Technische Universität München

Department of Informatics

Master's Thesis in Informatics

Modelling a Secure IT-Infrastructure based on BSI IT Baseline Protection

Michael Peter

Technische Universität München

Department of Informatics

Master's Thesis in Informatics

Modelling a Secure IT-Infrastructure based on BSI IT Baseline Protection

Modellierung einer sicheren IT-Infrastruktur basierend auf dem BSI IT Grundschutz Katalog

Author	Michael Peter
Supervisor	Prof. Dr.-Ing. Georg Carle
Advisor	Dr. Andreas Paul & Dipl.-Inf. (Univ.) Martin Uhl & Dipl.-Inf. Stephan-A. Posselt
Date	August 15, 2015

Informatik VIII
Chair for Network Architectures and Services

I confirm that this thesis is my own work and I have documented all sources and material used.

Garching b. München, August 15, 2015

Signature

Abstract

The goal of this Master's thesis is to automate chosen content of the BSI Baseline Protection Catalogues which give guidelines to secure IT systems.

This thesis illustrates that there is a missing of available automatic support tools and that the catalogues' content includes a large quantity of error-prone and time-consuming aspects making it beneficial to be automated. In this thesis the content of the catalogues were analysed in order to find suitable aspects which are beneficial to automate.

The focus of this work is on network related Baseline Protection safeguards containing recommendations what traffic is allowed in a computer network. In order to automate the evaluation of the implementation status of the selected safeguards, the required input is specified and a network topology model is developed and described. This model is based on directed property multi-graphs including up to four layers of the ISO/OSI layers, allowing to perform checks regarding the chosen safeguards. For the evaluation of the implementation status of the selected safeguards algorithms are designed and illustrated.

Furthermore a modular architecture was designed to read required inputs, build the network model, and evaluate the defined checks.

The thesis is rounded off with an use case implementation and the conclusion that BSI Baseline Protection involves aspects which can be automated and are therefore beneficial for overall network security.

Zusammenfassung

Das Ziel dieser Master Arbeit ist das Automatisieren von Inhalten der Grundschutz Kataloge des BSI. Zweck dieser Kataloge ist das Bereitstellen von Richtlinien zur Absicherung von IT Systemen.

Diese Arbeit beleuchtet das Fehlen von automatisierten Unterstützungstools sowie den umfangreichen Inhalt der Kataloge, bestehend aus fehleranfälligen und zeitintensiven Aspekten. Dieser Umstand macht eine Automatisierung ausgewählter Inhalte naheliegend. Die Kataloge wurden nach zur Automatisierung lohnenswerten und geeigneten Themen untersucht.

Der Fokus der Arbeit liegt im Bereich von Schutzmaßnahmen bezogen auf das Gebiet der Computer-Netzwerke. Im speziellen auf Schutzmaßnahmen welche Empfehlungen hinsichtlich des erlaubten Netzwerkverkehrs beinhalten. Um den Status der Umsetzung dieser Empfehlungen zu prüfen, werden die dafür benötigten Informationsquellen spezifiziert. Weiterhin wird ein Netzwerkmodell entwickelt und vorgestellt. Mit diesem Modell können Evaluierungen von Schutzmaßnahmen die Kenntnisse des Netzwerks voraussetzen, unterstützt werden. Dieses Netzwerkmodell basiert auf gerichteten, mit Eigenschaften versehenen Multigraphen und bildet ISO/OSI Schichten eins bis vier ab. Algorithmen zur Prüfung der Umsetzung der ausgewählten Schutzmaßnahmen werden entwickelt und vorgestellt.

Außerdem wird eine modulare Architektur entwickelt, welche die benötigten Informationen liest, das Netzwerkmodell erstellt und die definierten Prüfungen evaluiert.

Die Arbeit wird abgerundet mit einer Implementierung eines Anwendungsfalls und dem Ergebnis, das es Aspekte in den Grundschutz Katalogen gibt, welche automatisiert werden können und auch nützlich im Bereich der Netzsicherheit sind.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, IT plays an important role in every organisation to reach their business goals but as a consequence it is also an attractive target for attacks as one can see in the media. In the past many security breaches occurred, ranging from companies, e.g. Sony loosing sensitive data [1], to authorities as the German Bundestag [2], or the American OPM [3] where the resulting damage might be even worse. The overall data breach incidents are constantly in amount rising and affecting millions of people [4].

Especially with the rising amount of interconnected devices, network security is an important aspect of every IT department. The IT landscape of even small to medium sized companies consists of dozens or hundreds of network devices, which are connected and configured in some way. A typical network consists of intermediate network objects as switches, routers, firewalls and end devices as servers, virtual machines, and client computers leading to a complex topology where each device must be properly configured in order to maximize IT security.

IT security staff take huge effort in securing networks, as this may directly have impact on business goals, the image of the organization, secrecy and privacy of customers as well as citizens, or even cause human harm. The German Federal Office for Information Security (BSI) supports the implementation of IT security (including network security) by providing the Baseline Protection Catalogues. These are guidelines on how to harden IT-systems including but not limited to computer networks. They describe standard IT setups, their possible threats and the counter-measures called "safeguards" to protect IT systems against certain threats.

Applying these guidelines' network related recommendations manually is error-prone due to the amount of devices which are connected with each other. It is also a complex task as the set of connections itself is large and because objects are connected with different network objects like routers and switches. In other words, the data paths between two devices depend on various components and settings, not just the wiring. Additionally, as a result of these issues the procedure of manually applying the recommendations is time consuming and costly.

1.1 Research Questions

The aim of this Master's thesis is to design and develop a software tool, helping administrators in the field of network and IT security to evaluate some chosen safeguards from the Baseline Protection Catalogues and therefore being compliant to the BSI recommendations (limited to those safeguards). The focus lies in an automated way of support by contrast to the available tools (cp. 3.1). In order to accomplish this task the following issues must be solved:

Q1 Find suitable safeguards to be automated (4.1.1)
The Baseline Protection Catalogues provide plenty of safeguards which must be analysed and sorted out.

Q2 Build a model for verifying selected safeguards (4.2.2)
In order to perform checks according to the recommendations of the selected safeguards, a model of the input data is required.

Q3 Specify the information needed for safeguard checks (4.2.3)
To evaluate the selected safeguards input data must be specified.

Q4 Define checks to (partially) cover selected safeguards (4.2.4)
Checks to evaluate the implementation status for selected safeguards which require the data of Q3.

1.2 Structure

In this work the BSI and the Baseline Protection Catalogues are introduced in chapter 2 as we will work with them. Network information (how devices are connected and what traffic is allowed) is the basic information required to evaluate the selected safeguards, therefore an overview of related work in the field of network analysis is given in chapter 3. In chapter 4 the implementation status of the recommendations of the safeguards to be evaluated as well as the software to be developed are introduced. This chapter contains the requirements of the software and the different input data needed to evaluate

the selected safeguards. Chapter 5 describes the design of the software as well as the key approach of modelling a computer network topology. A possible implementation is described in chapter 6 where a Java prototype application is introduced. A use case with data from a company are illustrated in chapter 7. The benefits and an evaluation is described in chapter 8. At the end chapter 9 gives a conclusion of this thesis and an outlook.

Chapter 2

BSI Baseline Protection Overview

This chapter introduces the German Federal Office for Information Security (BSI) and describes the IT Baseline Protection Catalogues and its role in this thesis.

2.1 BSI

The Federal Office for Information Security abbreviated BSI [5] is the German Federal agency responsible for computer and communication security for the German government. Furthermore, BSI supports the private sector and citizens of Germany in questions of IT security by providing advice and guidelines, e.g. the Baseline Protection Catalogues.

These catalogues describe a methodology, how to guarantee a standard IT security level through all levels of a company's IT systems, ranging from technical to organizational topics. They are illustrated in the following sections.

BSI and Baseline Protection Catalogues are also the basis for a certification according to the ISO 27001 standard [6]. With this certification companies can show to its customers a trustworthy proof of implementing an approved security level to its customers.

2.2 Baseline Protection Catalogues

Purpose of the Baseline Protection (see [7]) is to provide an appropriate basic protection for typical business processes, applications and IT systems. Due to the complexity of IT security the Baseline Protection recommendations adopt a holistic approach covering organizational, personnel, infrastructural and technical aspects. To reduce the complexity of the IT security process and to structure the procedure, the catalogues are

organised in a modular design. Security officers can pick the necessary components according to their needs and perform a risk analysis based on their setup.

2.3 Catalogues Structure

The catalogues are structured in three different components:

1. Modules (count: 80)
   Each module represents a typical business process or IT system. It describes the components and gives an overview of the threats to be considered and the recommended safeguards. Modules itself are divided into 5 categories:

   - Common aspects (e.g. archiving, data protection) are dealing with comprehensive aspects of IT systems applicable to almost all or a great number of systems and/or processes
   - Infrastructure (e.g. Servers room, electrical cabling) is covering the physical and electrical aspects of IT systems
   - IT-Systems (e.g. general server, laptop) are referring to the individual IT system in an information composite. IT systems are technical devices which process data and form a self-contained functional unit.
   - Networks (e.g. modem, WLAN) are dealing with the way objects are connected with each other
   - Applications (e.g. web servers, active directory) are covering the applications running on IT systems

   An example module is "S 3.101 General server" which describes a general server in an IT system. Each module is structured in a brief descriptio, a list of threats, and a safeguard recommendation list.

2. Threats (count: 564)
   Threats exploit possible vulnerabilities of a system compromising certain protective objectives (e.g. confidentiality, integrity). As different IT systems and processes have different threats the Baseline Protection Catalogues list a number of threats which are assigned to applicable modules. Threats are also structured in 5 categories:

   - Basic threats (e.g. fire, data loss) are dealing with threats that may affect all or almost all components of a IT system
   - Force majeure (e.g. storm, burning cables)

- Organisational shortcomings (e.g. lack or insufficient logging, uncontrolled use of resources)
- Human error (e.g. exposed cables, errors in configuration and operation)
- Technical failure (e.g. failure of a database, dusty ventilators)
- Deliberate Acts (e.g. theft, Trojan horses) is covering all (active) attacks on an information system

Basic threats, Force Majeure, Organisational Shortcomings, Human Error, Technical Failure, Deliberate Acts

3. Safeguards (count: 1244)
Safeguards describe the methods to implement to prevent specific threats and are structured as follows:

   - Infrastructure (e.g. locked doors, smoke protection) safeguards to prevent physical threats of IT systems and buildings
   - Organisation (e.g. checking the log files, planning the use of VPN) safeguards are covering the planing and organisation of it systems and its components
   - Personnel (e.g. training, correct behaviour on the Internet) safeguards are dealing with the communication and encouragement of employees.
   - Hardware and Software (e.g. database encryption, handling of USB storage media) covers safeguards related individual IT components and their protection
   - Communication (e.g. secure use of browsers, use of SSL/TLS) safeguards are dealing all aspects of secure communication
   - Contingency Planning (e.g. tests and emergency drills, provisioning of redundant lines) safeguards are dealing with redundancy capacities and backups

   In the context of this thesis a closer look is taken on *safeguards* (see 4.1.1) as it is the main goal to provide a mechanism which automatically evaluates the state of the implementation of certain safeguards and therefore modules and threats are subsidiary.

Chapter 3

Related Work

3.1 BSI Support Tools

The BSI recommends several tools to support the IT security management process [8] and companies which offer support by introducing or performing IT security management processes either with services or tools [9]. BSI offers also checklists to support implementation of safeguards [10]. The BSI itself has provided its own software tool till 2013, which has been discontinued.

Representative for the available software, two tools are presented in the following.

3.1.1 Checklists

This tool provides a list of comma separated (CSV) files, one for each module of the Baseline Protection Catalogues.

The files consists of the following information: The column headers describe the threats listed in the corresponding module. The first column lists the recommended safeguards. The life cycle the safeguard belongs to is listed in the second column. Following abbreviations are used: PD for "Planning and Design", PU for "Purchasing", IM for "Implementation", OP for "Operation", DI for "Disposal", and CP for "Contingency Planning". The third column contains the classification level assigned to the safeguard. There are five levels of classification:

- A (entry) is essential for security reasons and must be implemented with high priority. The implementation of safeguards of level A is required for level B and C.
- B (secondary) are particularly important for establishing an information security that can be monitored and required for level C.

- C (certificate) must be implemented for an ISO 27001 certification based on IT Baseline Protection.
- Z (additional) safeguards of that level must not necessarily be implemented for an audit/certificate but are recommended for high security environments.
- W (knowledge) are intended to help and understand the implementation of other safeguards and are not required for an audit/certification.

An "X" in a field means that the corresponding safeguard is effective in countering the corresponding threat (see A.1 for an example).

Basically these checklists are giving an overview of the threats and safeguards of modules and a direction what needs to be implemented for a certain security level. There is neither automation nor any support figuring out the current status of the safeguards' implementation.

3.1.2 Verinice

Verinice is a multi-platform, open-source software tool to assure the compliance with BSI Baseline Protection [11]. Verinice allows to model the infrastructure by drag and drop (the Baseline Protection Catalogues' content can be imported) the appropriate modules (e.g. server) and manually assigning the required safeguard to them. For example a staff member in charge needs to achieve a level A security in operation with the module "S 3.101 General server ". According to the module description, four safeguards are required to achieve level A of this module: S 2.273, S 4.24, S 4.238, S 4.239 (see [7]). These four safeguards are assigned to the module in Verinice by drag and drop the corresponding safeguards to the module item. At this point a safeguard is neither implemented nor validated. With the modeled modules and their safeguards Verinice allows to generate several kinds of reports which can be delegated to the staff (e.g. network administrators) who are responsible for actually implementing the safeguards.

Generally, those tools only provide check lists (some in a more advanced way) to help those responsible for security management rather than supporting administrators to implement the recommendations. There is minimal automation in necessary data gathering and in security evaluation. The tools are limited to automatically import the content of the catalogues and manually model/assign certain issues of IT systems and generate reports.

The disadvantages of these tools are that they are very time consuming as recommendations, more accurate safeguards, have to be checked manually and with increasing IT size this procedure gets also very error-prone. In addition, due to the fact that safeguards are generally not that precise there might be a problem with different interpretations from the involved persons.

3.2 Network Modelling and Analysis

The focus of this thesis is an automated approach of network modelling and analysis with respect to the chosen context of the Baseline Protection Catalogues. Therefore, it is required to model a network topology and its relations.

3.2.1 Network Analysis Tools

There is a number of tools to support network administrators in managing and analysing networks. *MaSSHandra* [12] is a 3D network visualization tool which also offers the ability to model networks and connections in different floors of a building. *MaSSHandra* offers an auto network discovery and creation of connections.

The Dude is a network monitoring tool which can model networks, monitor services and alerts in case of errors [13] . *The Dude* offers an auto network discovery and layout.

10Strike Network Diagram offers a graphical visualization of networks [14]. *10Strike Network Diagram* offers an auto network discovery and layout including links of devices.

Spiceworks is an all-in-one tool for managing the network inventory, monitoring the network, managing mobile devices and cloud services [15]. *Spiceworks* offers auto network discovery including layout and links, monitoring network stats, managing different types of devices (e.g. Windows/Linux/Mac OS X Laptops/Computers and mobiles), and (software) license management.

The mentioned tools have in common how they gather the network information. They actively scan the network with different methods and different aggressiveness. Most tools heavily rely on SNMP and SNMP enabled devices (e.g. switches) but also other mechanisms such as ping, DNS, ARP or NetBIOS are used. The downside of this approach is, that network configurations might differ e.g. not allowing ping or SNMP requests and therefore the completeness of the discovered network is not guaranteed. Another issue is that those methods doesn't differ that much from attacks. Networks may be configured to block those methods and in the worst case network functionality may be influenced by the scans.

The downside of all mentioned tools in this section is the lack of information needed for a verification of several BSI Baseline Protection safeguards as all of them are not incorporating configuration, e.g. firewall configuration, routing information. Another issue is that those tools just model the current state of the network which might lack important information as machines could be powered down/on or disconnected.

3.2.2 Network Modelling Languages

Van der Ham et al. describes the Network Description Language (NDL) for modelling optical networks [16]. NDL is a language developed to describe optical networks allowing applications to query network capabilities and to answer provisioning questions. After introducing NDL this paper shows its application in some optical and hybrid research networks. Hybrid networks offer heterogeneous services like IP routed traditional connections as well as dedicated circuits (lightpaths). The work is emphasizing NDL's possibility to model the circuit-switched part of hybrid networks as well as management and provisioning tasks in a network.

Dijkstra et al. is also dealing with hybrid networks using NDL [17]. The paper presents NDL and illustrates some network topology examples.

A standardized schema to describe networks is defined from Łapacz et al., developed from the Network Mark-up Language Working Group (NML WG), a group of researchers at the University of Amsterdam [18]. It illustrates the Network Mark-up Language (NML) which was introduced in the field of research on hybrid networks. As NML is an abstract and generic model it can be applied to other networks, too. NML can describe multi-layer (e.g. virtualized) and multi-domain (abstract or aggregated) networks as well as its configuration and capabilities. Aim of NML is to describe logical connection-oriented network topologies rather than physical or packet-oriented networks.

Based on NML Ghijsen et al. introduces the Infrastructure Description Language (INDL) for modelling computer networks which extends the features of NML namely with the ability to provide an additionally syntax for virtual nodes and to describe capabilities (CPU, RAM,...) of components [19] [20] . The papers illustrate the use of INDL in some research projects and the use of extensions to describe energy monitoring.

Figure 3.1 illustrates the model of a sample physical connection between two devices by using INDL/NML. The picture shows the bidirectional wiring of device "b" and "dc1-switch01-02". Both devices have an "in" and "out" port. In and out ports of the devices are connected with two links named "in-out" and "out-in". The entities in the middle of the picture are the bidirectional parts of the connection. "b_e0a(netapp01-b" and "37" are the bidirectional ports of the corresponding device. The two links "in-out" and "out-in" are part of one bidirectional link. "dc1-switch01-02_SS" represents the "switching service" indicating that the device "dc1-switch01-02" is a network switch.

