.defs must be 077.

130. The system default umask in /etc/profile must be 077.

131. The system must allow locking of the console screen in text mode.

132. The system must be configured to use TCP syncookies.

133. The system must disable accounts after excessive login failures within a 15-minute interval.

134. The system must disable accounts after three consecutive unsuccessful logon attempts.

135. The system must display a publicly-viewable pattern during a graphical desktop environment session lock.

136. The system must employ a local IPv4 firewall.

137. The system must ignore ICMPv4 bogus error responses.

138. The system must ignore ICMPv4 redirect messages by default.

139. The system must ignore ICMPv6 redirects by default.

140. The system must limit users to 10 simultaneous system logins, or a site-defined number, in accordance with operational requirements.

141. The system must log Martian packets.

142. The system must not accept ICMPv4 redirect packets on any interface.

143. The system must not accept ICMPv4 secure redirect packets by default.

144. The system must not accept ICMPv4 secure redirect packets on any interface.

145. The system must not accept IPv4 source-routed packets by default.

146. The system must not accept IPv4 source-routed packets on any interface.

147. The system must not have accounts configured with blank or null passwords.

148. The system must not permit interactive boot.

149. The system must not permit root logins using remote access programs such as ssh.

150. The system must not respond to ICMPv4 sent to a broadcast address.

151. The system must not send ICMPv4 redirects by default.

152. The system must not send ICMPv4 redirects from any interface.

153. The system must prevent the root account from logging in from serial consoles.

154. The system must prevent the root account from logging in from virtual consoles.

155. The system must provide VPN connectivity for communications over untrusted networks.

156. The system must require administrator action to unlock an account locked by excessive failed login attempts.

157. The system must require at least four characters be changed between the old and new passwords during a password change.

158. The system must require authentication upon booting into single-user and maintenance modes.

159. The system must require passwords to contain a minimum of 14 characters.

160. The system must require passwords to contain at least one lowercase alphabetic character.

161. The system must require passwords to contain at least one numeric character.

162. The system must require passwords to contain at least one special character.

163. The system must require passwords to contain at least one uppercase alphabetic character.

164. The system must retain enough rotated audit logs to cover the required log retention period.

165. The system must rotate audit log files that reach the maximum file size.

166. The system must set a maximum audit log file size.

167. The system must use a FIPS 140-2 approved cryptographic hashing algorithm for generating account password hashes (libuser.conf).

168. The system must use a FIPS 140-2 approved cryptographic hashing algorithm for generating account password hashes (login.defs).

169. The system must use a FIPS 140-2 approved cryptographic hashing algorithm for generating account password hashes (system-auth).

170. The system must use a Linux Security Module at boot time.

171. The system must use a Linux Security Module configured to enforce limits on system services.

172. The system must use a Linux Security Module configured to limit the privileges of system services.

173. The system must use a reverse-path filter for IPv4 network traffic when possible by default.

174. The system must use a reverse-path filter for IPv4 network traffic when possible on all interfaces.

175. The system must use a separate file system for /tmp.

176. The system must use a separate file system for /var.

177. The system must use a separate file system for /var/log.

178. The system must use a separate file system for the system audit data path.

179. The system must use a separate file system for user home directories.

180. The system must use SMB client signing for connecting to samba servers using smbclient.

181. The system package management tool must cryptographically verify the authenticity of all software packages during installation.

182. The system package management tool must cryptographically verify the authenticity of system software packages during installation.

183. The systems local IPv4 firewall must implement a deny-all, allow-by-exception policy for inbound packets.

184. The telnet daemon must not be running.

185. The telnet-server package must not be installed.

186. The TFTP daemon must operate in secure mode which provides access only to a single directory on the host file system.

187. The tftp-server package must not be installed.

188. The Transparent Inter-Process Communication (TIPC) protocol must be disabled unless required.

189. The xinetd service must be disabled if no network services utilizing it are enabled.

190. The xinetd service must be uninstalled if no network services utilizing it are enabled.

191. The xorg-x11-server-common (X Windows) package must not be installed, unless required.

192. The ypbind service must not be running.

193. The ypserv package must not be installed.

194. There must be no .rhosts or hosts.equiv files on the system.

197. User passwords must be changed at least every 60 days.

198. Users must be warned 7 days in advance of password expiration.

199. Users must not be able to change passwords more than once every 24 hours.

204. X Windows must not be enabled unless required.

(Defense Information Systems Agency, 2015)

II. Chef Cookbook for Implementing the New Hardening Standard

There are 6 recipes for each hardening requirement group in the section 4.4. The Chef cookbook *hardened-infra* project file tree overview:

```
hardened-infra
├── attributes
├── CHANGELOG.md
├── files
│   └── default
│       └── banner
├── .kitchen
│   ├── default-debian-7.yml
│   └── logs
│       ├── default-debian-7.log
│       ├── default-debian-8.log
│       ├── default-ubuntu-1404.log
│       └── kitchen.log
├── .kitchen.yml
├── metadata.rb
├── README.md
├── recipes
│   ├── 1_filepermissions.rb
│   ├── 3_sshd.rb
│   ├── 4_sysctls.rb
│   ├── 5_packages.rb
│   ├── 6_services.rb
│   ├── 7_protocols.rb
│   └── default.rb
├── .rubocop.yml
├── templates
│   └── default
│       ├── common-session.erb
│       ├── login.defs.erb
│       ├── protocols.conf.erb
│       ├── sshd_config.erb
│       └── sysctl.conf.erb
└── test
    └── integration
        └── default
            ├── controls
            │   ├── 1_filepermissions.rb
            │   ├── 3_sshd.rb
            │   ├── 4_sysctls.rb
            │   ├── 5_packages.rb
            │   ├── 6_services.rb
            │   ├── 7_protocols.rb
            │   └── inspec.lock
            ├── inspec.yml
            ├── .kitchen
            │   └── logs
            │       └── kitchen.log
            ├── libraries
            │   └── .gitkeep
            └── README.md

15 directories, 34 files
```

