Fachhochschule Hannover
University of Applied Sciences and Arts

Fachhochschule Hannover
University of Applied Sciences and Arts

Faculty IV
Department of Computer Science

Master Thesis

Analyzing and Integrating TNC and VPN Technologies

August 2009

Author: Alexander Reich

First Examiner: Prof. Dr. Josef von Helden

Second Examiner: Prof. Dr. Stefan Wohlfeil

Author

Alexander Reich
Peiner Straße 47
30519 Hannover
E-Mail: contact@alexander-reich.com

First Examiner

Prof. Dr. rer. nat. Josef von Helden
Fachhochschule Hannover
Ricklinger Stadtweg 120
30459 Hannover
E-Mail: josef.vonhelden@fh-hannover.de
Telefon: 0511 / 9296-1500

Second Examiner

Prof. Dr. rer. nat. Stefan Wohlfeil
Fachhochschule Hannover
Ricklinger Stadtweg 120
30459 Hannover
E-Mail: stefan.wohlfeil@fh-hannover.de
Telefon: 0511 / 9296-1818

Thesis Declaration

I warrant, that the thesis is my original work and that I have not received outside assistance. Only the sources cited have been used in this draft. Parts that are direct quotes or paraphrases are identified as such.

[German Translation]

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die eingereichte Masterarbeit selbständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Hannover, den 31. August 2009

Alexander Reich

Contents

List of Figures

List of Tables

Listings

Acronyms

TPM Trusted Platform Module

TCG Trusted Computing Group

TNC Trusted Network Connect

TNC-WG Trusted Network Connect Work Group

AR Access Requestor

PEP Policy Enforcement Point

PDP Policy Decision Point

VPN Virtual Private Network

NAC Network Access Control / Network Admission Control

IKE Internet Key Exchange

TLS Transport Layer Security

LDAP Lightweight Directory Access Protocol

1 Introduction

1.1 Motivation

Network Access Control (NAC)[1] refers to a technology for the protection of a network against potentially dangerous endpoints. Endpoints are all devices that intend gaining access to the network, like among other things workstations, notebooks and servers. This verification takes place as soon as an endpoint requests access to the protected network. Trusted Network Connect (TNC) claims to be a secure alternative to existing NAC technologies due to offering a reliable detection of lying endpoints by using the capabilities of a Trusted Platform. Furthermore, TNC offers a high interoperability with existing standards and technologies.

At present, TNC implementations are usually implement on top of port authentication according to IEEE 802.1X since in that way network access can be securely prevented if the endpoint does not match the required security policy. As a result, Trusted Network Connect can only be deployed in environments with corresponding hardware and software capable of IEEE 802.1X (switches, routers, security gateways, etc.). Furthermore, that approach restricts the use only to local endpoints. However, in the last years VPNs (Virtual Private Networks) became more important for companies and today for lots of companies it is unimaginable to do without. VPNs are being used for connecting several locations or for granting employees access to the corporate network from outside.

TNC in combination with IEEE 802.1X offers a reliable network access control for local endpoints but not for remote endpoints accessing the network via the VPN. Without considering remote access the gained security increase due to the application of a NAC technology like TNC becomes vulnerable because potentially dangerous endpoints may access the network by using the VPN (of course, valid authentication credentials are required).

Integrating TNC and VPNs will offer a significant security increase to modern networks where endpoints are connected by local switches as well as by a Virtual Private Network.

[1] In the literature sometimes also referred to as Network Admission Control.

1.2 Objectives

The objective of this master thesis is the analysis of requirements and the development of a concept regarding the integration of Trusted Network Connect and Virtual Private Networks. A result of analyzing the requirements will be a detailed catalog of requirements which is being used as basis for the development of the integration concept. Furthermore, an exemplary implementation of the concept as well as the establishment of a test environment for future research will be covered by this thesis.

1.3 tNAC Project

This master thesis is written within the scope of the tNAC project[2]. The tNAC project is a research & development project with the aim of developing an open NAC implementation based on the open security platform *Turaya*[3] and the open Trusted Network Connect implementation *TNC@FHH*[4]. The project was started in July 2008.