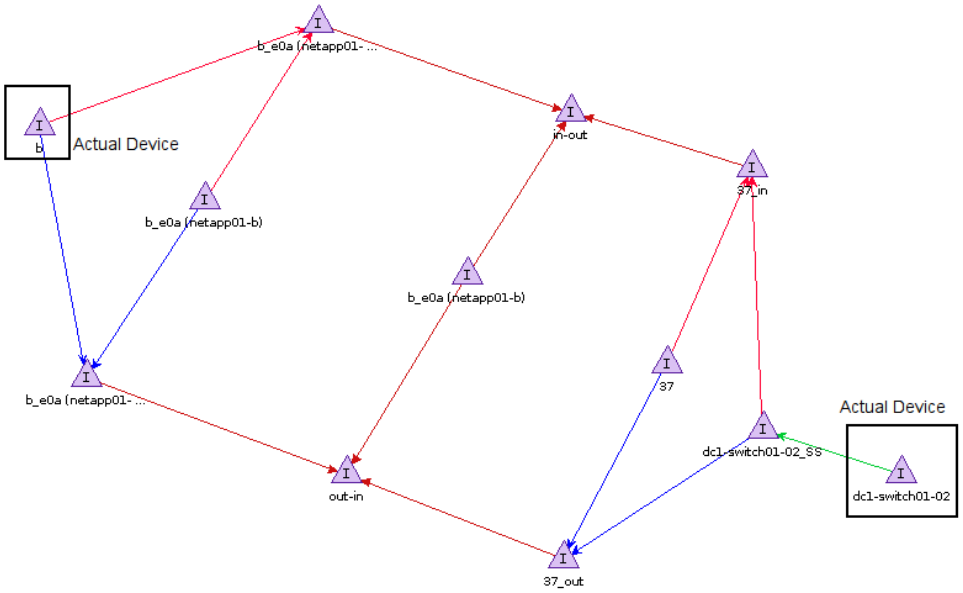

Figure 3.1: One physical connection between two elements in INDL/NML

Chapter 4

Analysis

4.1 BSI Safeguards

In order to automate the processes of the Baseline Protection Catalogues it is necessary to determine safeguards to be automated. This section describes the focus of the safeguards and the selected safeguards.

4.1.1 Safeguards Focus

The focus on the safeguards is as follows:

- It must be possible to automate checks related to the safeguard's recommendations.
- By automating the verifying of the safeguards implementation there should be a benefit for the affected area because manually checking the safeguards' recommendations would be error-prone and complex.
- The selected safeguards are time consuming to manually check, especially in larger environments, hence beneficial to automate

The area of computer networks fulfil all three mentioned aspects. Networks are offering beneficial possibilities to automate error-prone, manual tasks with rising complexity in larger environments. Therefore, there is a benefit by automating the implementation status of network related safeguards as it saves time and costs and is less error-prone. In order to further refine the focus *not* to be covered are topics related to:

- Infrastructural and management-related aspects
  All kinds of building and room security such as locking server room doors or organizational related topics such as employees' awareness in handling potentially malicious emails are not covered.

- Applications
  Application security such as a secure configuration of a database or mail server is not covered.

- Communication other than Ethernet-based networks
  Only safeguards referring to Ethernet-based communication is covered because Ethernet networks are in most companies the standard communication. Therefore, not covered are communication methods such as ISDN and RS-232.

These points are not regarded for the following reasons: On the one hand as mentioned, there should be an emphasis on (computer) infrastructural topics not in terms of building security, room/server rack locking or disaster protection. On the other hand organizational and management-related aspects are difficult to automate or could be broken down to assertions like

> ”Is there a person responsible for evaluation of log data?”

(see [7] safeguard S 2.64 Checking the log files).

4.1.2 Selected Safeguards

In the following the selected safeguards based on the former focus points are briefly described (full text in [7]):

4.1.2.1 Selection and Implementation of Suitable Filter Rules (Safeguard S1)

This safeguard refers to the packet filter rules for a security gateway also known as firewall. The main recommendations related for this work are as follows (see S 2.76 in [7]):

> “As a basic principle, the white list strategy should be applied [..]”

> “If there is a need for user-specific authentication, it must be clarified which users from the internal network may use which services [...]”

> “All computers in the internal network must be taken into consideration.”

> “The filter rules for packet filters should be summarised in a table, one axis of which reflects the destination IP addresses and the other axis of which reflects the source IP addresses.”

4.1.2.2 Secure Operation of a Firewall (Safeguard S2)

The safeguard's main recommendations to be checked (see S 2.78 in [7]) are similar to the one former presented:

> "The rule "everything not expressly permitted is prohibited" must be implemented"
>
> "It should be checked regularly whether new accesses bypassing the security gateway have been created."

4.1.2.3 Change of Preset Passwords (Safeguard S3)

This safeguard recommends that standard/preset passwords of network devices (e.g. management access of firewalls) must be changed (see S 4.7 in [7]).

> "The default password settings specified by the manufacturer or administrator should be changed immediately after installation, and if not at that time, then before the hardware or software is put into operation for the first time."

4.1.2.4 Secure use of Protocols and Services (Safeguard S4)

This safeguards recommendations can be summed up in implementing proper filter rules at the firewall and therefore check what traffic is allowed/forbidden (see S 5.39 in [7]).

> "Is there an overview of the protocols admitted at the security gateway?"
>
> "Depending on the operational scenario, certain ICMP message types should be allowed or blocked selectively"

4.1.2.5 Configuration of Access Control Lists on Routers (Safeguard S5

A similar safeguard to S1 describing the mechanism of ACLs (Access Control Lists) to control network access and traffic (see S 5.111 in [7])).

> "The ACLs must be defined in accordance with the specifications of the security policy."
>
> "The more restrictive white list approach should be preferred in general during configuration [...]"
>
> "[...] ACLs must always be configured in such a way that rejected access attempts are logged."

4.1.2.6 Handling of ICMP on the Security Gateway (Safeguard S6)

This safeguard describes the handling of ICMP (Internet Control Message Protocol) messages at the security gateway (see S 5.120 in [7]). The safeguard distinguishes between “Computers in the internal network” and “"Public" servers in the DMZ”. The recommended ICMP handling for both is listed in a table containing the ICMP type, the direction of the message (incoming/outgoing) and the action (permit/deny), see tables 5.2 and 5.3.

Table 4.1 lists the chosen safeguards and their checks to be performed as well as the required input information.

4.2 Software to be Developed

For the purpose of automating the evaluation of safeguards according to table 4.1 it is required to get the needed information and build a model from these data on which the checks to be evaluated perform.

4.2.1 General Requirement (General Requirement G1)

For further extensions and different inputs the architecture to be developed should be designed with respect to modularity and extensibility. The architecture shall be independent from different input formats.

4.2.2 Build Model

To perform operations on the gathered data an appropriate model shall be built from the different kinds of information formats.

4.2.2.1 Build a Representation of the Network Topology (Model requirement M1)

In order to perform validation on a network, it is necessary to know the topology meaning to know how machines are connected with each other and how possible traffic paths look like. The model of the network shall be:

1. Machine readable
   The aim is to develop software and therefore the model must be understandable by a computer.

Table 4.1: Overview of Safeguards and Required Inputs

Nr.	Safeguard	To be automated	Inputs needed
S1	S 2.76 Selection and implementation of suitable filter rules	- whitelist approach compliance - user specific traffic settings - considering all internal computers	- firewall configuration - network topology
S2	S 2.78 Secure operation of a firewall	- whitelist approach - devices able to bypass the firewall - no preset password for firewall admin user	- firewall configuration - network topology
S3	S 4.7 Change of preset passwords	- no preset password for admin user	- configuration of intermediate network devices (e.g. switch, router, firewall,...)
S4	S 5.39 Secure use of protocols and services	- handling of ICMP - TCP and UDP differentiation in filter rules	- firewall configuration - network topology
S5	S 5.111 Configuration of access control lists on routers	- which computer has what (network related) capabilities - whitelist approach - firewall logging of rejected traffic	- firewall configuration - network topology
S6	S 5.120 Handling of ICMP on the security gateway	- ICMP messages limited/blocked on the security gateway	- firewall configuration - network topology

2. Suitable for physical and virtual network objects
   The model must handle physical computers as well as virtual machines.
3. Able to model connections between different kinds of network objects
   The model must be able to model different layers of the ISO/OSI model, e.g. data layer or network layer.
4. Able to model objects and connection properties
   The model must be able to store information about network devices and connections, e.g. an IP address of a devices or a VLAN id of a connection.

In chapter 3.2 a closer look was taken at INDL and NML. At first glance both look suitable for this thesis as they offer:

- Machine readable RDF/XML format
- Application independent exchange format
- A layer model
  A layer model is an advantage in terms of safeguards recommendations as some checks require a differentiation of network layers, for example S2. In safeguard S2 one recommendation is that devices are not allowed to communicate with each other without a firewall between them. In a layer model it is possible to check this recommendation through all layers, e.g. if there is a firewall at the physical layer or if there is a firewall at the network layer.
- Virtualization

Nevertheless, INDL and NML do not offer a benefit for our needs (performing checks of the status of the implementation of the recommendations):

- Networks modelled in INDL and MNL have a high amount of entities. For a basic physical connection between two devices as illustrated in 3.2.2 twelve entities and 15 edges (relations in OWL) need to be created. For modelling the next layer at least the same amount of entities and edges are added as INDL/NML build the layer on top. That is, additionally to the entities of the lower layer, edges (relations) for the upper layer are created. These relations provide the connection of both layers as INDL and NML model networks "vertically" (imagine as a stack of layers). The created layer on top of the previous layer adds another set of unidirectional and bidirectional entities. The high amount of entities is justified by modelling in and out bound ports which is not required in the context of safeguards' recommendations. For our needs it is not necessary to model respectively distinguish between out- and in-bound ports.
- Furthermore, both models are designed to be strictly unidirectional (justified in the origins in the area of optical and hybrid networks) in their use case which is a drawback for our networks as we will only consider Ethernet-based bidirectional

networks. As unidirectionality is a basic principle of the model of INDL and NML it is an integral part of the model.

- Additionally, INDL and NML model device internal interconnecting ports (e.g. in switches). To actually use the model in our context an additional abstraction layer is required which provides functionality to encapsulate those aspects. That is, the abstraction must provide a function which transforms the unidirectional approach into a bidirectional approach. Furthermore, the abstraction must provide a functionality which handles the device internal connections to provide a path finding in the network.
- There is no specific tool support or wide spread (public) usage. Despite intensively research no tools or libraries which handle or work with INDL and NML were found. Although it is a standard the commonness is not high and INDL and NML itself still appear to be an area of research. Therefore, the benefit of being standardized and offering an exchange format is no real benefit for INDL and NML.

For this reasons and that the focus on this thesis is validating safeguards the approach is to develop a specific network model which fulfils our needs.

4.2.2.2 Model Firewall Filter Rules (Model Requirement M2)

Firewall filter rules are required for the network topology model as they can limit data paths. Additionally, they are used to evaluate checks C4 and C5 (see in the following section). Therefore, an appropriate model to store and access the filter rules is required.

4.2.2.3 Model IT Policies (Model Requirement M3)

The IT policies are required to compare the requirements of the company to those actually implemented in the firewall filter rules. Therefore, an appropriate model to store and access this data is required.

4.2.3 Read the Required Inputs (Input Requirement)

This paragraph describes the different inputs required to perform the checks C1-C7.

4.2.3.1 Network Cables I1

Cables as physical connections between machines must be considered as first layer in the network model to be developed (and in the ISO/OSI model). For this thesis only

Ethernet-based connections are considered. The cabling should be documented in an appropriate way to be parsed by a specific parser providing a set of links and devices from the whole network.

4.2.3.2 Virtual Machines I2

Virtualization plays an important role in IT infrastructure due to the benefits (e.g. easy and fast provisioning, scalability, maximum resource utilization). There are several companies and technologies offering virtualisation solutions (e.g. VMware, KVM, Citrix, Proxmox, ...). As virtual machines can be treated as a "normal" physical server they must be taken into consideration, too. To reduce the complexity of modern hypervisors with respect to (virtual) networking a simplified model which will be suitable for our needs is developed.

4.2.3.3 Switch I3

To determine possible traffic paths switches play an important role. Switches can create network segmentation on layer two of the ISO/OSI model by VLANs. If packages from one VLAN are not switched to another one (dropped by the switch) there would be no need to check firewall filter rules as they are usually working on layers three and/or four and will not receive the packages. A switch configuration must provide the mapping of (physical) interfaces a device is connected to, to the VLAN id. One interface can be mapped to several VLAN ids. Several switches offer link aggregation meaning that different physical interfaces are bundled which must be taken into consideration. Additionally, the possibility of stacked switches (meaning several physical devices act as one switch) must be modelled.

In addition to the information required for modelling the data layer in the network model, a parser to be developed must read the password of the management interface of the switch(es).

4.2.3.4 Router I4

After layer one and two another important component is the routing (end-to-end connection) in a network (layer three). Similar to switches on layer two routers may drop packages and therefore there is no need to evaluate data at the firewall level. According to that the routing paths in a network must be available for making statements about the paths of the network.

In addition to the information required to model the data layer, a parser must read the password of the management interface of the router(s).

4.2.3.5 VPN Gateway I5

VPNs form a virtual network in an existing network. They are used to provide access for external networks to internal networks, to build a private WAN without the costs of a real one, and to grant encrypted access to resources based on a user authentication. To distinguish possible traffic paths a network topology model should include VPNs as well. VPN information must be parsed form a configuration (e.g. the VPN gateway) with a specific parser to create VPN routes in the network model to be developed.

In addition to the information required to model the data layer, a parser must read the password of the management interface of the VPN gateway(s).

4.2.3.6 Firewall I6

To determine allowed traffic paths in a network, it is necessary to gather information from the firewalls which are responsible for allowing and dropping packages in the network. In this context the most important information are the defined filter rules of the security gateway. A rule's minimal set of information for an IP based firewall is *Source IP*, *Destination IP*, *Source Port*, *Destination Port*, and *Action*.

Incoming packets are based on their source and destination information compared to the filter rule table and the first rule which matches all four parameters is applied.

In addition to the information required to model the data layer, a parser must read the password of the management interface of the firewall(s) and check if logging of traffic activities is enabled.

4.2.3.7 Read IT Requirements I7

The last input needed is the specification of allowed (or in same case forbidden) traffic in some kind of rules. BSI Baseline Protection Catalogues do not specify exact (filter) rules for networks. Instead they give examples in which way one should define filter rules but clearly postulate that there should be such a document (cp. S 2.76 [7]).

Table 4.2 lists the described requirements.

4.2.4 Perform Checks

Based on the network topology model and the recommendations of the safeguards, several checks shall be performed according to the column *to be automated* in table 4.1.

Table 4.2: Requirements Overview

Requirement	**Summary**
G1	Modular and extensible architecture
M1	Network topology model
M2	Firewall filter rules model
M3	IT policies model
I1	Physical wiring
I2	Virtual machines
I3	Switch configuration
I4	Router configuraion
I5	VPN Configuration
I6	Firewall Configuration
I7	IT Policy

- Check Preset Passwords of Network Devices (Check C1 "Passwords") This check is from safeguards S2 and S3. The aim is to check management passwords of network devices.
- Check if Firewall Logging is Enabled (Check C2 "Logging")
  This check is from from safeguard S5. The aim is to check the logging of dropped/accepted packages of firewall filter rules.
- Find Connections Bypassing a Firewall (Check C3 "Bypassing")
  This check is from safeguard S2. The aim is to check data paths allowing communications between devices without control of at least one firewall.
- Check the Existence of Filter Rules for All Devices (Check C4 "Filter Rules")
  This check is from safeguard S1. The aim is to check the existence of a filter rule for every device in the network.
- Check the Whitelist Approach in Filter Rules (Check C5 "Whitelist")
  This check is from safeguards S1 and S2. The aim is to check that the filter rule table has no deny rule.
- Find Possible (TCP/UDP) Data Paths and Compare them to the IT Requirements (Check C6 "Policy")
  This check is from safeguards S1, S4, and S5. The aim is to check allowed traffic in the network and compare them to the IT policies.
- Check the Handling of Different ICMP Types (Check C7 "ICMP")
  This check is from safeguards S4 and S6. The aim is to check the handling of ICMP at the firewall.

Chapter 5

Design

This chapter describes the design of the architecture to be developed as well as the model of the network topology and the checks in order to evaluate the implementation status of the selected safeguards.

5.1 Purpose

The architecture to be developed should be able to automate the verification of chosen BSI Baseline Protection's safeguards. It should gather the information required for the specified safeguards and store it in an appropriate data structure. With this data, checks related to the chosen safeguards should be performed.

5.2 Architecture Overview

Figure 5.1 gives a rough overview of the components and their relationship of the architecture to be developed. According to the requirements (see 4.2) there are several major sets of data which are described in section 4.2.3. From those inputs the needed information is gathered and internal models are built (see 4.2.2). The different checks (from 4.2.4) are described at the end of this chapter.

The architecture has three major inputs

1. The network data (see section 4.2.2.1)
2. The firewall configuration (see section 4.2.3.6)
3. IT policies (see section 4.2.3.7)

The model is generated with theses inputs containing

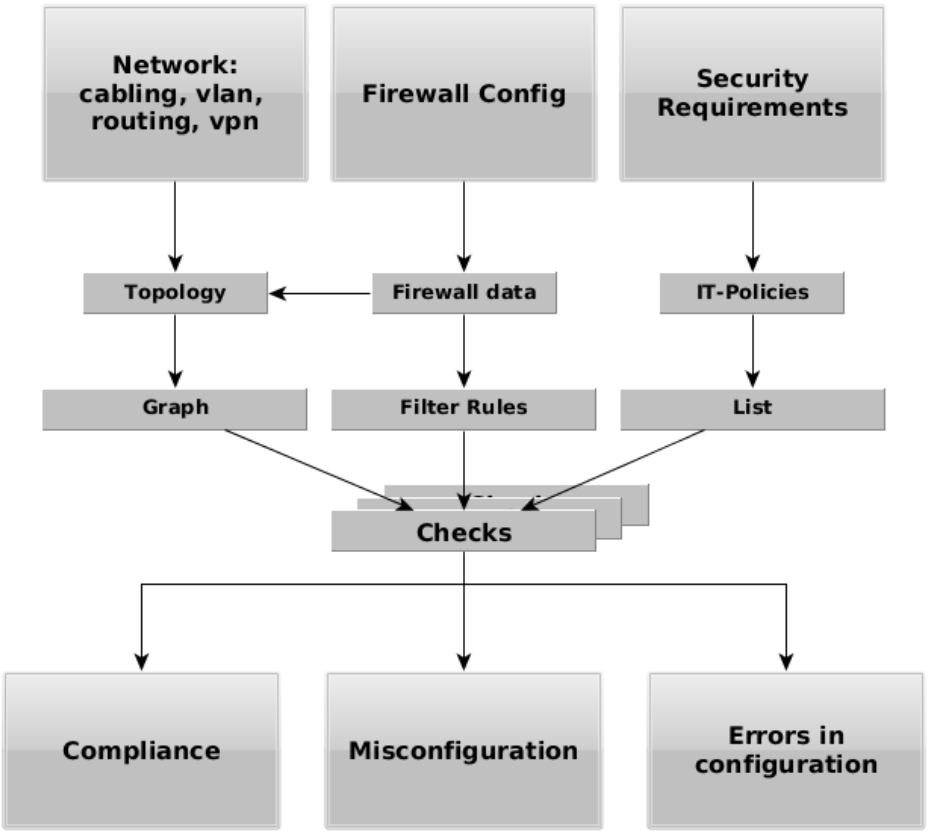

Figure 5.1: Components - Overview

- A network model from layer one to three with path finding capabilities
- Firewall filter rules (including layer four)
- IT policies

The following sections describe the algorithms and the design of the architecture to be developed according to the requirements (see 4.2).