While developing and implementing the hardening requirements, we have used DISA STIG for Red Hat (Defense Information Systems Agency, 2015) *guidelines and recommendatios where possible.*

Chef recipes

Default.rb:

```
# Make sure that image has latest DSA packages
execute 'aptupgrade' do
  command 'apt-get update; apt-get upgrade --force-yes -y'
  action :run
end

# Harden our infrastructure
include_recipe 'hardened-infra::1_filepermissions'
include_recipe 'hardened-infra::3_sshd'
include_recipe 'hardened-infra::4_sysctls'
include_recipe 'hardened-infra::5_packages'
include_recipe 'hardened-infra::6_services'
include_recipe 'hardened-infra::7_protocols'
```

1_filepermissions.rb:

```
# 1. Secure file and directory permissions and ownership

# All syslog-ng generated log files owned by root (req. 5)
# We don't use rsyslog, remove it
package 'rsyslog' do
  action :purge
end
# Install syslog-ng instead
package 'syslog-ng' do
  action :install
end

# NOTE: permissions should be fixed in syslog-ng.conf as well
['auth.log', 'cron.log', 'daemon.log', 'kern.log', 'lpr.log', 'mail.log',
 'syslog', 'user.log', 'uucp.log', 'messages'
].each do |logfile|
  next unless File.exist?("/var/log/#{logfile}")
  file "/var/log/#{logfile}" do
    owner 'root'
    mode 00640
  end
end

# Stricter mode for password, shadow and group files - owned by root, mode 0640
# or less (req. 37-49)
['/etc/group', '/etc/gshadow', '/etc/passwd', '/etc/shadow'].each do |file|
  file file.to_s do
    mode '0640'
    owner 'root'
    group 'root'
  end
end

# Bootloader configuration files owned by root, mode 0600 (req. 120-122)
['/boot/extlinux/extlinux.conf', '/boot/extlinux/linux.cfg',
 '/boot/extlinux/memdisk.cfg', '/boot/grub/menu.lst', '/etc/grub.conf',
 '/boot/grub/grub.conf'
].each do |file|
  next unless File.exist?(file.to_s)
  file file.to_s do
    mode '0600'
    owner 'root'
    group 'root'
  end
end
```

```
# Secure umask is set by default for both users and daemons (req. 126-127,129)
# Debian ship with pam_umask. This allows to configure umask in /etc/login.defs
# and have it applied on whole system.
template "/etc/login.defs" do
  source "login.defs.erb"
end

template "/etc/pam.d/common-session" do
  source "common-session.erb"
end
```

3_sshd.rb:

```
# 3. Secure SSHD configuration

template '/etc/ssh/sshd_config' do
  source 'sshd_config.erb'
  notifies :restart, 'service[ssh]', :delayed
end

cookbook_file '/etc/banner' do
  source 'banner'
  owner 'root'
  group 'root'
  mode 00644
  notifies :restart, 'service[ssh]', :delayed
end

service 'ssh' do
  service_name 'ssh'
  supports [:restart, :reload, :status]
  action [:enable, :start]
end
```

4_sysctls.rb

```
# 4. Secure kernel (sysctls) configuration

template "/etc/sysctl.conf" do
  source "sysctl.conf.erb"
  owner "root"
  group "root"
  mode 0644
  notifies :run, "execute[reload-sysctl]", :immediately
end

execute "reload-sysctl" do
  command "sysctl -p"
  action :nothing
end
```

7_protocols.rb

```
# 7. Security related to protocols

# Remove not needed kernel module
blacklist = %w(dccp rds sctp tipc ipv6 netconsole)
blacklist.each do |kmodule|
  execute "Remove kernel module #{kmodule}" do
    command "modprobe -r #{kmodule}"
    action :run
    only_if "grep -q '^#{kmodule} ' /proc/modules"
  end
end

# Stop loading kernel module during startup
template '/etc/modprobe.d/protocols.conf' do
```

```
  source 'protocols.conf.erb'
  owner 'root'
  group 'root'
  mode 00744
  variables(
    dmod: blacklist
  )
end
```

Chef templates

We have taken default configuration files as a baseline that are available in default VM image for Debian in Azure. While developing Chef cookbook, we have used DISA STIG for Red Hat (Defense Information Systems Agency, 2015) *guidelines where possible (for SSHD, Sysctl and umask settings).*

sshd_config.erb:

```
# Generated by Chef

Port 22
# Display login banner (req. 2, 111)
Banner /etc/banner
# Only protocol version 2 is allowed (req. 110)
Protocol 2
# Ignore .rhosts files (req. 112)
IgnoreRhosts yes
# Do not allow host-based authentication and authentication using an empty pass-
word (req. 113-114)
HostbasedAuthentication no
PermitEmptyPasswords no
# Do not permit user environment settings (req. 115)
PermitUserEnvironment no
# Idle timeout is enforced, set for 15 minutes (req. 116-117)
ClientAliveCountMax 0
ClientAliveInterval 900
# Root login is disabled (req. 149)
PermitRootLogin no
# Password based authentication is disabled (extra req.)
PasswordAuthentication no
# Other SSHD configuration parameters that are not in scope of this thesis (pa-
rameters and its vales below are default ones of SSHD that is shipped with
Debian VM image in Azure)
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
UsePrivilegeSeparation yes
KeyRegenerationInterval 3600
ServerKeyBits 768
SyslogFacility AUTH
LogLevel INFO
LoginGraceTime 120
StrictModes yes
RSAAuthentication yes
PubkeyAuthentication yes
RhostsRSAAuthentication no
X11Forwarding no
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
AcceptEnv LANG LC_*
Subsystem sftp /usr/lib/openssh/sftp-server
UsePAM yes
ChallengeResponseAuthentication no
```

sysctl.conf.erb:

Recommended Sysctl parameters and its values are taken from (Defense Information Systems Agency, 2015).

```
# Generated by Chef
net.ipv4.ip_forward = 0
net.ipv4.tcp_syncookies = 1
net.ipv4.icmp_ignore_bogus_error_responses = 1
net.ipv4.conf.default.accept_redirects = 0
net.ipv6.conf.default.accept_redirects = 0
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.icmp_echo_ignore_broadcasts = 1
net.ipv4.conf.default.send_redirects = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.all.rp_filter = 1
# disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

protocols.conf.erb:

```
# Generated by Chef
<% @dmod.each do |h| -%>
install <%= h %> /bin/false
<% end -%>
options ipv6 disable=1
```

login.defs.erb:

```
# Generated by Chef
MAIL_DIR        /var/mail
FAILLOG_ENAB            yes
LOG_UNKFAIL_ENAB        no
LOG_OK_LOGINS           no
SYSLOG_SU_ENAB          yes
SYSLOG_SG_ENAB          yes
FTMP_FILE       /var/log/btmp
SU_NAME         su
HUSHLOGIN_FILE  .hushlogin
ENV_SUPATH      PATH=/usr/local/sbin:/usr/lo-
cal/bin:/usr/sbin:/usr/bin:/sbin:/bin
ENV_PATH        PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
TTYGROUP        tty
TTYPERM         0600
ERASECHAR       0177
KILLCHAR        025
UMASK           077
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_WARN_AGE   7
UID_MIN                  1000
UID_MAX                 60000
GID_MIN                  1000
GID_MAX                 60000
LOGIN_RETRIES           5
LOGIN_TIMEOUT           60
CHFN_RESTRICT           rwh
DEFAULT_HOME    yes
USERGROUPS_ENAB yes
ENCRYPT_METHOD SHA512
```

common-session.rb:

```
# Generated by Chef
session [default=1]                     pam_permit.so
session requisite                       pam_deny.so
session required                        pam_permit.so
session required                        pam_unix.so
session optional                        pam_winbind.so
session optional                        pam_ck_connector.so nox11
session optional                        pam_umask.so
```

The *hardened-infra/test/integration/default/controls* directory is used for integration tests that we use while developing the OS hardening requirements in to Chef *cookbook*. These InSpec tests are also used for security compliance auditing and we are describing these in Appendix 3.

III. InSpec Profile for Auditing the New Hardening Standard

We have created InSpec controls (tests) and integrated it with Test Kitchen. For Chef *cookbook* development, we have added InSpec controls (tests) to the *hardened-infra/test/integration/default/controls* directory as follows:

```
hardened-infra/test
└── integration
    └── default
        ├── controls
        │   ├── 1_filepermissions.rb
        │   ├── 3_sshd.rb
        │   ├── 4_sysctls.rb
        │   ├── 5_packages.rb
        │   ├── 6_services.rb
        │   ├── 7_protocols.rb
        │   └── inspec.lock
        ├── inspec.yml
        ├── .kitchen
        │   └── logs
        │       └── kitchen.log
        ├── libraries
        │   └── .gitkeep
        └── README.md
```

InSpec profile overview. This profile is used for executing the included controls (tests) on any remote server via SSH and get the results automatically:

```
inspec-hardened-infra
├── controls
│   ├── 1_filepermissions.rb
│   ├── 3_sshd.rb
│   ├── 4_sysctls.rb
│   ├── 5_packages.rb
│   ├── 6_services.rb
│   ├── 7_protocols.rb
│   └── example.rb
├── inspec.yml
├── libraries
│   └── .gitkeep
└── README.md

2 directories, 10 files
```

While developing security compliance auditing tests, we have used DISA STIG for Red Hat (Defense Information Systems Agency, 2015) *auditing guidelines where possible, such as Linux commands what to execute for verifying if specific security requirement has been implemented (For example, find commands that are using InSpec command resource below in 1_filepermission.rb).*

1_filepermissions.rb:

```
# 1. Secure file and directory permissions and ownership

control '1. Secure file and directory permissions and ownership' do
  impact 1.0
  title 'Only secure file and directory permissions and ownership should be
used'
  # Correct owner for public directories
  describe command('find / -xdev -type d -perm -0002 -uid +499 -print') do
    its('stdout') { should eq '' }
  end
```