1.4 Structure

The structure of this thesis is divided into three parts each with an own focus. The first part gives a short introduction to Virtual Private Networks and introduces with IPsec and OpenVPN two well known VPN technologies in chapter 2. Chapter 3 introduces the basics and especially the architecture of Trusted Network Connect. These two chapters builds the basis for understanding the requirements and the concept introduced in the subsequent chapters.

The second part of the thesis deals with the analytical and conceptual part of an integration of both technologies. Chapter 4 introduces several requirements concerning an integration and finishes the chapter with presenting a catalog of requirements. The subsequent chapter 5 presents and evaluates several integration methods and selects one as basis for the development of the integration concept. Chapter 6 introduces the integration concept in detail and finishes with an evaluation and compares the concept with the catalog of requirements from chapter 4.

The third and last part of this thesis presents the exemplary implementation of the integration concept and explains the essential development steps. This chapter will also illustrate that an endpoint can only access the protected network by passing

[2] http://www.tnac-project.org/
[3] http://www.emscb.de/content/pages/About-Turaya-de.htm
[4] http://trust.inform.fh-hannover.de/

the VPN authentication as well as the TNC assessment. A conclusion at the end of this thesis gives a summary of the integration of TNC and VPN and provides perspectives concerning possible further research.

1.5 Scope of Content

This master thesis provides no comprehensive introduction to the technologies TNC and VPN but presents them as a kind of overview and only explains those parts in detail which are relevant regarding the analysis and integration. The prototyped implementation of the concept is only exemplary and neither destined for a productive application nor qualified. Indeed, the exemplary implementation is almost independent of the operating system but the establishment of a test environment as well as the development of a prototype is done under Linux.

The installation and configuration of TNC, VPNs and further required components is not described in detail within this thesis but the reader can find references to further literature at the corresponding sections.

2 Virtual Private Networks

This chapter introduces Virtual Private Networks (VPN) and explains the different use cases of a VPN. Furthermore, this chapter presents two popular VPN technologies which are commonly deployed in order to establish a Virtual Private Network. Both technologies will be evaluated in regard to the planned integration of TNC and VPN.

2.1 Introduction

A Virtual Private Network (VPN) allows connecting two or more distributed networks by using an existing network connection. In practice, VPNs are commonly used to connect two or more business units over the internet. In the past, companies were forced to buy dedicated connections between their business units in order to realize a company-wide network. This approach is very expensive and only whole business units are connected. As a consequence, employees are not able to access the network when they are away on business or working at home.

A Virtual Private Network claims to be a cheap and secure solution for exactly that problem. Figure 2.1 illustrates a VPN between two local networks and demonstrates in addition, that employees (e.g. sales-man) are able to access that VPN from the outside.

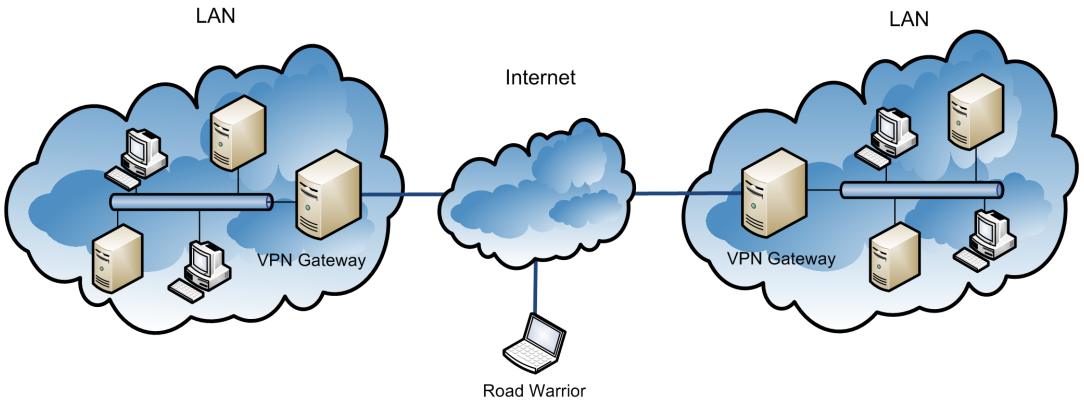

Figure 2.1: Overall structure of a VPN

As already mentioned, VPNs are usually implemented on top of a internet connection since the internet offers a cheap network connection between different locations. Unfortunately, the internet is not convenient for sending confidential information since the communication is not protected against eavesdropping. For that reason, popular VPN solutions use strong cryptography in order to enable secure communication between different business units and single connected hosts.