5.3 General Requirement G1

This architecture is designed to differentiate between the data (-sources) and the internal model. Therefore, it is possible to extend the functionality with other sources respectively parsers as long as they provide the necessary data for the network topology, the firewall filter rules, the IT policies, and the IPv4 space. The design also separates the checks making it easy to add other checks as they perform independent of each other.

5.4 Build Model

In the following, the different models are described according to 4.2.2.

5.4.1 Build a Representation of the Network Topology (M1)

For the evaluation of several safeguards a sophisticated knowledge of the network topology is required. This section describes the model and its entities.

5.4.1.1 Influences from INDL and NML

This section describes the influences on the graph model from INDL and NML (see 3.2.2):

- INDL's generic approach by providing "Deadaptation" and "Adaptation" services for modelling the transition between different layers is adopted. This is achieved by building several layers of the actual graph representing the different ISO/OSI layers.
- Everything in NML is unidirectional. Basically that is the right way especially when handling (unidirectional) firewall connection establishments. Therefore, the concept of unidirectional links was adapted to a directed graph model.
- INDL and NML model ports of devices and their connections by creating entities for each port and connection. This concept was adopted and simplified but storing the necessary information (ports, connection, etc.) in the property of the connection, respectively the edges of the graph.

5.4.1.2 Components of a Network Graph

Figure 5.2 illustrates the components of a network topology as it is to be modelled (compare the following sections for detailed description of the components). Network devices such as switches, routers, VPN gateways, and firewalls determine the possible paths of traffic. End devices as computers and virtual machines are traffic producer and consumer and the actual devices to be restricted in terms of network access. These components cover most standard enterprise networks only varying in their configuration and number.

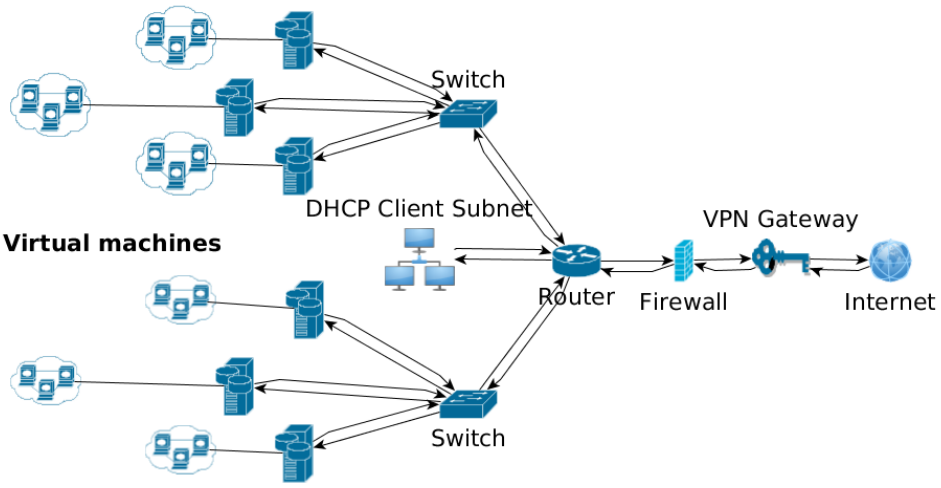

Figure 5.2: Components of an Example Network

5.4.1.3 Network Layer Modelling Approach

In order to perform an evaluation of the selected BSI safeguards a model of the required Inputs is needed. All connections are unidirectional as a directed graph is basis for the graph model. This allows differentiating between incoming and outgoing traffic. As there might be parallel edges the graph model is a (directed) multigraph. The different layers of a network are modelled as a standalone graph based on the graph of the underlying layer allowing to easily determine paths. Due to the relation that the set of edges of layer n+1 is based on the set of edges of layer n one can assume that paths of layer n are included in layer n+1. Nodes of the graph represent the different network elements, e.g. a switch or a router.

Each layer substitutes intermediate network device (e.g. switches, router,..) and extends the graph with edges representing the network topology information of the substituted device.

The benefit of this approach is to simplify the algorithms that work on this model. Algorithms which require searching for a data path between two devices just need to verify the existence of an edge between those devices.

Figure 5.3 illustrates the different layers of the network model, their inputs, relations and node types. For edge information see the corresponding section of the detailed layer description. Each layer depends on the underlying layer and the corresponding configuration inputs.

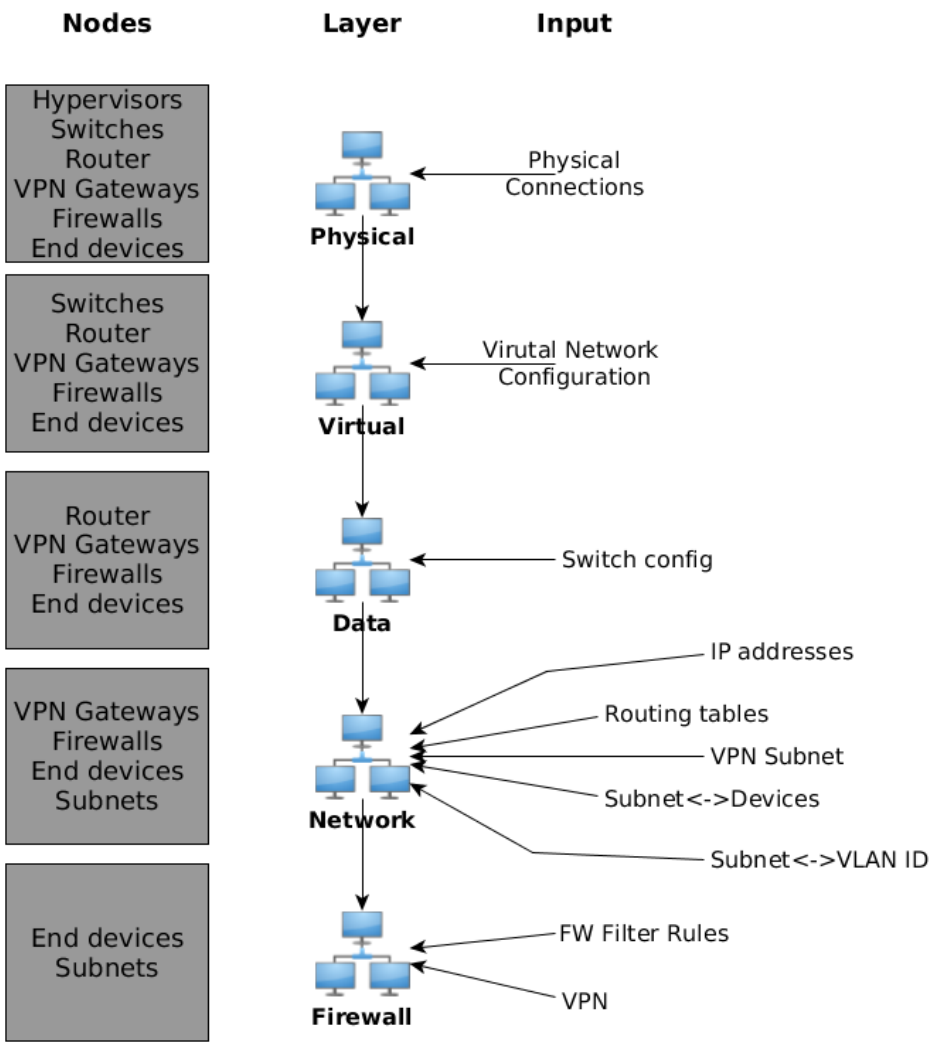

Figure 5.3: Overview - Network Layer Model

5.4.1.4 Physical Layer

For the physical layer the connections of (hardware) devices are needed. The graph consists of the following elements:

1. Nodes (representing network objects) with *name* and *type* values
   Possible values for types are *switch*, *router*, *vpn gateway*, *firewall*, *hypervisor*, and *end devices*

2. (Directed) Edges (representing the connections) with *name*, *nodes*, and *ports* values
   name is a (unique) name of the edge, *nodes* are the references to the two nodes

which are linked by this edge and *ports* are the port names of both nodes the link is attached to.

Figure 5.4 illustrates an example graph of a small network and its objects connected with cables.

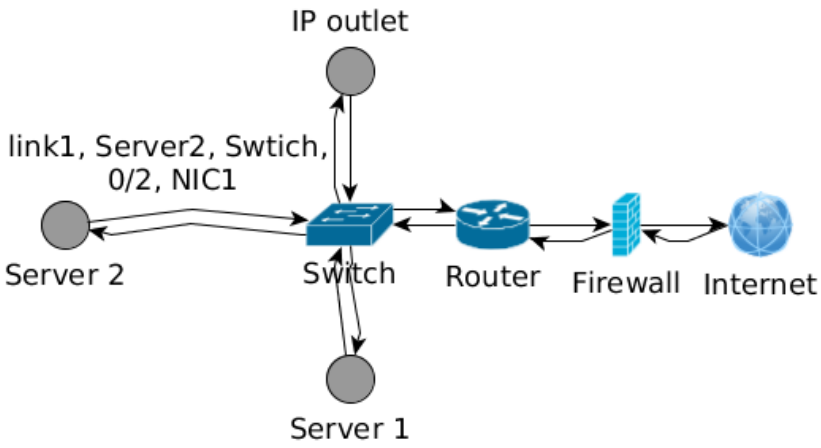

Figure 5.4: Layer 1 Example Graph Representation

5.4.1.5 Virtual Machines

Since all devices consuming and producing network traffic should be considered, virtual machines must be modelled in the network topology, too. The problem is where to locate them in the network graph, as they are not physical hardware but software. At first glance, virtual machines should be part of the corresponding host (in the context of virtualisation host refers to hardware server). The downside of this approach is that some hypervisors (e.g. VMware) have mechanism to dynamically spread the location of virtual machines across different hosts (clusters) in terms of resource usage optimizing so it is generally impossible to make exact statements where a virtual machine is located.

For this reason only the "snapshot" of the current state is regarded. As modern hypervisors implement a virtual switch connecting the VMs and the host's NICs we model virtualisation before the data layer. Aim is to extend the network graph with the virtual network provided by the virtualisation software by creating new nodes for each VM and virtual switch. Virtual switches can be treated like physical switches in the model that is, they are substituted in the next layer, too.

For this layer the physical layer and the virtual network configuration are needed. The graph consists of the following elements:

1. Nodes with *name* and *type* values
   Possible values for types are *router*, *vpn gateway*, *firewall*, *vm*, *end devices*. There is no special type for virtual switches as they are considered as physical switches in the model.

2. (Directed) Edges with *name*, *nodes*, and *ports* values This edge is similar to an edge of the physical layer.

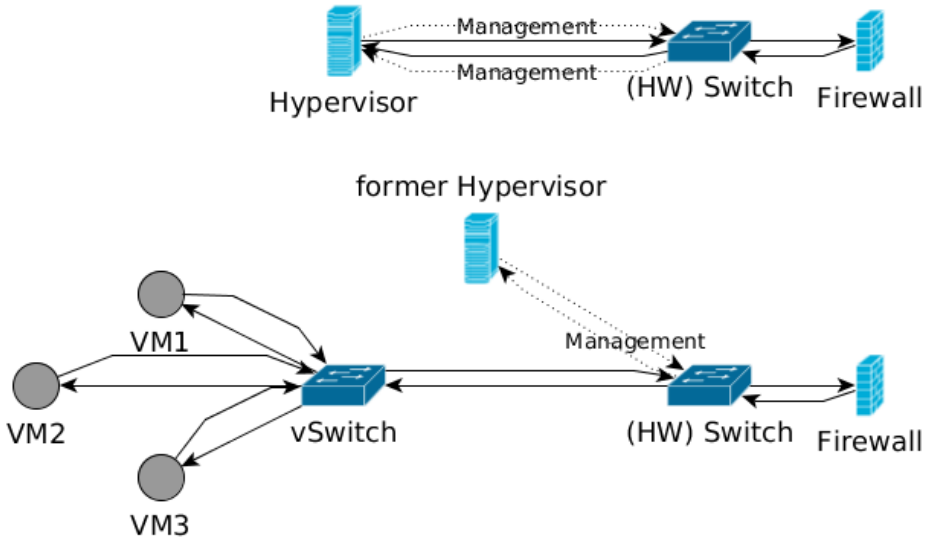

Figure 5.5: Virtualisation Model

Figure 5.5 illustrates the idea. Starting point is the physical connection of a hypervisor to another network device (here a hardware switch). The hypervisor's type is replaced by "former hypervisor" just connected with its management interfaces indicating that this node now acts as a regular end device in the model. The virtualisation is modelled by a newly created node "vSwitch" to which the different virtual machines of the hypervisor are connected. The (physical) hosts' interfaces (except management) are "replaced" and the edges from the hardware switch directly connected with the new vSwitch. This is possible as the physical interfaces of the host must be transparent for the VLANs of the virtual machines as otherwise there would be no connection possible. At this point the model consists of intermediate network devices (e.g. switches, routers,...) and end devices (VMs amongst others). With this state of the graph it is possible to apply the next step, the building of the data link layer.

As there are many virtualisation technologies available, the source and format of the required information differs. The following list describes the information in general that is needed from the virtualisation software for the model:

1. Name and VLAN configuration of each virtual switch per hypervisor

vSwitches substitute hypervisor nodes and act as intermediate network devices between the physical network and the virtual machines. Therefore, the VLAN configuration of each vSwitch similar to normal switches (see 4.2.3.3) to being able to model the data link layer of the virtualised machines is needed. Additionally, the physical interface of the host connected to the vSwitch is needed for the mapping of the physical wiring to the vSwitch.

2. Name and connections of each virtual machine
   To create new nodes for each running virtual machine the information how each VM is connected to which virtual switch is required.

5.4.1.6 Data Layer

For this layer the physical layer and the switch configurations are needed. The graph consists of the following elements:

1. Nodes with *name* and *type* values
   Possible values for types are *router*, *vpn gateway*, *firewall*, and *end devices*. The difference to the former layer is the missing of nodes of the type switch.

2. (Directed) Edges with *name*, *nodes*, *ports*, and *VLAN id* values
   Additionally, to the layer one graph the edges have the VLAN property which describes the edge's VLAN id.

Figure 5.6 illustrates an example graph. The switch device changed to a regular end device of type "former switch" as it might have a management interface and must be considered although it is meaningless for the other devices' communication.

The difference to the former graph is, that this graph includes VLAN information, e.g. *Server 2* is now connected with the *IP outlet* by two links, each of them having a (different) VLAN property.

5.4.1.7 Network Layer

For this layer the data layer, the routing tables of each router, a VLAN id to subnet IP mapping, a subnet to devices mapping, and the client subnets as well as the VPN subnets are needed. The graph consists of the following elements:

1. Nodes with *name*, *type*, *VLANid*, *subnet* and *IP* values
   Possible types are *firewall*, *vpn gateway*, *subnet*, and *end devices*. The difference to the former graph is the missing of node of the type *router*.

2. (Directed) Edges with *name*, *nodes*, *ports*. *VLAN*s, *IP1*, *IP2*, *subnet1*, *subnet2*, and *isVPN* values

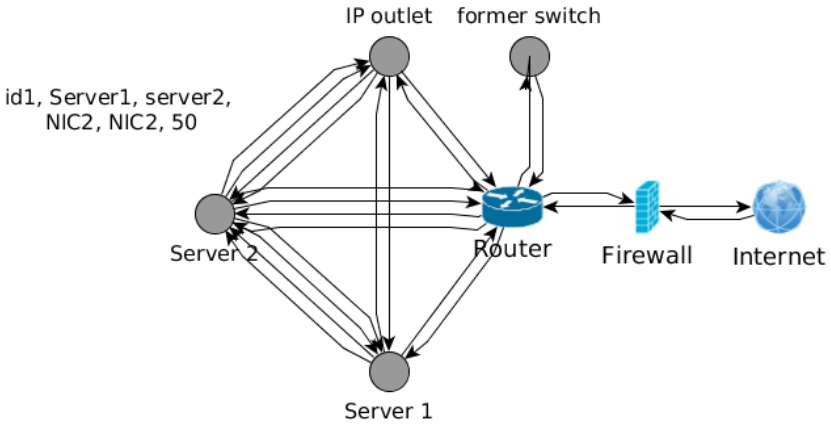

Figure 5.6: Layer 2 Example Graph Representation

Additionally to the values of the edges of layer two, the following new properties are added:

- *IP1* describes the IP of node1
- *IP2* describes the IP of node2
- *subnet1* describes the subnet of IP1 of node1
- *subnet2* describes the subnet of IP2 of node2
- *isVPN* indicates if this connection is a VPN route

VPN Gateways assign a new IP address to clients from a specific set of reserved addresses. As these addresses for VPN clients may currently not exist in the graph (as nodes) all regarded routes must be checked for an existent node in the graph. If the node does not exist, it must be checked if the subnet, which is the routing target is a member of the defined VPN subnet(s) and a new subnet node for the client VPN IP(s) is created. This approach is similar to the model of the DHCP-based client IP range.

Figure 5.7 illustrates an example graph (for better visualization only the new values of the edge are shown). The router was substituted by the possible routing paths according to the router configuration. In this case only "Server 2" is routed by the router to the firewall

As routers might also route DHCP based networks where an exact IP address of a client is not determinable a new node of type "client subnet" is in this case created.

The former router device changed to a "former router" device as it might have a management interface and must be considered although it is meaningless for the other devices' communication. The difference to the former graph is, that this graph includes layer

three (routing) information, e.g. *Server 2* is now connected with the *Firewall* by an edge with the required layer three information.