```
  # All syslog-ng generated log files owned by root (req. 5)
  ['auth.log', 'cron.log', 'daemon.log', 'kern.log', 'lpr.log', 'mail.log',
   'syslog', 'user.log', 'uucp.log', 'messages'
  ].each do |logfile|
    next unless file("/var/log/#{logfile}").exist?
    describe file("/var/log/#{logfile}") do
      its('owner') { should eq 'root' }
    end
  end
  # System command files owned by root, have mode 0755 or less permissive (req.
6,7)
  ['/bin', '/usr/bin', '/usr/local/bin', '/sbin', '/usr/sbin', '/usr/local/sbin'
  ].each do |dir|
    describe command("find -L #{dir} \! -user root") do
      its('stdout') { should eq '' }
    end
    describe command("find -L #{dir} -perm /022 -type f") do
      its('stdout') { should eq '' }
    end
  end
  # Shared libraries owned by root (req. 23)
  ['/lib', '/lib64', '/usr/lib', '/usr/lib64'].each do |library|
    describe command("find -L #{library} \! -user root") do
      its('stdout') { should eq '' }
    end
  end
  # Stricter mode for password, shadow and group files - owned by root, mode
0640 or less (req. 37-49)
  ['/etc/group', '/etc/gshadow', '/etc/passwd', '/etc/shadow'].each do |file|
    describe file(file.to_s) do
      it { should be_owned_by 'root' }
      its('group') { should eq 'root' }
      its('mode') { should cmp '00640' }
    end
  end
  # The /etc/passwd file must not contain password hashes (req. 46)
  describe passwd.passwords(/^(?!x$)/) do
    its('entries.length') { should be == 0 }
  end
  # Sticky bits for public writable directories (req. 118)
  describe command('find / -xdev -type d -perm -002 ! -perm -1000') do
    its('stdout') { should eq '' }
  end
  # Bootloader configuration files owned by root, mode 0600 (req. 120-122)
  # We use extlinux in Azure, Grub is used on some places as well
  ['/boot/extlinux/extlinux.conf', '/boot/extlinux/linux.cfg',
   '/boot/extlinux/memdisk.cfg', '/boot/grub/menu.lst', '/etc/grub.conf',
   '/boot/grub/grub.conf'
  ].each do |file|
    next unless file(file.to_s).exist?
    describe file(file.to_s) do
      its('owner') { should eq 'root' }
      its('group') { should eq 'root' }
      its('mode') { should cmp '00600' }
    end
  end
  # Secure umask is set by default for users and daemons (req. 126-127,129)
  describe file('/etc/init.d/rc') do
    its('content') { should match /^umask 022|umask 027$/ }
  end
# Debian ship with pam_umask. This allows to configure umask in /etc/login.defs
# and have it applied everywhere on the system
  describe file('/etc/pam.d/common-session') do
    its('content') { should include 'pam_umask.so' }
  end
  describe login_defs do
    its('UMASK') { should eq '077' }
  end
```

```
end
```

4_sysctls.rb:

```
control '4.  Secure kernel (sysctls) configuration' do
  impact 1.0
  title 'Only approved kernel parameters should be used'
  enabled_param = %w(
    net.ipv4.tcp_syncookies
    net.ipv4.icmp_ignore_bogus_error_responses
    net.ipv4.conf.all.log_martians
    net.ipv4.icmp_echo_ignore_broadcasts
    net.ipv4.conf.default.rp_filter
    net.ipv4.conf.all.rp_filter
    net.ipv6.conf.all.disable_ipv6
    net.ipv6.conf.default.disable_ipv6
    net.ipv6.conf.lo.disable_ipv6)
  disabled_param = %w(
    net.ipv4.ip_forward
    net.ipv4.conf.default.accept_redirects
    net.ipv6.conf.default.accept_redirects
    net.ipv4.conf.all.accept_redirects
    net.ipv4.conf.default.secure_redirects
    net.ipv4.conf.all.secure_redirects
    net.ipv4.conf.default.accept_source_route
    net.ipv4.conf.all.accept_source_route
    net.ipv4.conf.default.send_redirects
    net.ipv4.conf.all.send_redirects)
  # Enabled kernel params
  enabled_param.each do |eparam|
    describe kernel_parameter(eparam.to_s) do
      its('value') { should eq 1 }
    end
  end
  # Disabled kernel params
  disabled_param.each do |dparam|
    describe kernel_parameter(dparam.to_s) do
      its('value') { should eq 0 }
    end
  end
end
```

6_services.rb:

```
control '6.  Security related to services' do
  impact 1.0
  title 'Only approved servcices are allowed'
  blacklist = %w(
    autofs
    atd
    avahi-daemon
    bluetooth
    ntpdate
    oddjobd
    qpidd
    xinetd
    openbsd-inetd
    nis)
  whitelist = %w(cron)
  # Not allowed packages
  blacklist.each do |stopservice|
    describe service(stopservice.to_s) do
      it { should_not be_installed }
      it { should_not be_enabled }
      it { should_not be_running }
    end
  end
  # Allowed packages
```

```
  whitelist.each do |startservice|
    describe service(startservice.to_s) do
      it { should be_installed }
      it { should be_enabled }
      it { should be_running }
    end
  end
  # The following services should be also not running. In Debian they are
  # shipped by rsh-server and nis package.
  # rexec, rlogind etc is shipped by rsh-server package in Debian and it
  # executed via inetd, let's make sure that these binaries do not exist
  ['in.rexecd', 'in.rlogind', 'in.rshd', 'in.telnetd', 'ypbind', 'ypserv',
   'in.rdisc'
  ].each do |binaries|
    describe file('/usr/sbin/' + binaries.to_s) do
      it { should_not exist }
    end
  end
end
```

7_protocols.rb:

```
control '7.  Security related to protocols' do
  impact 1.0
  title 'Only approved protocols are allowed'
  blacklist = %w(dccp rds sctp tipc ipv6 netconsole)
  # Not allowed kernel modules
  blacklist.each do |rkmod|
    describe kernel_module(rkmod.to_s) do
      it { should_not be_loaded }
    end
  end
  # ipv6 should be disabled for all interfaces
  describe file('/proc/net/if_inet6') do
    its('size') { should eq 0 }
  end
end
```

IV. Results of Validating the Proof of Concept Solution

Automated security compliance auditing results based on one virtual machine that is created with non-hardened VM image:

```
inspec exec controls/ -t ssh://martin@<ip> -sudo

Target:  ssh://martin@<ip>:22

  ×  3. Secure SSHD configuration: Only approved SSH parameters are allowed (6
failed)
     ✓  SSH Configuration Protocol should eq "2"
     ×  SSH Configuration Banner should eq "/etc/banner"

     expected: "/etc/banner"
          got: nil

     (compared using ==)

     ✓  SSH Configuration IgnoreRhosts should eq "yes"
     ✓  SSH Configuration PermitEmptyPasswords should eq "no"
     ✓  SSH Configuration HostbasedAuthentication should eq "no"
     ×  SSH Configuration PermitUserEnvironment should eq "no"

     expected: "no"
          got: nil

     (compared using ==)

     ×  SSH Configuration ClientAliveCountMax should eq "0"

     expected: "0"
          got: nil

     (compared using ==)

     ×  SSH Configuration ClientAliveInterval should eq "900"

     expected: "900"
          got: "120"

     (compared using ==)

     ×  SSH Configuration PermitRootLogin should eq "no"

     expected: "no"
          got: "without-password"

     (compared using ==)

     ✓  SSH Configuration PasswordAuthentication should eq "no"
     ×  File /etc/banner should exist
     expected File /etc/banner to exist
  ✓  7. Security related to protocols: Only approved protocols are allowed
     ✓  Kernel Module dccp should not be loaded
     ✓  Kernel Module rds should not be loaded
     ✓  Kernel Module sctp should not be loaded
     ✓  Kernel Module tipc should not be loaded
     ✓  Kernel Module ipv6 should not be loaded
     ✓  Kernel Module netconsole should not be loaded
     ✓  File /proc/net/if_inet6 size should eq 0
  ✓  6. Security related to services: Only approved servcices are allowed
     ✓  Service autofs should not be installed
     ✓  Service autofs should not be enabled
```

```
     ✓  Service autofs should not be running
     ✓  Service atd should not be installed
     ✓  Service atd should not be enabled
     ✓  Service atd should not be running
     ✓  Service avahi-daemon should not be installed
     ✓  Service avahi-daemon should not be enabled
     ✓  Service avahi-daemon should not be running
     ✓  Service bluetooth should not be installed
     ✓  Service bluetooth should not be enabled
     ✓  Service bluetooth should not be running
     ✓  Service ntpdate should not be installed
     ✓  Service ntpdate should not be enabled
     ✓  Service ntpdate should not be running
     ✓  Service oddjobd should not be installed
     ✓  Service oddjobd should not be enabled
     ✓  Service oddjobd should not be running
     ✓  Service qpidd should not be installed
     ✓  Service qpidd should not be enabled
     ✓  Service qpidd should not be running
     ✓  Service xinetd should not be installed
     ✓  Service xinetd should not be enabled
     ✓  Service xinetd should not be running
     ✓  Service openbsd-inetd should not be installed
     ✓  Service openbsd-inetd should not be enabled
     ✓  Service openbsd-inetd should not be running
     ✓  Service nis should not be installed
     ✓  Service nis should not be enabled
     ✓  Service nis should not be running
     ✓  Service cron should be installed
     ✓  Service cron should be enabled
     ✓  Service cron should be running
     ✓  File /usr/sbin/in.rexecd should not exist
     ✓  File /usr/sbin/in.rlogind should not exist
     ✓  File /usr/sbin/in.rshd should not exist
     ✓  File /usr/sbin/in.telnetd should not exist
     ✓  File /usr/sbin/ypbind should not exist
     ✓  File /usr/sbin/ypserv should not exist
     ✓  File /usr/sbin/in.rdisc should not exist
   ×  1. Secure file and directory permissions and ownership: Only secure file
and directory permissions and ownership should be used (8 failed)
     ✓  Command find / -xdev -type d -perm -0002 -uid +499 -print stdout should
eq ""
     ✓  File /var/log/auth.log owner should eq "root"
     ✓  File /var/log/daemon.log owner should eq "root"
     ✓  File /var/log/kern.log owner should eq "root"
     ✓  File /var/log/lpr.log owner should eq "root"
     ✓  File /var/log/mail.log owner should eq "root"
     ✓  File /var/log/syslog owner should eq "root"
     ✓  File /var/log/user.log owner should eq "root"
     ✓  File /var/log/messages owner should eq "root"
     ✓  Command find -L /bin ! -user root stdout should eq ""
     ✓  Command find -L /bin -perm /022 -type f stdout should eq ""
     ✓  Command find -L /usr/bin ! -user root stdout should eq ""
     ✓  Command find -L /usr/bin -perm /022 -type f stdout should eq ""
     ✓  Command find -L /usr/local/bin ! -user root stdout should eq ""
     ✓  Command find -L /usr/local/bin -perm /022 -type f stdout should eq ""
```