Furthermore the VPN software must verify the authenticity of the opposite endpoint before establishing a secure encrypted channel. Endpoint authentication is absolutely mandatory in order to prevent so called *Man in the middle Attacks* (MitM). More information regarding MitM attacks can be found at [2].

The way of how the network traffic is being encrypted and the verification of authenticity is achieved depends on the deployed VPN technology. Two popular VPN technologies are IPsec and OpenVPN and will be introduced in section 2.3.1 and 2.3.2 of this chapter.

2.2 VPN Categories

VPNs can be divided into two categories depending on the kind of resource which is connected to the VPN. As already explained in the introduction to this chapter, VPNs can be used to connect single hosts with a network or whole networks with other networks. Any combination is also possible. Understanding both categories is of great importance when it comes to the development of an integration concept in the following chapters.

2.2.1 Site-to-Site VPNs

Site-to-Site VPNs are being used to connect two or more networks. Each network (site) is connected via a so called *VPN gateway*. The job of the VPN gateways is to establish the VPN connection with the opposite gateway and routing the network traffic from the local network to the destined network (identified by the IP address of the opposite gateway) and vice versa. The VPN software is only on the respective gateways required. The physical detachment of the virtual network will not be considered by the endpoints behind their respective VPN gateways.

For example, host A of network A wants to communicate with host B of network B. Host A sends the packet to the local gateway A. Gateway A then encapsulates and encrypts the packet and forwards it to gateway B over the internet. Gateway B decrypts and decapsulates the packet and forwards it to host B. The overall example is illustrated in figure 2.2.

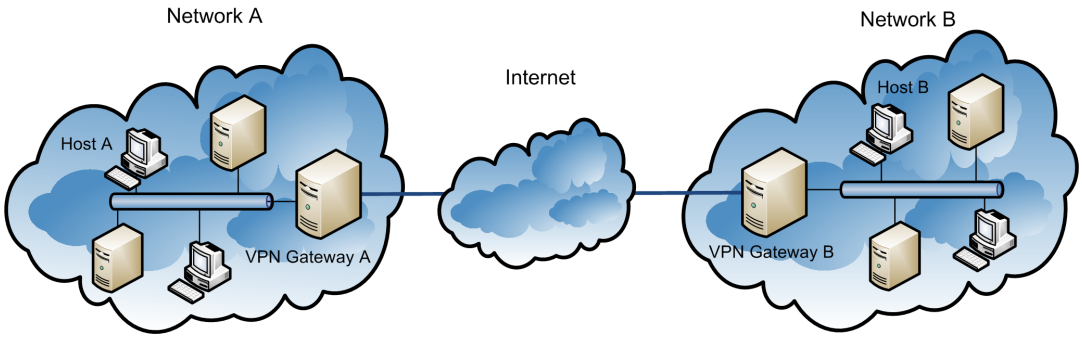

Figure 2.2: Structure of a Site-to-Site VPN

2.2.2 Road Warrior VPNs

Road Warriors describe single hosts which will be connected to the VPN. The main difference when compared to Site-to-Site VPNs is that the connected host directly communicates with the opposite VPN gateway and is therefore responsible for establishing the VPN connection. In this case the VPN software is additionally required on each Road Warrior. Figure 2.3 illustrates two Road Warriors which are connected to a LAN by using a Virtual Private Network.