The difference to the former graph is that this graph contains layer three routing information, that is one edge represents an end-to-end connection in contrast to the real connection might consist of several hops.

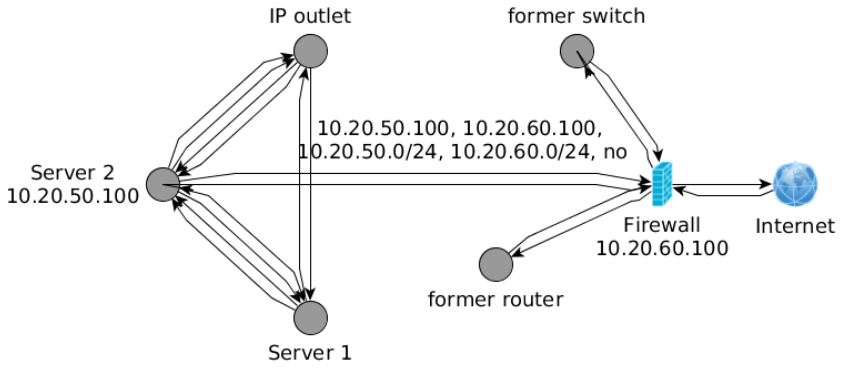

Figure 5.7: Layer 3 Example Graph Representation

5.4.1.8 Firewall Model

This "layer" is not an ISO/OSI layer but as filter rules, which are a main focus of the considered safeguards, are applied on the network and the transport layer (for TCP/IP firewalls), they are modelled at this point. The general algorithmic approach is further continued with the aim of substituting firewalls (filter rules) with edges in the graph representing their information. The basis for modelling filter rules are the network layer graph and the filter rules of the firewall configuration as well as the VPN endpoint configuration. For each filter rule with *allow* action a new edge is created with the appropriate information. Every allow rule (just matching the source IP) is modelled to get a set of edges representing all possible traffic paths. Deny rules are modelled as limitation of the subsequent allow rules. The graph has the following elements:

1. Nodes with *name*, *type*, *vlan id*, *subnet*, and *ip* values
   Possible values for types are all types of end devices and *subnet*s (for DHCP and VPN addresses).

2. (Directed) Edges with *name*, *nodes*, *ports*. *VLAN*s, *subnet1*, *subnet2*, *srcip*, *dstip*, *protocol*, *srcport*, *dstport* values
   Basically representing a layer three and four connection. Instead of *IP1* and *IP2* *srcip* and *dstip* is used to express the firewall affiliation of this edge. The additional properties are:

- *protocol* for the allowed protocol (TCP or UDP) *srcport* for the source port (not to be confused with ports of devices), e.g. 80 for an HTTP request *dstport* for the destination port

3. (Directed) Edges with *name*, *nodes*, *ports*. *VLAN*s, *subnet1*, *subnet2*, *srcip*, *dstip*, *icmptype*
   For modelling ICMP handling of the firewall (same as with filter rules, an edge meaning this ICMP type is allowed). Instead of *protocol*, *srcport*, and *dstport* the additional value *icmptype* is added to the edge. *icmptype* contains the type of the ICMP message, e.g. type 8 for a ping.

4. (Directed) Edges with *name*, *nodes*, *ports*. *VLAN*s, *subnet1*, *subnet2*, *srcip*, *dstip*, *protocol*, *srcport*, *dstport*, and *user* values
   For VPN connections the additional user property is added for storing the information which users are allowed.

As allowed traffic in a VPN is usually limited by a firewall it must be considered at this point. Every time an edge connected to a VPN subnet (meaning the edges' value of *isVPN* is true) is regarded, the VPN settings must be considered e.g. the users and the protocol allowed from this VPN (subnet) IP.

Figure 5.8 illustrates an example graph. In this graph the "former switch" and the "former router" are allowed to communicate with "Server 2" (for better visualization the properties of the edges are not shown) and "Server 2" is allowed to communicate with the node "Internet" with TCP over port 22 in both directions.

The difference to the former graph is that this graph contains firewall filter rules meaning that this graph models the allowed traffic in the network.

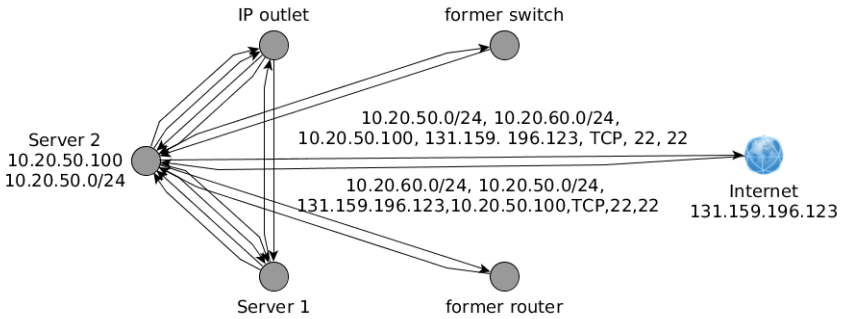

Figure 5.8: Firewall Filter Rules Representation

5.4.2 Algorithms

The following section describes the algorithms to build the network model. Section 5.4.2.1 describes the basic algorithm to substitute network devices in order to extend the graph with their information. Sections 5.4.2.2 5.4.2.3, 5.4.2.4, 5.4.2.5, and 5.4.2.6 describe the algorithms to model the corresponding layer. For better visualization of the algorithms, the mentioned fact that most devices remain in the graph due to their management interface is not shown.

5.4.2.1 Basic Algorithm for Substituting Network Devices

For each layer the aim is to substitute intermediate network devices with a number of edges which represent the replaced devices' functions. For all three layers a modified basic algorithm is used. The general approach is as follows:

Listing 5.1: Basic Algorithm

```
1  function substitudeNetObj(layer-graph, type, config,
       function())
2  iterate over all nodes of layer-graph
3    if node type == type
4      iterate over all outgoing edges of the node
5        device1 = connected node
6        function()
7      done
8      delete node from graph
9    end
10 done
11
12 function()
13   // varies for each layer depending on the device to be
       substituted
14   find connected object n2 based on config
15   create edge between device1 and device2 with config
       prperties
16 end
```

Figure 5.9 and listing 5.1 illustrate the algorithm which is based upon a breadth-first search as all neighbours of a node are regarded. The algorithm iterates over the set of nodes of the input graph and checks it for the type to be replaced (step 1 and lines 2-3). If the node is of the type to be replaced over the edges of this node is iterated (step 1 and line 4). For each node connected to the edges the connected remote device based on the configuration (step 2-3 and lines 5-6) are relinked with the appropriate values

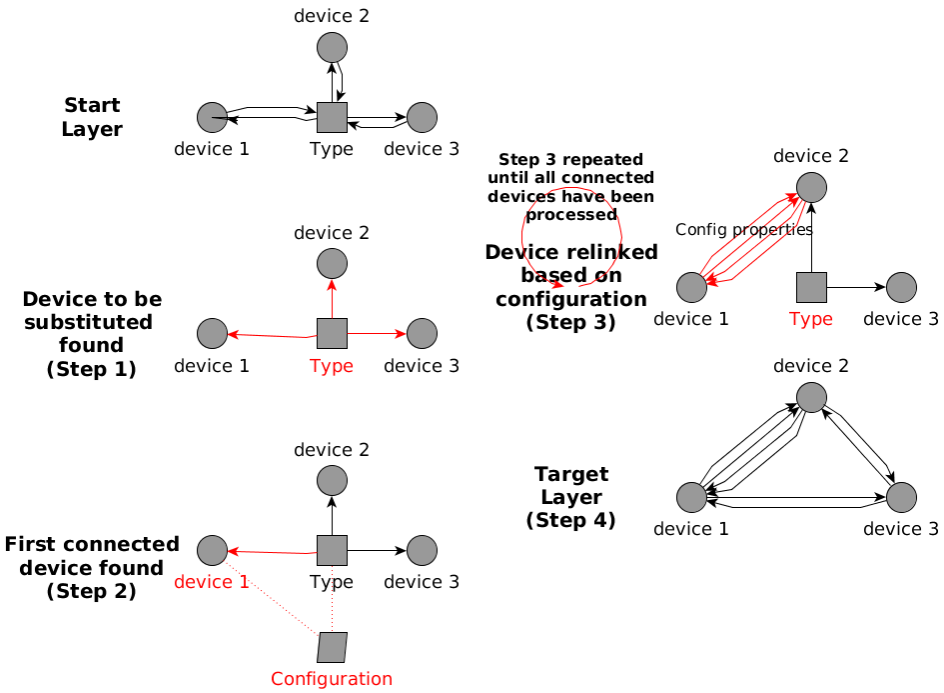

Figure 5.9: Basic Algorithm for Replacing Intermediate Network Objects

of the configuration (line 7). After all connected devices are processed the object to be substituted is deleted (step 4 and line 9) or at least transformed to a regular end device.

`function()` describes the layer specific part of the algorithm.

After line four in the algorithm it must be checked if the regarded outgoing edge is a management interface. If such an interface exists the device remains in the graph and gets a "former" prefix in its type description indicating that it does not influence the connections anymore. In this case line nine of the algorithm must be changed to a deletion of all "old" edges except the management interface. As mentioned this fact is not illustrated in the following descriptions but must be considered in the implementation.

5.4.2.2 Physical Layer Algorithm

Listing 5.2 illustrates the algorithm. The graph is built by iterating over all network objects and uniquely adding nodes for each object to the graph and their corresponding connections. The physical algorithm as basis for the model does not adapt the mentioned basic algorithm (5.4.2.1). Patch panels which only directly connect network objects without impact on the network itself must be resolved and deleted from the graph. Figure 5.10 illustrates the algorithm. By iterating over all nodes of the graph, all edges of each node of type (step 2) *patch panel* are relinked to their counterpart (step 3). The

Listing 5.2: Phyiscal Layer Algorithm

```
1 iterate over all nodes
2   if node1 not existing
3     create node1 with properties
4   if node2 not existing
5     create node2 with properties
6   create new link from node1 to node2 with properties
7   create new link from node2 to node1 with properties
8 done
```

basic algorithm is adapted and described in listing 5.3. In this case the "patch-panel-config" includes the configuration of the patch panel's internal wiring to distinguish the connections of the connected devices.

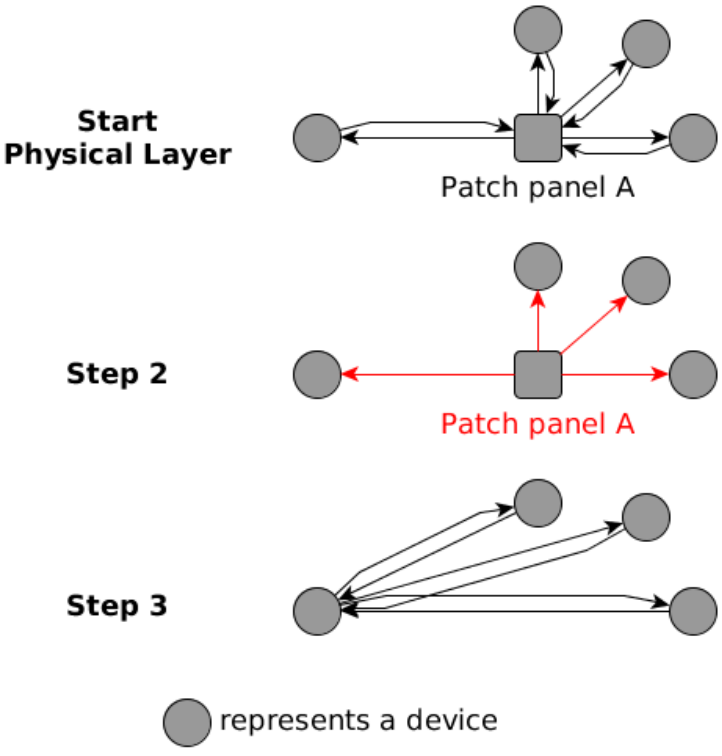

Figure 5.10: Patch Panel resolving Algorithm

Listing 5.3: Patch Panel Resolving Algorithmus

```
1 function substitudeNetObj(layer-graph="layer1", type=''
    patch panel'', config="patch-panel-config(s)" , layer1-
    function())
2
3 layer1-function()
4   find connected object n2 based on patch panel wiring
5   create edge(s) between both devices with the same
     properties
6   // patch panels do not influence the network itself
7 end
```

5.4.2.3 Virtual Machines Algorithm

The Basis for the virtualisation model is the layer one graph and the configuration of the virtualisation software (for each host). Figure 5.11 illustrates the algorithm. It is iterated over each node of the graph, searching for nodes of type *hypervisor*. For each hypervisor its virtual network configuration is mapped into the graph as follows:

- For each vSwitch a new node in the graph is created which is connected to the physical interface of the hardware switch at the same point as the corresponding hypervisor's interface (in the image these are "NIC1" and "IC2") was.
- For each VM a new node is created and connected to the corresponding vSwitch.

The vSwitch information is kept for modelling the next layer, where VLAN information is processed and the added vSwitch nodes are treated as regular switches.

5.4.2.4 Data Layer Algorithm

Basis for the VLAN model is the previous generated graph which models the actual wiring and virtual machines and the configurations of the switches in the network. As there are usually multiple switches in a network, the different switch data sets need to be stored for each switch.

Table 5.1: Switch Config Example

Interfaces	**VLAN**
0/1	1,2
0/10	3,4
0/20	1,2,3,4

A simple example for the required data looks like table 5.1. Additionally, in this example a device "A" is connected to the switch interface "0/1" and a device "B" is connected to

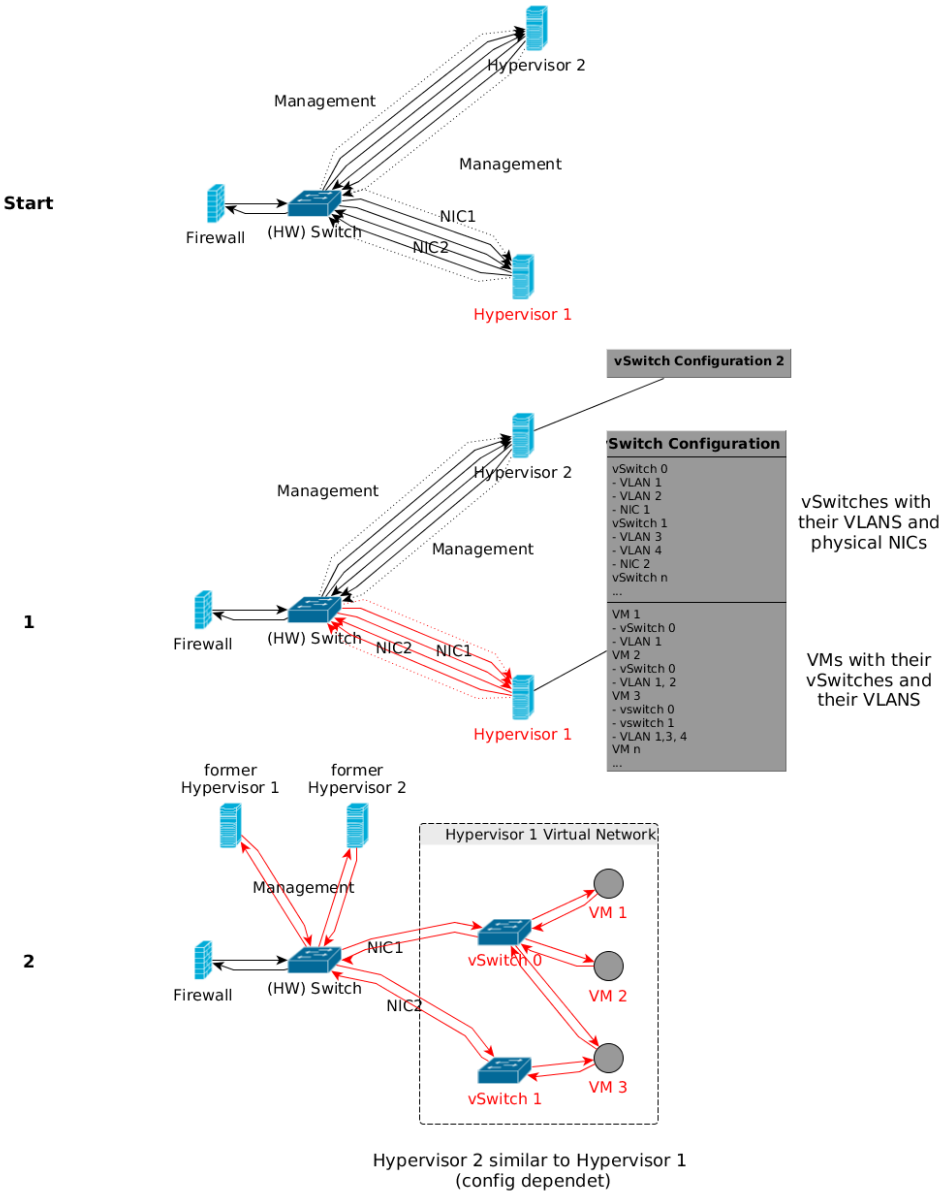

Figure 5.11: Virtual Machines Algorithm

the switch interface "0/10", a device "C" is connected to the switch interface "0/20". The algorithm will create for devices "A" and "B" each four edges connected to device "C". The edges $A \rightarrow B$ with VLAN 1 and 2 and the edges $B \rightarrow C$ with VLAN 3 and 4 (in both directions).

Figure 5.12 illustrates the algorithm, an adopted basic algorithm, which is also described in listing 5.4. By iterating over all nodes of the graph, all edges of each node of type *switch* (step 1 are relinked to their counterpart (step 3 and line 6) based on the switches

Listing 5.4: Switch Resolving Algorithm

```
1 function substitudeNetObj(layer-graph="layer1", type=''
    switch'', config="switch-config(s)", layer2-function())
2
3 layer2-function()
4   find connected object n2 based on config
5   for each VLAN id between these tow devices
6     create edge between both devices with VLAN ID
7   done
8 end
```

VLAN configuration (step 2 and lines 4-5). In this case the "switch-config" includes the configuration of the switch (see 5.1). For each VLAN id of the two regarded devices which are configured in the switch a new edge with the VLAN id property is created.

The result is a graph representing layer two in the network model.