```
     ✓  Command find -L /sbin ! -user root stdout should eq ""
     ✓  Command find -L /sbin -perm /022 -type f stdout should eq ""
     ✓  Command find -L /usr/sbin ! -user root stdout should eq ""
     ✓  Command find -L /usr/sbin -perm /022 -type f stdout should eq ""
     ✓  Command find -L /usr/local/sbin ! -user root stdout should eq ""
     ✓  Command find -L /usr/local/sbin -perm /022 -type f stdout should eq ""
     ✓  Command find -L /lib ! -user root stdout should eq ""
     ✓  Command find -L /lib64 ! -user root stdout should eq ""
     ✓  Command find -L /usr/lib ! -user root stdout should eq ""
     ✓  Command find -L /usr/lib64 ! -user root stdout should eq ""
     ✓  File /etc/group should be owned by "root"
     ✓  File /etc/group group should eq "root"
     ×  File /etc/group mode should cmp == "00640"

     expected: "00640"
          got: "0644"

     (compared using `cmp` matcher)

     ✓  File /etc/gshadow should be owned by "root"
     ×  File /etc/gshadow group should eq "root"

     expected: "root"
          got: "shadow"

     (compared using ==)

     ✓  File /etc/gshadow mode should cmp == "00640"
     ✓  File /etc/passwd should be owned by "root"
     ✓  File /etc/passwd group should eq "root"
     ×  File /etc/passwd mode should cmp == "00640"

     expected: "00640"
          got: "0644"

     (compared using `cmp` matcher)

     ✓  File /etc/shadow should be owned by "root"
     ✓  File /etc/shadow group should eq "root"
     ✓  File /etc/shadow mode should cmp == "00640"
     ✓  /etc/passwd with password == /^(?!x$)/ entries.length should be == 0
     ✓  Command find / -xdev -type d -perm -002 ! -perm -1000 stdout should eq
""
     ✓  File /boot/extlinux/extlinux.conf owner should eq "root"
     ✓  File /boot/extlinux/extlinux.conf group should eq "root"
     ×  File /boot/extlinux/extlinux.conf mode should cmp == "00600"

     expected: "00600"
          got: "0644"

     (compared using `cmp` matcher)

     ✓  File /boot/extlinux/linux.cfg owner should eq "root"
     ✓  File /boot/extlinux/linux.cfg group should eq "root"
     ×  File /boot/extlinux/linux.cfg mode should cmp == "00600"

     expected: "00600"
          got: "0644"

     (compared using `cmp` matcher)

     ✓  File /boot/extlinux/memdisk.cfg owner should eq "root"
```

```
     ✓  File /boot/extlinux/memdisk.cfg group should eq "root"
     ×  File /boot/extlinux/memdisk.cfg mode should cmp == "00600"

     expected: "00600"
          got: "0644"

     (compared using `cmp` matcher)

     ✓  File /etc/init.d/rc content should match /^umask 022|umask 027$/
     ×  File /etc/pam.d/common-session content should include "pam_umask.so"
     expected "#\n# /etc/pam.d/common-session - session-related modules common
to all services\n#\n# This file is i...ules (the \"Additional\" block)\nses-
sion\trequired\tpam_unix.so \n# end of pam-auth-update config\n" to include
"pam_umask.so"
     ×  login.defs UMASK should eq "077"

     expected: "077"
          got: "022"

     (compared using ==)

  ×  5. Security related to packages: Only approved packages are allowed (1
failed)
     ✓  System Package slapd should not be installed
     ✓  System Package rsh-server should not be installed
     ✓  System Package sendmail should not be installed
     ✓  System Package telnetd should not be installed
     ✓  System Package tftpd-hpa should not be installed
     ✓  System Package xinetd should not be installed
     ✓  System Package openbsd-inetd should not be installed
     ✓  System Package xserver-xorg should not be installed
     ✓  System Package nis should not be installed
     ×  System Package samhain should be installed
     expected that `System Package samhain` is installed
  ×  4. Secure kernel (sysctls) configuration: Only approved kernel parameters
should be used (14 failed)
     ✓  Kernel Parameter net.ipv4.tcp_syncookies value should eq 1
     ✓  Kernel Parameter net.ipv4.icmp_ignore_bogus_error_responses value should
eq 1
     ×  Kernel Parameter net.ipv4.conf.all.log_martians value should eq 1

     expected: 1
          got: 0

     (compared using ==)

     ✓  Kernel Parameter net.ipv4.icmp_echo_ignore_broadcasts value should eq 1
     ×  Kernel Parameter net.ipv4.conf.default.rp_filter value should eq 1

     expected: 1
          got: 0

     (compared using ==)

     ×  Kernel Parameter net.ipv4.conf.all.rp_filter value should eq 1

     expected: 1
          got: 0

     (compared using ==)

     ×  Kernel Parameter net.ipv6.conf.all.disable_ipv6 value should eq 1

     expected: 1
          got: 0
```

```
     (compared using ==)

     ×  Kernel Parameter net.ipv6.conf.default.disable_ipv6 value should eq 1

     expected: 1
          got: 0

     (compared using ==)

     ×  Kernel Parameter net.ipv6.conf.lo.disable_ipv6 value should eq 1

     expected: 1
          got: 0

     (compared using ==)

     ✓  Kernel Parameter net.ipv4.ip_forward value should eq 0
     ×  Kernel Parameter net.ipv4.conf.default.accept_redirects value should eq
0

     expected: 0
          got: 1

     (compared using ==)

     ×  Kernel Parameter net.ipv6.conf.default.accept_redirects value should eq
0

     expected: 0
          got: 1

     (compared using ==)

     ×  Kernel Parameter net.ipv4.conf.all.accept_redirects value should eq 0

     expected: 0
          got: 1

     (compared using ==)