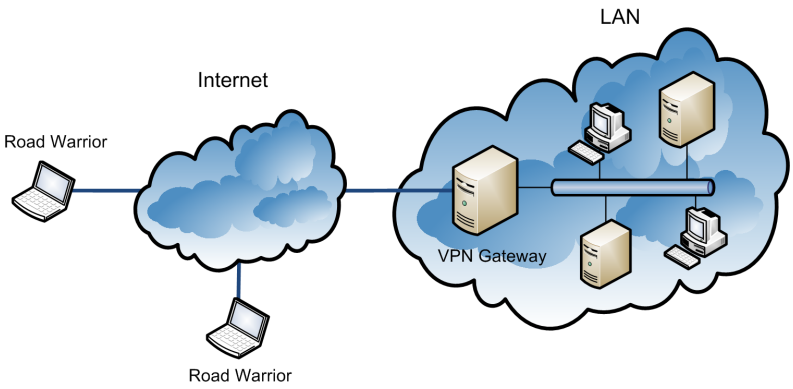

Figure 2.3: Structure of a Road Warrior VPN

2.3 VPN Technologies

The term VPN just refers to a general concept of how connecting several network segments through a public available network like the internet. For that reason there are a lot of different technologies and implementations available on the market.

VPNs can be realized on different layers of the OSI reference model[1]. A list of several VPN technologies and their corresponding OSI layer is available under [4].

Currently, *IPsec* and *OpenVPN* are two well known and competiting VPN technologies and are often being considered when it comes to the implementation of a Virtual Private Network. The following sections introduce both technologies in order to establish a basis and explain those details which are relevant for the further chapters (such as secure IP address assignment).

2.3.1 IPsec

IPsec is a widely supported VPN standard, implemented by a lot of network hardware manufacturers and is available for many operating systems. IPsec is arranged at layer three of the OSI reference model and can be seen as an extension to the Internet Protocol (IP). IPsec enhances the Internet Protocol by adding support for integrity, authentication and cryptography.

One advantage of IPsec is, that it is completely standardized in many RFCs and allows therefore interoperability between different operating systems and network hardware devices[2]. For that reason, a Site-to-Site VPN can be realized with a Cisco IPsec enabled appliance at one side and a Linux host with *strongswan* on the other side. IPsec is widely used in a lot of production environments due to the availability since many years and the support of many network hardware manufacturers like Juniper[3], Cisco[4], 3com[5] and dlink[6]. For Linux operating systems, two popular implementations are *strongswan*[7] and *openswan*[8].

As already mentioned IPsec is not a certain VPN implementation but a suite of many standardized protocols each with its own focus. Understanding the internals is not related to this thesis and is described in detail by [5]. However, there is one protocol which is important concerning an integration of IPsec and Trusted Network Connect - the IKE protocol.

2.3.1.1 Internet Key Exchange

The Internet Key Exchange (IKE) protocol is responsible for negotiating several parameters (such as supported authentication and encryption algorithms), authen-

[1] The OSI reference model is explained in detail under [3].
[2] Requires that completely standard compliant implementations are deployed in an environment.
[3] http://www.juniper.net/
[4] http://www.cisco.com
[5] http://www.3com.com/
[6] http://www.dlink.com/
[7] http://www.strongswan.org/
[8] http://www.openswan.org/

ticating the involved endpoints and for setting up a security association. The security association stores security parameters such as the prior negotiated key exchange and encryption algorithms. Detailed information regarding IKE is available at RFC 2409[6].

The latest version of the IKE protocol is version 2 and was released in December 2005. One enhancement of IKEv2 is the support of the Extensible Authentication Protocol for authenticating the VPN parties. That seems to be an important improvement of IKE regarding the planned integration of Trusted Network Connect and VPNs since TNC also supports EAP for authentication purposes. IKEv2 is specified in RFC 4306[7].

2.3.1.2 Secure IP Address Assignment

This section covers how IPsec implementations can detect or prevent that a legitimate VPN client changes his IP address in order to gain further access to the network or to circumvent IP based security restrictions.

Several implementations supports virtual IP address assignment. That feature can be used to assign a certain IP address to a VPN client. For example, the *strongswan* implementation can assign IP addresses which are either statically defined for a certain client or selected from a pool of available addresses[9]. Virtual IP address assignment can also be used by *strongswan* to prevent that a client changes the IP address in order to gain further access to the Virtual Private Network. However, whether that feature is also used to detect and prevent IP spoofing is up to the implementation.