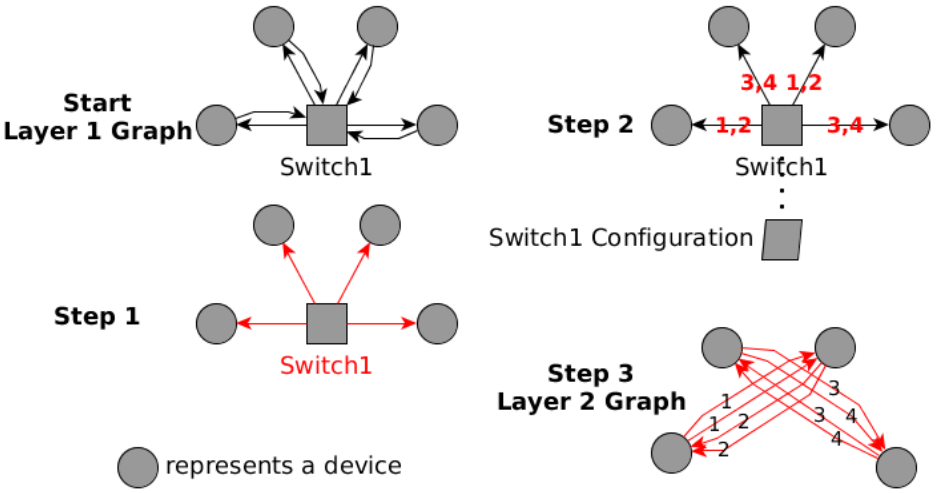

Figure 5.12: Switch Resolving Algorithm

5.4.2.5 Network Layer Algorithm

After layers one and two the next step upwards in the ISO/OSI model is inter-subnet connection on layer three (network layer). Aim is as like with switches to resolve routing tables and add edges to the graph to substitute routers. There are two kinds of routing tables, firstly the dynamic routes (the source IP is also checked) and secondly the static routes. The difference between them is that static routes are manually maintained

and the latter dynamically built by special router protocols (e.g. OSFS or RIP) which allow routers to build and exchange their routing information. At first a router would normally check the dynamic routes entries for the received package and if there is no matching entry the static routes are applied. As we will not model a path of a specific package but the possible paths of all packages all entries of the routing tables are applied as illustrated in figure 5.13. In this figure the properties of an example link are described. Therefore, there is among others, a routing table entry which routed the IP (10.20.40.100) of "Server1" to the IP of the firewall (10.20.50.10).

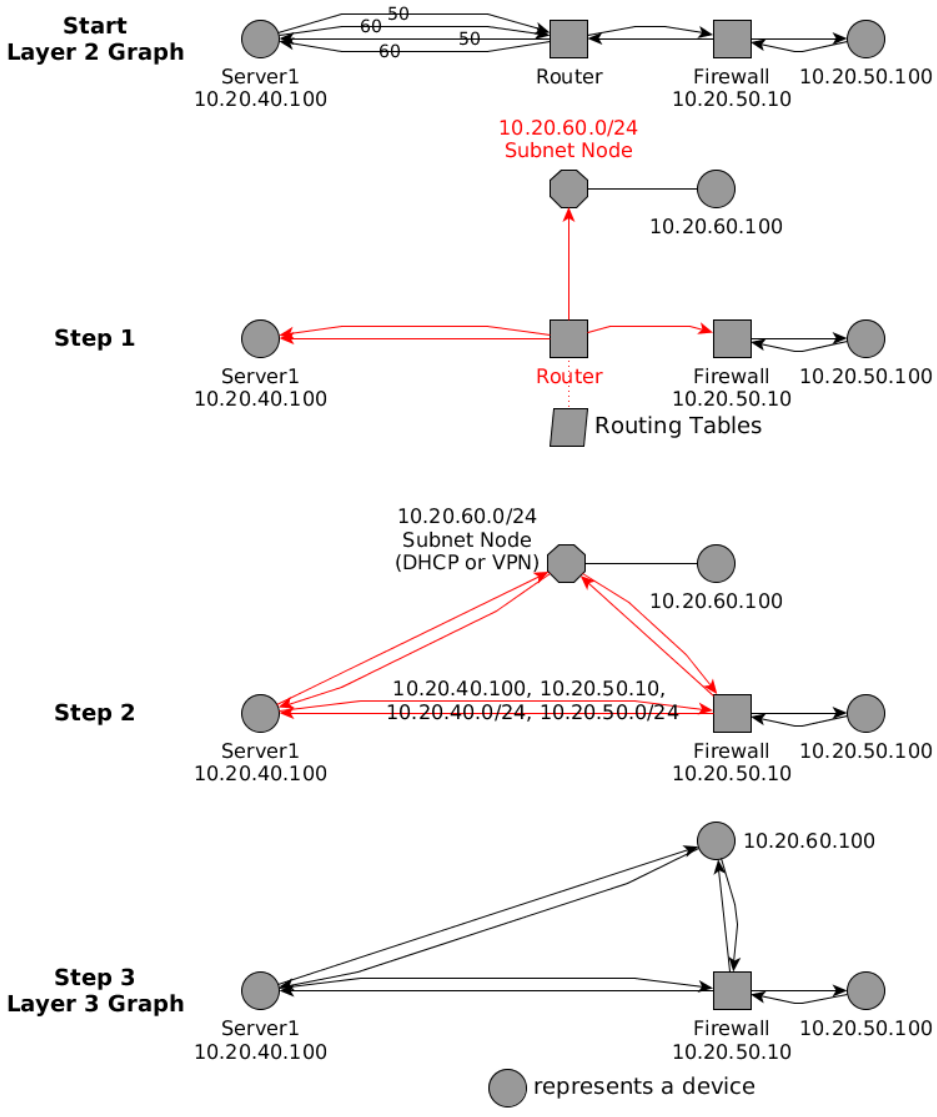

Figure 5.13: Routing resolving Algorithm

Listing 5.5 describes the algorithm which is based on the basic algorithm. By iterating

over all nodes of the graph, all edges of each node of type *router* are relinked to their counterpart (step 2 and line 11) based on the routing tables of the router (see figure 5.14 for an example how routing tables are applied). For each route which matches the regarded device a new edge with the IP properties is created. DHCP and VPN IPs are modelled with a node of type subnet (lines 5-10). At last the router is transformed to a regular end device or removed from the graph.

Listing 5.5: Router Resolving Algorithm

```
1  function substitudeNetObj(layer-graph="layer2", type=''
       router'', config="router-config(s)", layer3-function())
2
3  layer3-function()
4    for each matching routing entry for IP of device1
5      if route target is a VPN IP and VPN IP subnet node
       does not exist
6        create VPN IP subnet node
7      end
8      if route target is a DHCP IP and DHCP IP subnet node
       does not exist
9        create DHCP IP subnet node
10     end
11     create edge from device1 to target of routing table
       with IP properties (+ VPN value if applicable)
12   done
13 end
```

The assigning of routes is illustrated in 5.14 where the red link's source IP can be found in the dynamic routes whereas the green link's does not match a dynamic source IP, therefore all static routes are applied.

5.4.2.6 Firewall Algorithm

As mentioned, a minimal policy consists of a 5-tuple. To match a filter rule a packet's source and destination data must match the entries of a rule. In order to process the filter rules a data structure with the necessary information with respect to the firewalls configuration syntax is built.

As there is only a fixed source IP (the node) a new edge for every possible value according to the filter rules of the remaining three fields is created. Before the actual rules are modelled in the graph a preliminary step handling deny rules is necessary. In this step all allow rules are limited with all deny rules appearing in the rule table before. In the resulting graph the end devices are directly connected with each other with no

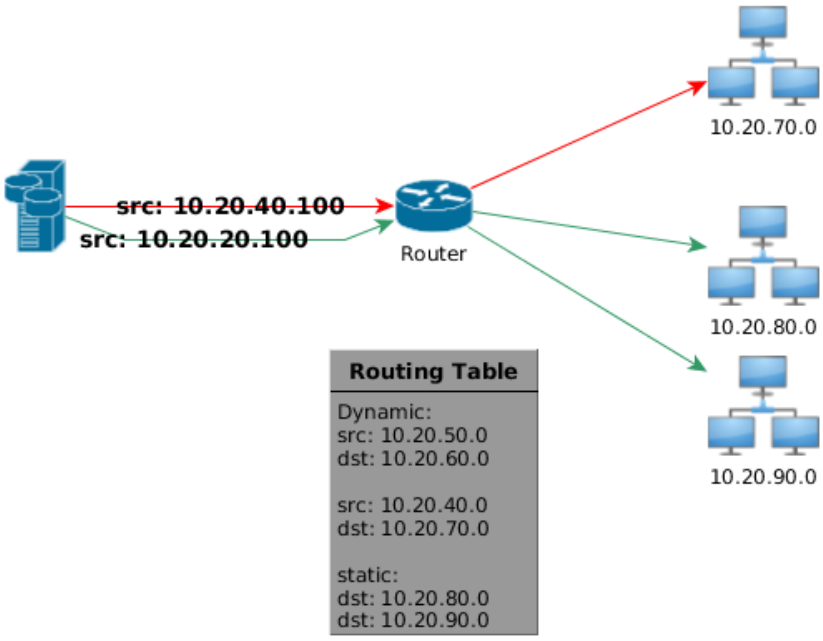

Figure 5.14: Example of getting Routes

intermediate network devices (at least with none acting as such). The algorithm is illustrated in figure 5.15 and listing 5.6. After deny rules are observed and processed in the first section of the listing the algorithm is similar to the former algorithms.

By iterating over all nodes of the graph, all edges of each node of type *firewall* (step 1) are relinked to their counterpart (step 2 and lines 15-25) based on the filter rule table of the firewall. For each filter rule matching the regarded device's IP address a new edge depending on the type (regular, ICMP, or VPN) is created. At last the firewall is transformed to a regular end device or removed from the graph.

5.4.3 Model Firewall Filter Rules (M2)

Firewall filter rules can be modelled as five tuple as mentioned in 4.2.3.6. The ordering of them must be preserved form the parser and in the stored representation. For VPN connections additionally the accepted user of the corresponding rule must be stored.

5.4.4 Model IT Policies (M3)

For verifying the implemented filter rules it is necessary to have the companies IT policies regarding network access control. As these lists are quiet similar to the actual

Listing 5.6: Firewall Algorithm

```
1  // process deny rules appearing before apply rules
2 iterate over all filter rules
3   if filter rule action = deny
4     save rule in an ordered list
5   else if filter rule action = allow
6     iterate over saved deny rules
7       eliminate deny from allow
8       save allow rule in an ordered list
9     done
10  end
11 done
12
13 function substitudeNetObj(layer-graph="layer3", type=''
     firewall'', config="firewall-config(s)", layerFW-
     function())
14
15 layerFW-function()
16   for each matching allow rule for IP of device1
17     if ICMP rule
18       create ICMP edge from device1 to destination of rule
19     else if VPN route
20       create edge from device1 to destination of rule with
     properties (protocol+port+user)
21     else
22       create edge from device1 to destination of rule with
     properties (protocol+port)
23     end
24   done
25 end
```

firewall rules they are modelled as tuple consisting of *Source IP/Port*, *Destination IP/Port*, *protocol*, and *Action*.

5.5 Read the Required Inputs

In the following, the input data is described according to 4.2.3. For each of the required inputs a parser needs to be developed which will build the data structures from the inputs.

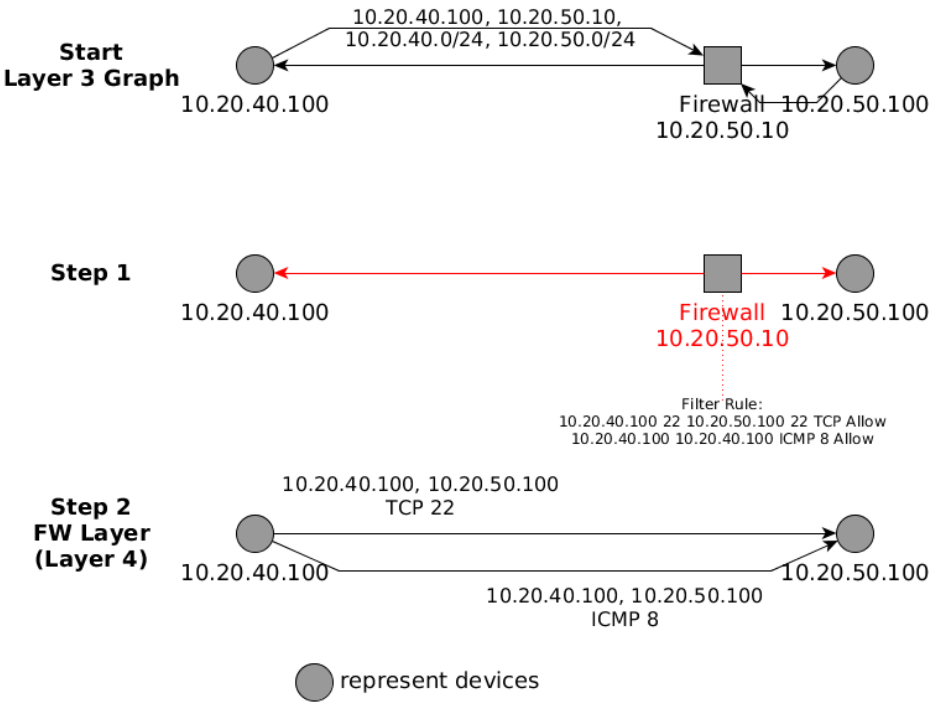

Figure 5.15: Firewall Filter Rules Algorithm

5.5.1 Physical Layer I1

For the physical layer some kind of parsable network documentation is required with the following information:

- Device names
- Device types
- Device interfaces
- Device connections

Additionally, if not available in other documentation, the list of IP addresses in use is required. A parser reads the mentioned information and stores them in a "physical-data" data structure.

5.5.2 Virtual Machines I2

Depending on the virtualisation technology a parser must read the network configuration of each host and stores the following information in a "vm-data" data structure:

- Name and VLAN configuration of each virtual switch per hypervisor
- Name and connections of each virtual machine

5.5.3 Data Layer I3

For the data layer the configuration of all switches is required. A parser must read the configuration and stores the following information in a "data-link" data structure:

- Interface (-numbers)
- VLANs on this interface
- Management interface password

Special features of a switch such as LACP (Link Aggregation Control Protocol) must be handled to be able to assign VLAN ids to the physical interfaces and not to bundled ones.

5.5.4 Network Layer I4

For the network layer the configuration of all routers must be read by a parser and the following information stored in a "network-data" data structure:

- static routing tables
- dynamic routing tables
- Management interface password

5.5.5 VPN Gateway I5

For the network layer the configuration of the VPN gateway must be read by a parser and the following information stored in a "vpn-data" data-structure:

- User(group)s
- IPs reserved for clients
- Routing tables from outgoing traffic if not acting like a bridge if clients getting local IPs

5.5.6 Firewall Layer I6

For the firewall layer the configuration of all firewalls must be read by a parser and the following information stored in a "firewall-data" data structure:

- Filter rules table
- Logging information (per filter rule and globally)
- Management interface password

5.5.7 Read IT Requirements I7

Check C6 requires the IT policies of the company which must be read by a parser and the following information stored in a "policies-data" data structure:

- srcip
- dstip
- srcport
- dstport
- protocol
- optional user
- optional service (e.g. ssh, ftp, ...)
- allowed or denied

5.5.8 Additional Input

Additionally, some information must be provided manually in a settings file:

- Paths to the firewall configuration file(s)
- Paths to the switch configuration file(s)
- The DHCP client subnet address(es)
- The VPN subnetworks
- The DMZ subnetworks

5.6 Perform Checks

This section describes the different checks to be performed (see 4.1). The checks should be implemented in modules to easily extend the architecture with additionally checks.

5.6.1 Check Preset Passwords of Network Devices (Check C1)

Baseline Protection Catalogues demands to change the preset passwords of devices (S 4.7 Change of preset passwords [7]). To verify this, the considered devices must be checked for their passwords. As passwords are usually not saved as plain text the following two ways are possible:

- Knowing the hashing/encryption algorithm
  In this case one can easily compare the value of the password with the preset password of the considered system by generating the hash with the appropriate algorithms. If the standard password is set an error is reported.
- Not knowing the algorithm
  If it is possible to reset a (testing) devices (multiple times to guarantee the same result) one can look up the preset password hash. This password hash can be compared to other devices' password hash. If it is the same hash, chances are high that there is still the preset password in use.

One must keep in mind that for both cases encrypting and hashing may be based on some (unknown) encryption key or hashing salt. Depending on how this is implemented the resulting value may be only valid for the machine at a certain state. This safeguard is applicable to all (managed) devices which provides access to some kind of management interface (e.g. switch, firewalls,..).

5.6.2 Check if Firewall Logging is Enabled (Check C2)

According to the BSI, firewalls should write logs in general and also policy specific log information (S 2.78 Secure operation of a firewall [7]). The configuration of the considered firewalls must be checked for enabled logging for both, general and policy specific logs. As the firewall parser provides the logging configuration this check only evaluates if the logging value of each policy is enabled and additionally if the global logging is activated. In case of a lack of logging a report is generated showing where the logging is missing.