     ×  Kernel Parameter net.ipv4.conf.default.secure_redirects value should eq
0

     expected: 0
          got: 1

     (compared using ==)

     ×  Kernel Parameter net.ipv4.conf.all.secure_redirects value should eq 0

     expected: 0
          got: 1

     (compared using ==)

     ×  Kernel Parameter net.ipv4.conf.default.accept_source_route value should
eq 0

     expected: 0
          got: 1

     (compared using ==)

     ✓  Kernel Parameter net.ipv4.conf.all.accept_source_route value should eq 0
     ×  Kernel Parameter net.ipv4.conf.default.send_redirects value should eq 0

     expected: 0
```

```
          got: 1

     (compared using ==)

     ×  Kernel Parameter net.ipv4.conf.all.send_redirects value should eq 0

     expected: 0
          got: 1

     (compared using ==)


Profile Summary: 2 successful, 4 failures, 0 skipped
Test Summary: 109 successful, 29 failures, 0 skipped
```

Automated security compliance auditing results based on one virtual machine that is created with hardened VM image:

```
inspec exec controls/ -t ssh://martin@<ip> --sudo
Target:  ssh://martin@<ip>:22

  ✓  3. Secure SSHD configuration: Only approved SSH parameters are allowed
     ✓  SSH Configuration Protocol should eq "2"
     ✓  SSH Configuration Banner should eq "/etc/banner"
     ✓  SSH Configuration IgnoreRhosts should eq "yes"
     ✓  SSH Configuration PermitEmptyPasswords should eq "no"
     ✓  SSH Configuration HostbasedAuthentication should eq "no"
     ✓  SSH Configuration PermitUserEnvironment should eq "no"
     ✓  SSH Configuration ClientAliveCountMax should eq "0"
     ✓  SSH Configuration ClientAliveInterval should eq "900"
     ✓  SSH Configuration PermitRootLogin should eq "no"
     ✓  SSH Configuration PasswordAuthentication should eq "no"
     ✓  File /etc/banner should exist
  ✓  7. Security related to protocols: Only approved protocols are allowed
     ✓  Kernel Module dccp should not be loaded
     ✓  Kernel Module rds should not be loaded
     ✓  Kernel Module sctp should not be loaded
     ✓  Kernel Module tipc should not be loaded
     ✓  Kernel Module ipv6 should not be loaded
     ✓  Kernel Module netconsole should not be loaded
     ✓  File /proc/net/if_inet6 size should eq 0
  ✓  6. Security related to services: Only approved servcices are allowed
     ✓  Service autofs should not be installed
     ✓  Service autofs should not be enabled
     ✓  Service autofs should not be running
     ✓  Service atd should not be installed
     ✓  Service atd should not be enabled
     ✓  Service atd should not be running
     ✓  Service avahi-daemon should not be installed
     ✓  Service avahi-daemon should not be enabled
     ✓  Service avahi-daemon should not be running
     ✓  Service bluetooth should not be installed
     ✓  Service bluetooth should not be enabled
     ✓  Service bluetooth should not be running
     ✓  Service ntpdate should not be installed
     ✓  Service ntpdate should not be enabled
     ✓  Service ntpdate should not be running
     ✓  Service oddjobd should not be installed
```

```
     ✓  Service oddjobd should not be enabled
     ✓  Service oddjobd should not be running
     ✓  Service qpidd should not be installed
     ✓  Service qpidd should not be enabled
     ✓  Service qpidd should not be running
     ✓  Service xinetd should not be installed
     ✓  Service xinetd should not be enabled
     ✓  Service xinetd should not be running
     ✓  Service openbsd-inetd should not be installed
     ✓  Service openbsd-inetd should not be enabled
     ✓  Service openbsd-inetd should not be running
     ✓  Service nis should not be installed
     ✓  Service nis should not be enabled
     ✓  Service nis should not be running
     ✓  Service cron should be installed
     ✓  Service cron should be enabled
     ✓  Service cron should be running
     ✓  File /usr/sbin/in.rexecd should not exist
     ✓  File /usr/sbin/in.rlogind should not exist
     ✓  File /usr/sbin/in.rshd should not exist
     ✓  File /usr/sbin/in.telnetd should not exist
     ✓  File /usr/sbin/ypbind should not exist
     ✓  File /usr/sbin/ypserv should not exist
     ✓  File /usr/sbin/in.rdisc should not exist
  ✓  1. Secure file and directory permissions and ownership: Only secure file
and directory permissions and ownership should be used
     ✓  Command find / -xdev -type d -perm -0002 -uid +499 -print stdout should
eq ""
     ✓  File /var/log/auth.log owner should eq "root"
     ✓  File /var/log/cron.log owner should eq "root"
     ✓  File /var/log/daemon.log owner should eq "root"
     ✓  File /var/log/kern.log owner should eq "root"
     ✓  File /var/log/lpr.log owner should eq "root"
     ✓  File /var/log/mail.log owner should eq "root"
     ✓  File /var/log/syslog owner should eq "root"
     ✓  File /var/log/user.log owner should eq "root"
     ✓  File /var/log/messages owner should eq "root"
     ✓  Command find -L /bin ! -user root stdout should eq ""
     ✓  Command find -L /bin -perm /022 -type f stdout should eq ""
     ✓  Command find -L /usr/bin ! -user root stdout should eq ""
     ✓  Command find -L /usr/bin -perm /022 -type f stdout should eq ""
     ✓  Command find -L /usr/local/bin ! -user root stdout should eq ""
     ✓  Command find -L /usr/local/bin -perm /022 -type f stdout should eq ""
     ✓  Command find -L /sbin ! -user root stdout should eq ""
     ✓  Command find -L /sbin -perm /022 -type f stdout should eq ""
     ✓  Command find -L /usr/sbin ! -user root stdout should eq ""
     ✓  Command find -L /usr/sbin -perm /022 -type f stdout should eq ""
     ✓  Command find -L /usr/local/sbin ! -user root stdout should eq ""
     ✓  Command find -L /usr/local/sbin -perm /022 -type f stdout should eq ""
     ✓  Command find -L /lib ! -user root stdout should eq ""
     ✓  Command find -L /lib64 ! -user root stdout should eq ""
     ✓  Command find -L /usr/lib ! -user root stdout should eq ""
     ✓  Command find -L /usr/lib64 ! -user root stdout should eq ""
     ✓  File /etc/group should be owned by "root"
     ✓  File /etc/group group should eq "root"
     ✓  File /etc/group mode should cmp == "00640"
```

```
     ✓  File /etc/gshadow should be owned by "root"
     ✓  File /etc/gshadow group should eq "root"
     ✓  File /etc/gshadow mode should cmp == "00640"
     ✓  File /etc/passwd should be owned by "root"
     ✓  File /etc/passwd group should eq "root"
     ✓  File /etc/passwd mode should cmp == "00640"
     ✓  File /etc/shadow should be owned by "root"
     ✓  File /etc/shadow group should eq "root"
     ✓  File /etc/shadow mode should cmp == "00640"
     ✓  /etc/passwd with password == /^(?!x$)/ entries.length should be == 0
     ✓  Command find / -xdev -type d -perm -002 ! -perm -1000 stdout should eq
""
     ✓  File /boot/extlinux/extlinux.conf owner should eq "root"
     ✓  File /boot/extlinux/extlinux.conf group should eq "root"
     ✓  File /boot/extlinux/extlinux.conf mode should cmp == "00600"
     ✓  File /boot/extlinux/linux.cfg owner should eq "root"
     ✓  File /boot/extlinux/linux.cfg group should eq "root"
     ✓  File /boot/extlinux/linux.cfg mode should cmp == "00600"
     ✓  File /boot/extlinux/memdisk.cfg owner should eq "root"
     ✓  File /boot/extlinux/memdisk.cfg group should eq "root"
     ✓  File /boot/extlinux/memdisk.cfg mode should cmp == "00600"
     ✓  File /etc/init.d/rc content should match /^umask 022|umask 027$/
     ✓  File /etc/pam.d/common-session content should include "pam_umask.so"
     ✓  login.defs UMASK should eq "077"
  ✓  5. Security related to packages: Only approved packages are allowed
     ✓  System Package slapd should not be installed
     ✓  System Package rsh-server should not be installed
     ✓  System Package sendmail should not be installed
     ✓  System Package telnetd should not be installed
     ✓  System Package tftpd-hpa should not be installed
     ✓  System Package xinetd should not be installed
     ✓  System Package openbsd-inetd should not be installed
     ✓  System Package xserver-xorg should not be installed
     ✓  System Package nis should not be installed
     ✓  System Package samhain should be installed
  ✓  4. Secure kernel (sysctls) configuration: Only approved kernel parameters
should be used
     ✓  Kernel Parameter net.ipv4.tcp_syncookies value should eq 1
     ✓  Kernel Parameter net.ipv4.icmp_ignore_bogus_error_responses value should
eq 1
     ✓  Kernel Parameter net.ipv4.conf.all.log_martians value should eq 1
     ✓  Kernel Parameter net.ipv4.icmp_echo_ignore_broadcasts value should eq 1
     ✓  Kernel Parameter net.ipv4.conf.default.rp_filter value should eq 1
     ✓  Kernel Parameter net.ipv4.conf.all.rp_filter value should eq 1
     ✓  Kernel Parameter net.ipv6.conf.all.disable_ipv6 value should eq 1
     ✓  Kernel Parameter net.ipv6.conf.default.disable_ipv6 value should eq 1
     ✓  Kernel Parameter net.ipv6.conf.lo.disable_ipv6 value should eq 1
     ✓  Kernel Parameter net.ipv4.ip_forward value should eq 0
     ✓  Kernel Parameter net.ipv4.conf.default.accept_redirects value should eq
0
     ✓  Kernel Parameter net.ipv6.conf.default.accept_redirects value should eq
0
     ✓  Kernel Parameter net.ipv4.conf.all.accept_redirects value should eq 0
     ✓  Kernel Parameter net.ipv4.conf.default.secure_redirects value should eq
0
     ✓  Kernel Parameter net.ipv4.conf.all.secure_redirects value should eq 0
```

```
     ✓  Kernel Parameter net.ipv4.conf.default.accept_source_route value should
eq 0
     ✓  Kernel Parameter net.ipv4.conf.all.accept_source_route value should eq 0
     ✓  Kernel Parameter net.ipv4.conf.default.send_redirects value should eq 0
     ✓  Kernel Parameter net.ipv4.conf.all.send_redirects value should eq 0

Profile Summary: 6 successful, 0 failures, 0 skipped
Test Summary: 138 successful, 0 failures, 0 skipped
```

V. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Martin Jõgi**,

(*author's name*)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

Establishing, Implementing and Auditing Linux Operating System Hardening Standard for Security Compliance,

(title of thesis)

supervised by Truls Tuxen Ringkjob, Raimundas Matulevičius

(supervisor's name)

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **12.05.2017**