5.6.3 Find Connections Bypassing a Firewall (Check C3)

With the complete network topology and all relevant devices are included in the graph (see 5.6.4.1) this algorithm checks if devices are able to communicate bypassing the firewalls (figure 5.16 illustrates the algorithm). It is iterated over all nodes of the underlying graph model. For each node the possible outgoing traffic paths are evaluated (step 1). If a single connection is possible to another device not being of the type firewall this

would break the safeguards recommendation (step 2). In this case the corresponding devices and their connection is saved for an error report

As this method is layer independent all modelled layers ranging from layer one to three can be the input of these checks. The revealed noncompliance would be as follows:

- Physical Layer
  Devices directly connected through an Ethernet cable without firewall
- Data Layer
  Devices connected on the same broadcast domain without a firewall
- Network Layer
  Devices of different subnetworks connected without a firewall

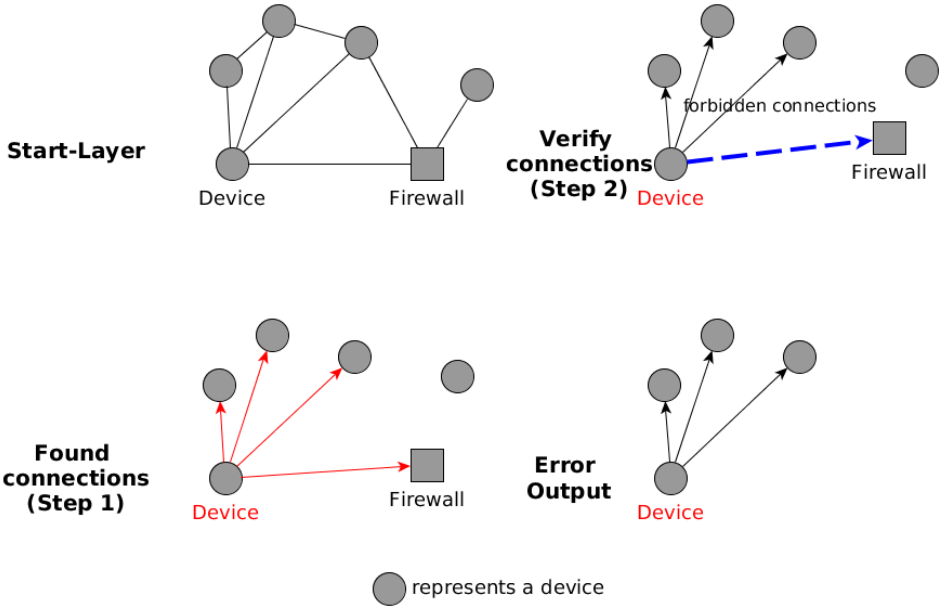

Figure 5.16: Bypassing Firewalls Algorithm

5.6.4 Check the Existence of Filter Rules for All Devices (Check C4)

This statement is part of the safeguard S1. For this thesis, computers are defined as machines in a network consuming or producing data traffic, including e.g. virtual machines and IP outlets. Furthermore, machines are not regarded as one object but with a sophisticated view on their NICs. In an enterprise environment it is not uncommon that devices have with multiple interfaces and IP addresses. To check this statement three kinds of information is needed:

1. The network topology (see section 5.4)

2. IPv4 addresses in use, named IP list
3. Firewall rule table (compare 4.2.3.6)

5.6.4.1 Include all Computers

Two check if the graph includes all relevant network objects two checks are needed:

1. Verify that all IP addresses of the IP list are included in the graph (listing 5.7)
2. Verify that all IP addresses of the graph's nodes are in the IP list (listing 5.8)

With those checks errors in the graph respectively in the IP list are revealed. This is important as safeguard S1 demands that all internal computers must be considered.

Listing 5.7: Verify IPs of the Graph are defined

```
1 iterate over all nodes
2   iterate over node's IP addresses
3     if IP is not in IP list
4       report error
5     end
6   done
7 done
```

Listing 5.8: Verify IPs in IP list are in the graph

```
1 iterate over IPs from IP list
2     if IP not in the graph
3       report error
4     end
5 done
```

5.6.4.2 Verify Existence of Firewall Filter Rules

After it is certain to consider all internal computers in the model, the next step is to verify that all of them are matched by at least one "allow" filter rule. For this check it is convenient to compare the source IP field of the filter rule with the IPs of the network devices. For each filter rule matching this way, the according action is saved. The action of the saved rules is checked as there should be no deny rule (whitelisting).

5.6.5 Check the Whitelist Approach in Filter Rules (Check C5)

Whitelist approach is mentioned several times (S1 and S2) and can be verified by checking each policy's *action* value of all firewalls. It should be set to *allow* (or the corre-

Listing 5.9: Verify Existence of Firewall Filter Rules

```
1  iterate over all nodes
2    iterate over all IPs from node
3      if filter rule table contains matching src ip field
4        if filter rule action == allow
5          rule found
6        end
7        if filter rule action == deny
8          report error in whitelist approach
9        end
10     else // in case there is no deny all rule
11       report error in missing filter rule
12     end
13   done
14 done
```

sponding value depending on the firewall vendor) otherwise there is a deny rule which is not compliant to the whitelist approach. If there is a *deny* action, the corresponding rule is saved and written to the result as warning of this check.

5.6.6 Find Possible (TCP/UDP) Data Paths and Compare them to the IT Requirements (Check C6)

For this check the firewall layer and the IT policies are required. Aim is to compare the enforced data traffic rules by the firewall rule tables to the requirements of the IT policies. The algorithm is illustrated in listing 5.10. Two steps are required to check whether one set of rules is a strict subset (illustrated in figure 5.17) which would be an error as both sets must be equivalent.

5.6.7 Check the Handling of Different ICMP Types (Check C7)

Safeguard S 5.120 ([7]) contains recommendations for the handling of ICMP messages in a network. It distinguishes between internal computers and public servers in a DMZ.

5.6.7.1 Internal Network

To check the internal ICMP handling the following information is required:

- Firewall layer graph
- The recommendations from the safeguard (which will be hard coded)

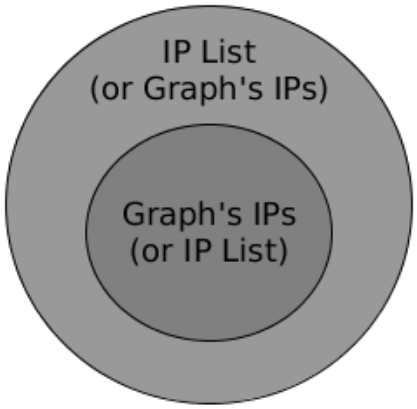

Figure 5.17: Possible Errors in Comparing Subsets

Listing 5.10: Find Data Paths and Compare them to IT Requirements

```
iterate over nodes of firewall layer
  iterate over outgoing edges of node
    get values of edge (depending on edge type):
      <- srcip
      <- dstip
      <- srcport
      <- dstport
      <- protocol
      <- icmp
      <- users
    save values in <values>
    if edge values are not matching IT any policy entry
      report error // Filter rule is missing in IT policy
    end
  done
done

iterate over IT policies
  if policy is not in values
    report error // IT policy is missing in filter rules
  end
done
```

By iterating over the graph, each node's outgoing and incoming edges are checked if their protocol is of the type ICMP and their connected object is an internal computer. With respect to the edges' direction, the ICMP type is compared to the allowed one's in the safeguard. They must match exactly for both directions as only allowed traffic

Table 5.2: Internal Computer ICMP Handling according to BSI Baseline Protection

ICMP	Incoming	Outgoing
Type 8	deny	allow
Type 0	allow	deny
Type 3	allow	allow
Type 11	allow	allow

Table 5.3: Public Server in a DMZ - ICMP Handling according to BSI Baseline Protection

ICMP	Incoming	Outgoing
Type 8	allow	allow
Type 0	allow	deny
Type 3	allow	allow
Type 4	allow	deny
Type 11	allow	allow

is modelled and the rest is denied. Table 5.2 shows the recommendations. ICMP types which should be blocked for both directions are not shown (as mentioned only allowed types must exactly match). In case of noncompliance a report is generated.

5.6.7.2 Public Server in a DMZ

To check the public server (DMZ) ICMP handling the following information is required:

- Firewall layer graph
- DMZ Subnet information
- The recommendations from the safeguard (which will be hard coded)

By iterating over the graph each node which has at least one IP of the specified DMZ subnet(s) are regarded. All ICMP edges of that nodes where the connected node is not part of the internal network (e.g. node Internet) are compared to the recommendations (for both directions) with the same methodology as internal computers (regarding the number of allow edges). In case of noncompliance a report is generated.

Table 5.3 shows the recommendations where ICMP types which should be blocked for both directions are not shown (as mentioned only allowed types must exactly match).

Figure 5.18 illustrates this based on a simple example. For full compliance each pair of connected nodes must have the *exact* amount of illustrated edges as they represent the allowed ICMP messages. Each ICMP type which is not modelled as edge with the properties of the ICMP type in the graph is blocked.

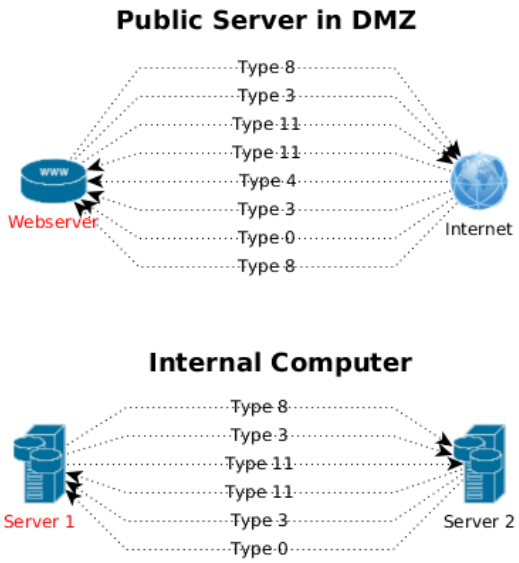

Figure 5.18: ICMP Handling Illustration

Chapter 6

Implementation

The developed software is a pure Java application with certain libraries included as follows:

- JDBC MySQL connector [21]
  The standard Java MySQL interface for querying MySQL databases for querying network information from the documentation. This is required to read the physical network information (see 5.5.1).
- JUNG Graph library [22]
  A Java graph library (see 6.1) for building the network topology which implements the section 5.4.1.3.
- Json library [23]
  A Java JSON parser to handle the tool's settings file in JSON format to implement the extensible approach of section 5.3.

It was coded with Oracle JDK 1.8 (but with no features of this recent version used) and Eclipse Luna 4.4.0.

Table 6.1 shows the implemented requirements. A "X" indicates that this requirement is fully implemented, an empty cell indicates that this requirement is missing in the implementation.

6.1 JUNG

This section describes the usage of the Java library JUNG [22] to model a (computer) network. Some concepts of INDL/NML are slightly adopted. The basic idea behind JUNG is the possibility to create own (Java) classes for nodes and edges and their methods with full featured Java inheritance due to JUNG's generic design. That is, a node ”BasicNode” can provide common attributes and methods and some specialized nodes ”NodeA” and

Table 6.1: Implementation Status

Requirement	Implementation Status
D1	Layer 1 implemented
D2	Implemented
D3	
D4	Implemented except I2
R1	Implemented
C1	Implemented
C2	Implemented
C3	Implemented
C4	
C5	Implemented
C6	
C7	

”NodeB” may implement new data but also inherit from ”BasicNode”. The same will apply to edges.

The advantages of JUNG are as follows:

- Extremely customizable for our needs by allowing to define own node and edge classes with needed properties
- Built-in graph algorithms and methods
- filtering options e.g. weighted edges
- Several different types of graphs

The network model is implemented with JUNG and own classes for vertices and edges which inherit from each other. Figure 6.1 illustrated the inheritance model of the classes. For each layer of the network model a specific edge class exists. Layer three and firewall layer edges have several constructors according to the possible different values of an edge (see 5.4.1.8) respectively the isVPN property. Figure 6.2 illustrates the inheritance of vertices. For layers one to three a vertex is modelled. Vertices of the firewall layer have the same attributes as `Layer3Vertex`.

6.2 Package Overview

The modularity approach is pursued by separating the different functionalities in several packages and classes. The following list shows the packages structure of the software tool.

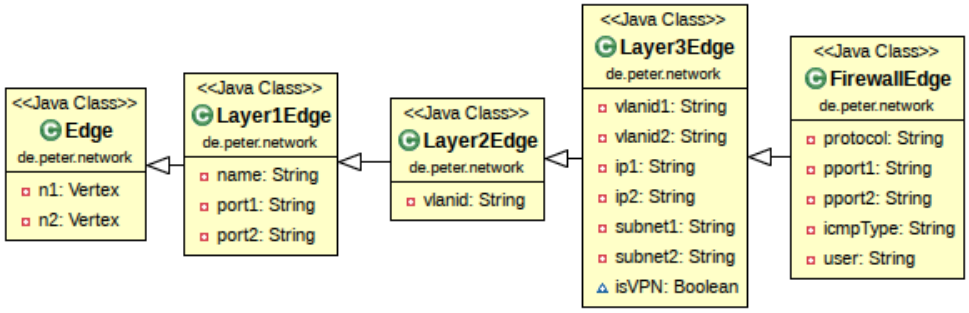

Figure 6.1: Edge Inheritance Model

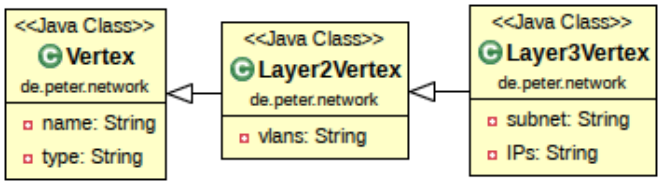

Figure 6.2: Vertex Inheritance Model

- de.tum.peter.main
  Main entry point for the software
- de.tum.peter.racktables
  Package for parsing and storing network information
- de.tum.peter.fw
  Provides classes for parsing and storing firewall configuration information
- de.tum.peter.sw
  Defines classes for parsing and storing switch configuration data
- de.tum.peter.network
  Classes for building the network model with JUNG
- de.tum.peter.reqs
  Parsing of IT policies
- de.tum.peter.check
  Includes all classes to check the selected BSI safeguards implementation status
- de.tum.peter. auxiliary
  Includes the settings parser and an auxiliary class

6.3 Build Model

6.3.1 Build a Representation of the Network Topology (D1)

This section describes the implementation of the network topology model.

6.3.1.1 Physical Layer

The first graph building function which simply builds a `DirectedSparseMultigraph` (allowing directed and parallel edges) from all physical Ethernet connections.. `Layer1Edges` (for each direction) and `Vertex` objects are instantiated. Both vertices are added to the graph with JUNG's method `addVertex(V vertex)` as well as both links are added to the graph with the provided `addEdge(E e, V v1, V v2, EdgeType edgeType)` and `EdgeType.DIRECTED`

Based on the former graph `layer1a` all *PatchPanel*s are going to be substituted. As one cannot modify a graph on which is being iterated, at first a clone of `Layer1a` is created (JUNG does not implement a clone method itself). At the end of this method the new connections are added to the graph (checking that no new nodes are being created) and all *PatchPanel*s are removed (automatically removes the edges).

6.3.2 Model Firewall Filter Rules (D2)

The firewall filter rules are part of the `FirewallObject` (see 6.4.3).

6.4 Read the Required Inputs (D4)

6.4.1 Physical Parser I1

`PhysicalParser` provides a parser to get the necessary information needed to build the physical network topology. With these information a `PhysicalObject` is built consisting of `PhysicalConnectionObjects` (illustrated in 6.3). One `PhysicalConnectionObject` consists of the following information:

- the names of both devices linked to each other
- the types of both devices
- the port numbers of both devices

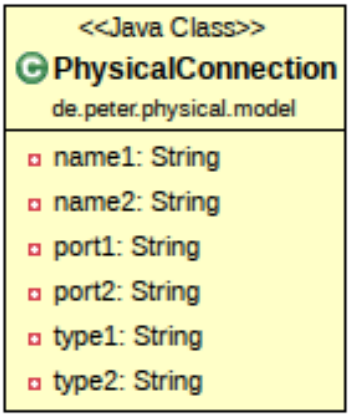

Figure 6.3: Physical Connection

6.4.2 Switch Parser I3

`SwitchParser` provides a parser which reads the config file line by line and fills the necessary information of a `SwitchObject` With information for each switch (respectively each switches configuration file) a `SwitchObject` is build and managed as figure 6.4 illustrates. For each switch (-config) one `SwitchObject` is built with the necessary

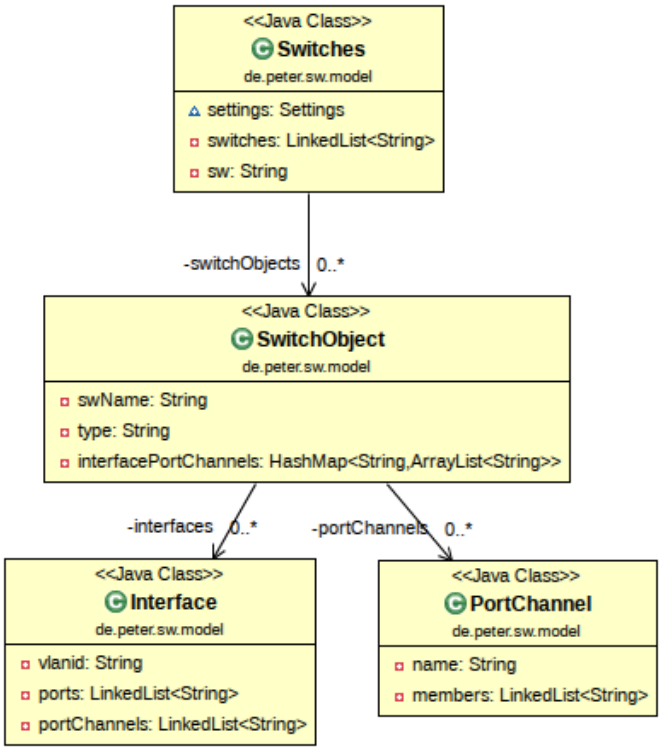

Figure 6.4: Switch Model

information to map its interfaces to its specific VLAN id(s). Each `SwitchObject` is saved in a `Switches` object as a value of a `LinkedList` to provide a data structure containing all switches.

6.4.3 Firewall Parser I4 I5 I6

The firewall parser reads router, VPN gateway and firewall information. `FirewallParser` provides a parser which reads the config file line by line and generates temporary files containing the needed sections of the firewall configuration. This simplifies parsing because it reduces the complexity of the data to be parsed as the files consist only of one section each. After that the generated files are read in and filtered for the needed information with Java regular expression Patterns and the temporary file is deleted. The result is a `FirewallObject`.

The parser also reads the line `set status` of the sections `config log syslogd` and `log memory`. According to this line a boolean variable in `FirewallObject` is set to indicate whether global logging is activated. Furthermore, the parser reads each filter rule's logging status by reading `set logtraffic` and `set log-unmatched-traffic` in the `config firewall policy` section of the config. Accordingly, the parser sets each policies' boolean value in the `FirewallObject`

In addition the parser reads the password information of the configuration file and stores it in the `FirewallObject`.

The required information form the firewall(s) configuration file is parsed into one `FirewallObject` for each firewall as shown in figure 6.5. Such an object itself consists of certain objects representing the different kinds of inputs sections described in 4.2.3.6. The object `Firewalls` contains all `FirewallObjects` in a `LinkedList`.

The firewall parser also parses the routing information and the VPN gateway settings.

6.5 General Requiremetn (G1)

This software was designed to be extensible respectively extended with other inputs (e.g. firewall or switch models). Therefore, one must just provide new parsers for each input type to fill the corresponding model objects (`PhysicalObject` , `SwitchObject`, `FirewallObject`).

In `Settings.json` different paths for the needed input files which will be parsed by `Settings.java` and returned to the classes needing those input files are configured. `Settings.json` contains the following settings:

- The paths to the firewall(s) configuration files

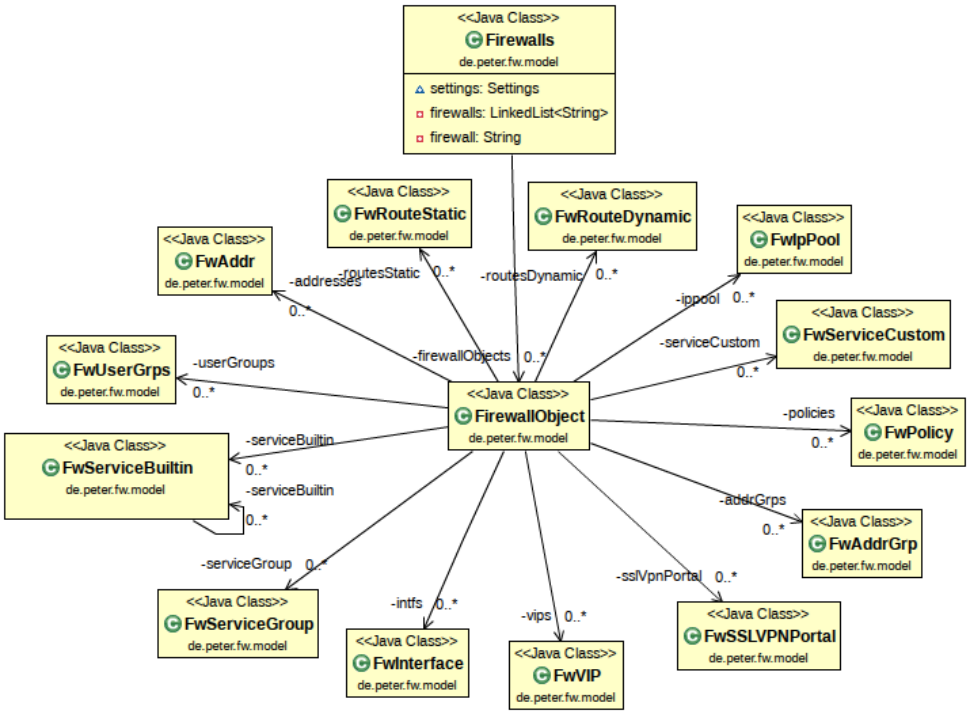

Figure 6.5: FirewallObject

- The paths to the switch(es) configuration files
- The (DHCP) client subnetworks
- The VPN subnetworks
- The DMZ subnetworks

6.6 Perform Checks

In this section the implemented checks are described.

6.6.1 Check Preset Passwords of Network Devices (Check C1)

Required input: `FirewallObject`

This check iterates over each `Firewallobject` (representing a firewall configuration) and checks the value of the `userPWs`. In case of an error a warning is printed.

6.6.2 Check if Firewall Logging is Enabled (Check C2)

Required input: `FirewallObject`

This check iterates over each `FirewallObject` (representing a firewall configuration) and checks the value of the global logging. Additionally, the check iterates over all filter rules of each `FirewallObject` and checks both logging values.

6.6.3 Find Connections Bypassing a Firewall (Check C3)

Required input: layer one graph

This check evaluates paths between devices. The aim is to find connections between devices without a firewall between them. The check iterates over the graph and checks for each found node its outgoing edges with `layer.getOpposite(node, edge)`. If the opposite of an node is not a firewall a warning is printed.

6.6.4 Check the Whitelist Approach in Filter Rules (Check C5)

Required input: `FirewallObject`

This check iterates over each filter rule of the `FirewallObject` and checks the `getAction()` for not allowed values (e.g. “deny”). In case of an error a warning is printed.

Chapter 7

Use Case at BörseGo AG

As being BörseGo AG, a German company located in Munich offering financial charts and analysis platforms, partner for this thesis, the inputs needed are based on the systems in action at the company which are as follows:

1. Racktables [24] (network documentation tool)
2. Fortinet Fortigate 620B and FGT80C (firewalls, routers and VPN gateways)
3. Force10 S25-01-GE-24T, 3Com 4500G 48-port and 3Com 5500G-EI 48-port (switches)

7.1 Network Topology Data Gathering

This section describes the different inputs to model the network.

7.1.1 Racktables

At the company, Racktables ([24]) is (only) used to document the network wiring. Racktables allows to define locations, racks and objects and assigning links to them through a web interface. Figure 7.1 illustrates some connections of a switch. One can see on the left side the interfaces provided by the switch, their labels and types and on the right side the connected remote object and its port. There is also the possibility to manage the IP space in Racktables what is done and also parsed. Figure 7.2 illustrates a patch panel and the relationship between *Portx* and *Cablex*. Racktables stores the data in a MySQL database. Three major sets of information are queried:

- all objects (servers, switches, routers,...)
- all links between those objects represented by the set of edges in a graph
- the complete IPv4 address list in use

ports and links

Local name	Visible label	Interface	L2 address	Remote object and port	
mgmt ...	Management 0/0	1000Base-T		dc1-switch01-02	2/28
serial mgmt	MGMT_uplink	RS-232 (RJ-45)		RS-232 Switch	Port15
strom1		AC-in		ipps01	Port6
strom2		AC-in		ipps02	Port6
0/0 ...	dcesx01_nic1	10GBase-T		dcesx01	10G-1
0/1 ...	dcesx02_nic1	10GBase-T		dcesx02	10G-1

Figure 7.1: Extract of the connections of a switch

Common name: Patchpanel MUC J
Object type: PatchPanel

ports and links

Local name	Visible label	Interface	L2 address	Remote object and port		Cable ID
Cable23		1000Base-T		Patchpanel MUC Hinten	Cable19	
Cable24		1000Base-T		Patchpanel MUC Hinten	Cable20	
Port23	uplink_muc-switch01	1000Base-T		muc-switch01-00	0/19	
Port24	uplink_muc-switch01	1000Base-T		muc-switch01-01	1/19	

Figure 7.2: PatchPanel Object in Racktables

7.1.2 Switch

There are several switches in use but their configuration can be broken down to the two manufacturers: *force10* and *3com*.
Aim is to gather VLAN configuration of the network and therefore parse the interfaces of the switches and getting the VLAN ids of the connections. An example for a force10 configuration entry is illustrated in 7.1

Listing 7.1: force10 Switch Snippet

```
1 interface Vlan 50
2  description private_access
3  no ip address
4  tagged Port-channel 3-4,10-21,31-34
5  untagged GigabitEthernet 2/19-20
6  untagged Port-channel 35-38
7  no shutdown
```

interface Vlan 50 is describing VLAN id 50 and *(un)tagged (Ten)GigabitEthernet* the interfaces of that VLAN additionally to the *Port-channel*s which aggregate several interfaces. An example of an entry for Port-Channels is illustrated in 7.2.

Listing 7.2: force10 Portchannel Snippet

```
1 interface Port-channel 10
2  description esx01
3  no ip address
4  switchport
5  channel-member GigabitEthernet 1/19
6  channel-member GigabitEthernet 3/19
7  sflow enable
8  no shutdown
```

According to this, the *channel-member*s of that Port-Channel are in VLAN 50, too. (All Port-Channels have to be resolved to their member interfaces to get exact information about which interfaces are part of a VLAN id).

The configuration file for 3com switches is simpler, e.g one relevant block looks like the listing 7.3 illustrates. For each interface (here 1/0/39) there is an entry *port access VLAN 2020* telling the VLAN id (here 2020).

Listing 7.3: 3com Switch Snippet

```
1 interface GigabitEthernet1/0/39
2  port access VLAN 2020
3  broadcast-suppression pps 3000
4  undo jumboframe enable
5  stp disable
6  stp edged-port enable
7  undo ndp enable
8  undo ntdp enable
```

`SwitchParser` provides a parser for reading switch (VLAN) data. Due to the fact that switches of two manufacturers are in action, the parser must distinguish between those with a simple `switch statement` For force10 switches the parser needs to read *Port-Channel* information (bundled physical interfaces) to resolve the VLAN to actual port numbers. 3com config files can be parsed straight forward with simple entry matching/parsing with one minor exception. There are two possible occurrences meaning the same thing. *access VLAN* and *hybrid VLAN*. That is just an internal convention when modifying one entry through the web interface. Getting the actual content is implemented by regular expressions.

7.1.3 Fortigate Firewall

In the company Fortigate firewalls act as firewall, router and VPN gateway. Therefore, the parser must gather router information, VPN information, and the filter rules.

7.1.3.1 Password

In case of Fortigate firewalls the default password is the empty (none) password. In order to check this, a testing device was reset and the Log-in was successful without a password. To automate this process the configuration file needs to be checked for the line *password ENC* in the *config system admin* section. This line is not present if the standard password is unchanged. As the mechanism how the firewall stores the password is not public no further statements cannot be made.

7.1.4 Firewall Functionality

The firewalls filter rules are described in the following. (Appendix A.2 gives an illustrated overview).
Filter Rules (FortiOS v4.00 M3 Patch 18) are structured as follows:

- Interfaces
  Interfaces distinguishes between ”srcintf” and ”dstintf”. The interface of the policy is translated in the interface section of the firewall. An example of an interface is:

```
1       edit "internal1"
2         set vdom "root"
3         set ip 10.0.0.1 255.255.255.0
4         set allowaccess ping https ssh
5         set type physical
6         set description "Admin Port - Access to all
      networks"
7         set alias "admin port"
8       next
9
```

- Addresses
  Addresses also distinguishes between ”srcaddr” and ”dstaddr” and is translated from either the ”addresses”-, ”AdrGroups”- or ”Vips”-section of the config. An example for an address is:

```
1       edit "infra04"
2         set subnet 10.20.30.22 255.255.255.255
3       next
```

4

- Services
  "Services" is the generic term for either a built-in or custom service or a group of services determining the protocol or ICMP and its type. The built-in services cannot be found in the configuration file but are listed in the web interface. An example of a service is

```
1         edit "8080"
2           set protocol TCP/UDP/SCTP
3           set tcp-portrange 8080
4         next
5
```

  describing a TCP service at port 8080 with the name ”8080”.

- IPPool
  Used to change the destination IP of an IP package (e.g. internal IP replaced with external IP) and the destination IP remains (also called source NAT).

- config firewall vip
  Used to change the destination IP of an IP package (e.g external IP replaced with internal IP) and the source IP remains (also called destination NAT).

- config firewall addrgrp
  Groups of addresses that will be used in the filter rules

- config user group
  User groups

- Action
  What is done with the packet (”accept” or ”ssl-vpn” for identity based rules). This entry would be missing in case of an deny rule as the standard action is deny.

- IdentBased
  This (optional) setting is set when the policy applies to user based rules through a VPN connection. Rules with this settings are further split up into ”user group” (the groups allowed), ”vpn ssl web portal” and ”widget” determining the allowed services/applications.

For Fortigate Firewalls, whitelisting is the default approach and implemented by an implicit deny all rule which will be applied to packages not matching any former rules [25]. This behaviour is not customizable by the user. However if some administrator want to enforce a blacklisting strategy the rule table would contain the deny rules at first ending with allow all rules so that packages at least will be matched by them and

the firewalls implicit deny rule would be never applied. Fortigate firewalls enforce a whitelisting per default settings.

7.1.4.1 Router Functionality

Fortigate Firewalls also act as router in the network. This information can be found in the following two sections of the configuration file:

- *config router static*
  Contains the static routing table. An example entry is illustrated in listing 7.4. Here packages arriving at the interface "officebgo-dcbgo" are routed to the destination, independent of their source ip.

Listing 7.4: (Static Routes Example)

```
1 edit 16
2   set device "officebgo-dcbgo"
3   set dst 10.20.35.64 255.255.255.192
4 next
5
```

- *config router policy*
  Contains the dynamic routing table. An example entry is illustrated in listing 7.5. Packages arriving at the interface "jfd_access" and only if they are matching the source ip ("scr") are routed to the destination ip ("dst") on interface "officebgo-dcbgo".

Listing 7.5: (Dynamic Routes Example)

```
1 edit 2
2   set input-device "jfd_access"
3   set src 10.20.80.0 255.255.255.0
4   set dst 192.168.1.0 255.255.255.0
5   set output-device "officebgo-dcbgo"
6 next
7
```

7.1.4.2 VPN Functionality

The parser reads the firewalls VPN information which is stored in `config user group` for groups (based on LDAP groups) and `config vpn ssl web portal` for allowing configured services.

7.1.4.3 Firewall and policy logging

According to BSI firewalls should write logs in general and also policy specific log information (S 2.78 Secure operation of a firewall [7]). For the first one, Fortigate has two options which should be enabled.

1.

```
config log memory setting
  set status enable
end
```

2.

```
config log syslogd setting
  set status enable
end
```

Both will be checked automatically by parsing those lines and check them for existence and for *enable*. The second point can be checked by parsing *set logtraffic* for logging allowed traffic and *set log-unmatched-traffic* for dropped (denied) traffic per policy entry.

7.2 Network Topology Building

The following sections describe how to model step by step an (IP) network with JUNG graph library.

7.2.1 Layer 1a

The physical layer is pretty straight forward. By iterating over all connections from Racktables the necessary information is provided. A hashmap of vertices guarantees that nodes are created uniquely. Two edge objects for each connection (as being a directed graph) are created and added to the graph. The resulting graph is illustrated in 7.3

7.2.2 Layer 1b

Layer 1b is also related to the physical layer and the difference to the former graph

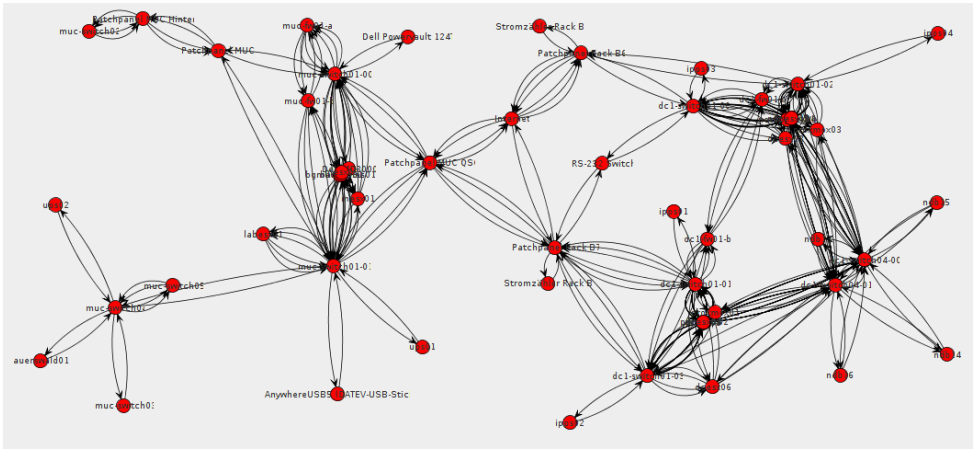

Figure 7.3: Layer 1 Graph of BörseGo AG

is that all patch panels are removed as they are transparent for the above layer. To accomplish that it must be iterated over a clone of the former graphs vertices being checking for nodes of type *PatchPanel*. For each node found that way all edges of that node must be replaced by a direct connection between the two network devices the panel elongates. Obviously the last step is to remove all *PatchPanel* nodes from the graph. Figure 7.4 shows the graph without patch panels.

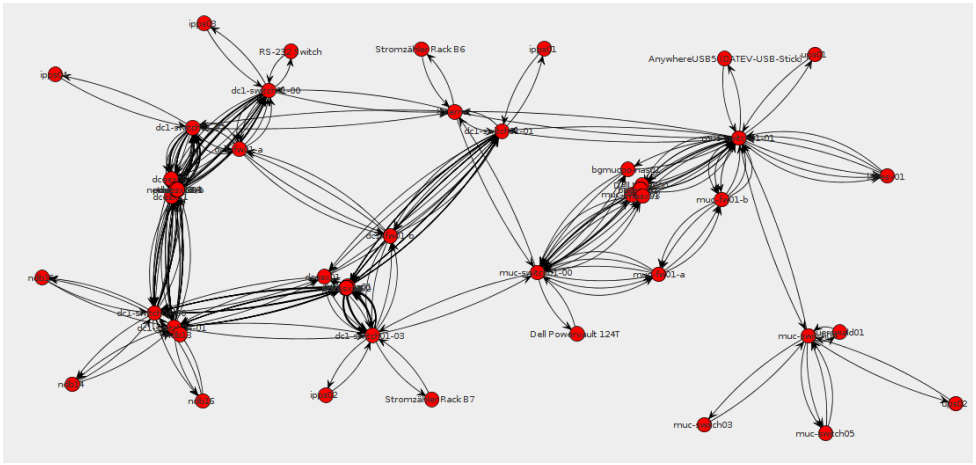

Figure 7.4: Layer 1 Graph without Patch Panels

Chapter 8

Evaluation and Results

In this chapter the results of the thesis are described.

8.1 Research Questions

In section 1.1 four research questions were introduced and were answered in the course of this work:

Q1 Find suitable safeguards to be automated
In section 4.1.1 several safeguards are described which offer a benefit in automating their evaluation.

Q2 Build a model of the required data for safeguard checks
The model to evaluate the implementation recommendations the following requirements need to be fulfiled:

- M1 A network topology model should be developed (see section 4.2.2.1)
- M2 A model for the filter rules of firewalls (see section 4.2.2.2)
- M3 A model for IT policies (see section 4.2.2.3

Q3 Specify needed information for verifying selected safeguards
To build the network topology model different inputs are required as follows:

- I1 Network Cables (see section 4.2.3.1)
- I2 Virtual Machines (see section 4.2.3.2)
- I3 Switch configuration (see section 4.2.3.3)
- I4 Router configuration (see section 4.2.3.4)
- I5 VPN configuration (see 4.2.3.5)

I6 Firewall configuration (see 4.2.3.6)

I7 IT Requirements (see 4.2.3.7)

Q4 Define checks to (partially) cover selected safeguards
Section 4.2.4 illustrates the checks that are required in order to verify the implementation status of the selected safeguards. In section 5.6 algorithms to for each check to evaluate the implementation of the safeguard's recommendations are described.

Table

Table 8.1: Requirements Fulfilled

Requirement	**Status**	**fulfiled with**
G1	fulfiled	5.3
M1	fulfiled	5.4.1
M2	fulfiled	5.4.3
M3	fulfiled	5.4.4
I1	fulfiled	5.5.1
I2	fulfiled	5.5.2
I3	fulfiled	5.5.3
I4	fulfiled	5.5.4
I5	fulfiled	5.5.5
I6	fulfiled	5.5.6
I7	fulfiled	5.5.7

In the following sections the results of this thesis are described.

8.2 BSI Safeguards

In this thesis Baseline Protection safeguards were analyzed with respect to their possibility of automating their the implementation status of the recommendations. The evaluation of the recommendations of the selected safeguards would be error-prone and complex (see 4.1.1). This work describes an approach of automatically evaluating the status of selected safeguards (see 5.6) which differs from the available support tools (see 3.1).

This thesis shows a method of automatically verifying the main recommendations of six safeguards which are complex to be verified manually. The benefits of an automatically approach of these safeguards in contrast to the available tools are as follows:

1. An automatic approach is less error-prone
2. Saves time and costs

3. Supports the preparation of an audit
4. Regularly checks are possible
5. Represent the actual network configuration

The following list describes the degree of automation of the implementation status of each selected safeguard:

- Selection and Implementation of Suitable Filter Rules (see 4.1.2.1)
  The following recommendations of this safeguard could be automated:
  - Whitelist approach with C5 "Whitelist"
  - User specific traffic setting with C6 "Policy"
  - Considering all internal computers with C4 "filter rules"

  The following recommendations (see [7]) could not be automated:
  - The reason and the initiator of the filter rules must be documented
  - Rules at the application level gateway must be treated similar to the firewall filter rules
- Secure Operation of a Firewall (see 4.1.2.2)
  The following recommendations of this safeguard could be automated:
  - Whitelist approach with C5 "Whitelist"
  - find devices that bypass the firewall with C3 "Bypassing"
  - No preset passwords in security gateways with C1 "Passwords"

  The following recommendations (see [7]) could not be automated:
  - Application level gateway related
  - Device configuration which are part of the security gateway
  - Secure software of the components of the security gateway
  - Integrity tests of the software of the security gateway
  - Crashing, backup and restoring of components of the security gateway
- Change of Preset Passwords (see 4.1.2.3)
  The following recommendations of this safeguard could be automated:
  - Change preset passwords with C1 "Passwords"

  The following recommendations (see [7]) could not be automated:

  - Depending on the the device and its method how to store the password it is not always possible to check the password.

- Secure use of Protocols and Services (see 4.1.2.4)
  The following recommendations of this safeguard could be automated:
  - ICMP handling with C7 "ICMP"
  - Find data paths for different protocols and ports with C6 "Policy"
  - An overview overview of protocols admitted at the security gateway with the parsing of the firewall configuration I6

  The following recommendations (see [7]) could not be automated:
  - The configuration recommendations for different services such as ftp or ssh
  - The use of secure routing protocols
- Configuration of Access Control Lists on Routers (see 4.1.2.5)
  The following recommendations of this safeguard could be automated:
  - The firewall filter rules should be compliant with the IT requirements with with C6 "Policy"
  - Whitelist approach with C5 "Whitelist"
  - Logging of rules must be enabled with C2 "Logging"

  The recommendations could be fully automatically verified.
- Handling of ICMP on the Security Gateway (see 4.1.2.6)
  The following recommendations of this safeguard could be automated:
  - Handling of ICMP messages of computers in the internal network an of public servers in the DMZ with C7 "ICMP"

  The following recommendations could not be automated:
  - Is the handling of ICMP messages documented

The focus of this thesis was a small selection of safeguards and provides a groundwork for further work which may extend this approach with additional checks to evaluate additional recommendations.

8.3 Network Model

In the course of this work a network model as basis input for the evaluation of the selected safeguards was developed. The model, which is based on directed graphs

allows to represent a Ethernet-based enterprise network and its data paths up to the firewall (layer four). The developed model has the following benefits:

- The model allows simple path checks as all possible connections between end devices are modelled as a pair of nodes and one edge. Therefore, in order to query a connection between "A" and "B" it is only necessary to check if there is an edge between those nodes. There is no need for graph traversing.
- The model's approach is straight forward in basically always using the same algorithm to build the different layers. Accordingly, the approach of substituting intermediate network elements with new edges in the graph is the basic idea of the network model.
- The network model can be used in another context whether for other safeguards or for other use cases, e.g. kind of a documentation of the network for administrators.
- The network model is kept simple and for its implementation only a directed property multi graph is required. This allows simple extensions and debugging in the course of further work.
- Graphs are a wide spread model and there are several tools to visualize graphs. This model, as being based on a graph, can be easily visualized for administrators to have an overview of the network.

8.4 Results from the Use Case

In the following sections the results of the implemented checks are illustrated.

8.4.0.1 BSI Safeguards

- Check C1 "Passwords"
  Listing 8.1 illustrates the output of the check. The Fortigate firewall configuration file **only** has this entry if the standard password is changed.

Listing 8.1: C1 Output (fake)

```
1 =============== Passwords ================
2 admin;AKAfsarFgLHJAd36/J+wfasf3GASF3gdgSas=
3 backup-ro;AGDAD/Sfas0X9xgCqyQXIpCtafsGEGAddQasfa=
4
```

- Check C2 "Logging"
  Listing 8.2 illustrated the output of this check. Accordingly to the configura-

tion file global logging is enabled but the policy specific traffic (both logging of accepted and denied) packages is not activated (for all filter rules).

Listing 8.2: C2 Output

```
1  =============== Log Memory ===========
2  true
3
4  =============== Syslog ================
5  true
6
7  ...
8  CheckFWLogs: Warning! logTraffic of policy 106 not
       activated
9  CheckFWLogs: Warning! unmatchedTraffic of policy 106
       not activated
10 ...
11
```

- Check C3 "Bypassing" "
  Listing 8.3 illustrates the output of check C3. 240 unidirectional connections respectively 120 bidirectional connections were revealed. Devices part of these connections are directly connected on the regarded layer one.

Listing 8.3: C2 Output

```
1  Warning: dc1-switch04-00 is directly connected to
       dcesx01
2  Warning: dc1-switch04-00 is directly connected to
       proxmox04
3  ....
4
```

- Check C5 "Whitelist" "
  Listing 8.4 illustrates the output of this check. Three filter rules do have an empty "action" value. For Fortigate firewalls these policies enforce the default action which is in this case deny and therefore not compliant to the recommendations.

Listing 8.4: C5 Output

```
1  CheckWhitelist: Warning! policy 236 has " " as action
2  CheckWhitelist: Warning! policy 237 has " " as action
3  CheckWhitelist: Warning! policy 302 has " " as action
4
```

8.4.0.2 Other Results

Furthermore, in the course of this work a lack in the network documentation was revealed:

- Missing devices in the documentation
- lack of a uniform naming of devices in the documentation and configuration files

8.5 Extensible

The architecture's design has a clear focus on its extensibility (see section 4.2.1 and section 5.3. The architecture consists of four major components:

- Parsers which read the required inputs
- Data structures which stores the information from the parsers
- The network model built from the data structures
- Checks operating with the network model and the inputs

By providing additional parsers which fill the data structures the architecture can be adapted to custom inputs, e.g. devices from different manufacturers. The model itself is built only with the internal data structures and independent from the parsers. The checks are independent form each other and can be easily extended with new ones.

Chapter 9

Conclusion and Outlook

In this work the question has been raised whether an automatic support for verifying the guidelines the BSI Baseline Protection Catalogues recommends is possible. This thesis answers this question by analyzing the content of the catalogues and presenting a selection of recommendations whose evaluation is automated.

A closer look at the Baseline Protection Catalogues resulted in six different safeguards where an automatic approach for verifying the implementation status of the safeguards' recommendations looked suitable. These safeguards are all in the field of computer networks with a strong focus but not limited to firewall filter rules. Seven checks which can be automated were defined covering the main recommendations of the selected safeguards (see chapter 4).

The software requirements which are needed to evaluate those recommendations are described as well as their input sources (see section 4.2).

Additionally, a network topology model which allows to model ISO/OSI layer one till four was developed as some checks require knowledge of the network (see 5.4).

Based on these findings an example implementation was coded and four checks in a use case scenario evaluated (see chapters 6 and 7).

This thesis describes an approach of supporting the implementation of the recommendations of the safeguards. It is illustrated that there is some content being beneficial to automate. Besides that a network topology model based on directed property multigraphs is developed and introduced.

This work illustrates that on the one hand there is a lack of automatic tool support (see section 3.1) and on the other hand that there is potential in the Baseline Protection Catalogues to automate certain aspects. Among others this thesis illustrates how to verify if devices are able to communicate with each other bypassing a firewall or how to check the data paths in a network including ICMP messages. Further results describes

the checks for evaluation the whitelist approach and the handling of deny rules (see section 5.6).

All in all this thesis shows a possibility to support the Baseline Protection implementation and may be a foundation for further work in the field of BSI safeguards recommendation automation and network topology modelling.

Focus of the developed architecture was its modularity (see 5.3). Further work may include other input resources such as devices from different vendors respectively their configuration files. Additionally, more checks may be added to increase the number of safeguards recommendations which are supported.

The advantage of BSI Baseline Protection covering a huge area of today's IT systems is also a disadvantage. Due to the huge amount of information (see section 2.3) it is a time-consuming task to even stay on top of things. An approach to handle the information may be in the area of natural language processing and semantic search. With the help of this technologies the data of the catalogues can be "parsed" and worked with in a natural and dynamic way.

The model (and the checks) are only related to network layer four or lower, further research could be done in the field of application security. In this context application security is not related to secure code but to secure configuration of applications. In this context two major issues to automatically check may be focused on.

- Secure Configuration of (network-related) applications such as web servers, DHCP Servers, or DNS servers.
- Up-to-date software of services, operating systems and applications.

The algorithms respectively the software was not developed with respect to performance and works fast on small sized networks. Further work my evaluate the performance impact with large networks and develop optimization regarding the algorithms and the memory management.

One limitation of the network model is the lack of the ability of modelling the dynamic migration of virtual machines between different hosts. Future work may add this feature to the network model by modelling the dynamics of the migration (e.g assign a virtual machine to a cluster of hosts) and not only handling them statically (as a snapshot).

The introduced network topology model in 5.4 can be formalized in a further step. Future work may develop a meta description (similar to the ones introduced in section 3.2.2) for interchangeability and therefore as input for other tools or libraries or an export function to INDL/NML if those become wide spread.

Bibliography

[1] “Sony Hack.” [Online]. Available: http://www.heise.de/thema/Sony_Pictures_Hack

[2] “Bundestag Hack.” [Online]. Available: http://www.heise.de/newsticker/meldung/Angriff-auf-Datennetz-des-Bundestags-2651339.html

[3] “OPM Hack.” [Online]. Available: http://www.golem.de/news/nach-hackerangriff-opm-chefin-katherine-archuleta-tritt-zurueck-1507-115175.html

[4] “Data Breaches.” [Online]. Available: http://www.businessinsider.com/heres-how-big-the-most-recent-hacking-data-breaches-have-been-2014-10?IR=T

[5] “BSI: Startseite Bundesamt für Sicherheit in der Informationstechnik.” [Online]. Available: https://www.bsi.bund.de/DE/Home/home_node.html

[6] G. federal office for information security, “Information security audit (IS audit),” 2008.

[7] F. Office, “BSI: IT-Grundschutz catalogues - 13th version 2013,” pp. 1–4220.

[8] “BSI Tools.” [Online]. Available: https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Hilfsmittel/GrundschutznaheTools/grundschutznahetools_node.html

[9] “BSI Toolangebote.” [Online]. Available: https://www.bsi.bund.de/DE/Themen/weitereThemen/GSTOOL/AndereTools/anderetools_node.html

[10] “Checklisten.” [Online]. Available: https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/hilfmi/checklisten/checklisten.html

[11] “Verinice.” [Online]. Available: http://www.verinice.org/

[12] “MaSSHandra.” [Online]. Available: https://www.masshandra.com/

[13] “The Dude.” [Online]. Available: http://www.mikrotik.com/thedude

[14] "10Strike Network Diagram." [Online]. Available: https://www.10-strike.com/network-diagram/

[15] "Spiceworks." [Online]. Available: http://www.spiceworks.com/de/

[16] J. V. D. Ham, P. Grosso, R. V. D. Pol, A. Toonk, and C. D. Laat, "Using the Network Description Language in Optical Networks," *10th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 199–205, 2007. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4258536

[17] F. Dijkstra, J. V. D. Ham, and A. Patil, "Network topology descriptions in hybrid networks," *Network*, pp. 1–14, 2009. [Online]. Available: http://ogf.org/Public_Comment_Docs/Documents/2010-01/NML-WG-D1-Context-20091012.pdf

[18] R. Łapacz, J. Zurawski, G. Working, D. Gwd, and R. P. R-p, "Network Markup Language Base Schema version 1," pp. 1–69, 2012.

[19] M. Ghijsen, J. Van Der Ham, P. Grosso, and C. De Laat, "Towards an infrastructure description language for modeling computing infrastructures," *Proceedings of the 2012 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA 2012*, pp. 207–214, 2012.

[20] M. Ghijsen, J. Van Der Ham, P. Grosso, C. Dumitru, H. Zhu, Z. Zhao, and C. De Laat, "A semantic-web approach for modeling computing infrastructures," *Computers and Electrical Engineering*, vol. 39, pp. 2553–2565, 2013.

[21] "MySQL Connector." [Online]. Available: http://dev.mysql.com/downloads/connector/j/

[22] "Jung." [Online]. Available: http://jung.sourceforge.net/

[23] "Java Json." [Online]. Available: http://www.json.org/java/

[24] "Racktables." [Online]. Available: http://racktables.org/

[25] "Fortigate Handbook." [Online]. Available: http://docs-legacy.fortinet.com/fos50hlp/50/index.html#page/FortiOS5.0Help/policies.068.04.html

Appendix A

Appendix

A.1 Checklists from the BSI

Figure A.1 illustrates a checklist for the module "B1.3" (business continuity management). This module has five threats ("G x.x") and eleven safeguards "M 6.110 - M 6.120"). Each of the safeguards counter the corresponding threat (indicated by the "X"). Column two "Lebenszyklus" describes the life cycle phase and column three "Siegelstufe" the classification level. The corresponding englisch life cycle phases are as follows: PK is "Planning and Design", BE is "Purchasing", UM is "Implmentation", BT is"Operation", AU is "Disposal", and NV is "Contingency Planing ".

```
"B 1.3";"Lebenszyklus";"Siegelstufe";"G 1.1";"G 1.2";"G 1.10";"G 1.18";"G 1.19"
"M 6.110";PK;C;X;X;X;X
"M 6.111";PK;A;X;X;X;X
"M 6.112";UM;A;X;X;X;X
"M 6.113";UM;C;X;X;X;X
"M 6.114";UM;A;X;X;X;X
"M 6.115";UM;C;X;X;X;X
"M 6.116";UM;C;X;X;X;X
"M 6.117";BT;B;X;X;X;X
"M 6.118";BT;A;X;X;X;X
"M 6.119";BT;C;X;X;X;X
"M 6.120";BT;C;X;X;X;X
" "
```

Figure A.1: Content of a Checklist

A.2 Fortinet Fortigate Policy Structure

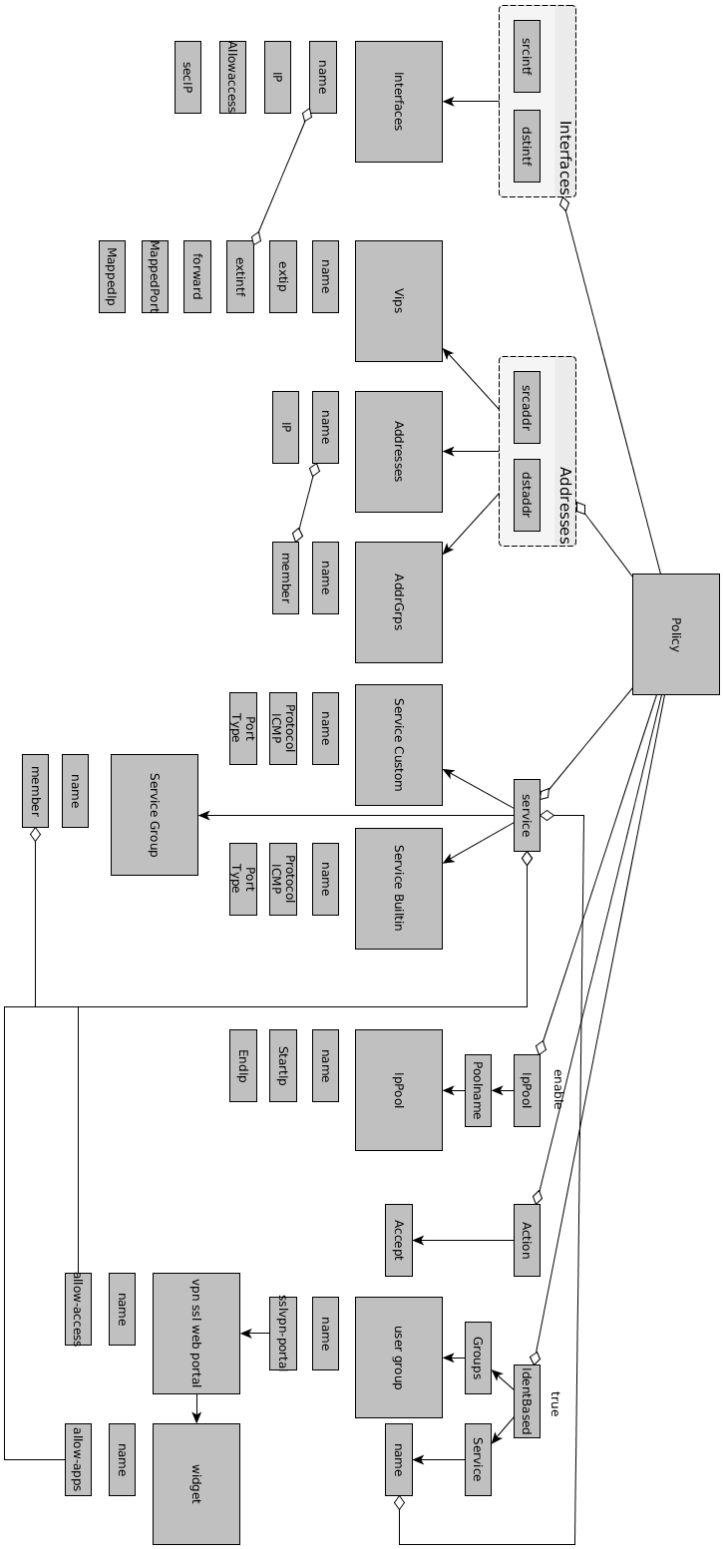

Figure A.2: Policy Structure