2.3.1.3 Evaluation

The main advantage of IPsec is that it is a fully standardized protocol suite and is widely supported by many operating systems and hardware vendors which results in multi-vendor compatibility. Furthermore IPsec is available for many years now and the protocol suite itself can be considered as secure. However, that does not apply for any implementation of IPsec.

One downside of IPsec is that it is a very complex protocol suite consisting of many protocols and standards. That is one reason why IPsec is rumored to be very difficult to configure and thus much more vulnerable and insecure than similar technologies which are easier to configure. The high complexity of IPsec and

[9] More information regarding this feature and the configuration is available under http://wiki.strongswan.org/wiki/strongswan/VirtualIp

the resulting liability for mistakes was also criticized by Bruce Schneier and Niels Ferguson in their cryptographic evaluation of IPsec [8].

Furthermore IPsec requires low level packet manipulation and this part is usually implemented in kernel space (e.g. as a kernel module). A security related vulnerability in this part of the application may jeopardize the overall system security.

2.3.2 OpenVPN

OpenVPN[10] is an open and free VPN implementation built on top of the TLS protocol[9]. The concept behind OpenVPN is quite simple. OpenVPN establishes a TLS encrypted tunnel between both endpoints and uses that tunnel for sending IP packets from one network (or host) to the other. Routing traffic through the TLS tunnel is realized due to the use of virtual network adapters. Every traffic which will be send through these virtual adapters will be encrypted by TLS before it will be submitted.

Using virtual adapters for sending VPN traffic between the endpoints instead of low level kernel modules allows OpenVPN to be completely realized as user land application. That behavior increases the overall system security since if a security related issue or bug is found in a kernel space application the entire system security and stability is vulnerable at the benefit of a better performance and throughput.

2.3.2.1 Establishing Secure Connections

As already mentioned above, OpenVPN uses no complex protocols like IKE for establishing a secure and encrypted tunnel. OpenVPN uses TLS for encrypting the tunnel between two hosts and a Certification Authority (CA) is commonly used for authentication purposes. The CA is used by OpenVPN to determine whether a client is allowed to access the VPN or not. Every certificate which is signed by that certain CA is allowed to access the VPN unless that certificate is listed on a Certificate Revocation List (CRL). However OpenVPN supports also password-only authentication but using this is discouraged and therefore disabled by default since it is vulnerable to brute-force attacks.

2.3.2.2 Secure IP Address Assignment

Similar to strongswan, OpenVPN is also capable of assigning IP addresses to a client. By default, OpenVPN prevents IP spoofing of a connected client by comparing the source address of any incoming packet with an internal association table. This table

[10] http://www.openvpn.net/

contains a list of valid IP addresses for each client. That behavior results from a configuration parameter which tells the OpenVPN server which IP addresses and networks are behind a certain client (required for Site-to-Site VPNs). If no further IP addresses or networks are mentioned, OpenVPN will drop all packets with a different IP address for that endpoint. For that reason, this behavior can be used to prevent IP spoofing of the VPN client. This feature is rarely documentated[11] and only some hints can be found in the *man* page of OpenVPN.

2.3.2.3 Evaluation

OpenVPN claims to be a secure and easy to configure alternative to the widely used IPsec protocol suite. OpenVPN is not standardized like IPsec but it is open source and free software. The concept of using virtual network interfaces in order to send VPN traffic through a TLS tunnel to the opposite VPN gateway is straightforward and allows the administrator to modify the routing table easily or add certain packet filter rules to the VPN gateways.

Furthermore OpenVPN is out of the box capable of routing non IP based traffic when using the TAP device. Further information regarding OpenVPN can be found at [4].

2.3.3 Conclusion

IPsec as well as OpenVPN can be used to implement a secure Virtual Private Network. The main difference is that IPsec is a complex protocol suite consisting of many defined standards whereas OpenVPN is a free and open source VPN solution. OpenVPNs strength lies on the straightforward design and the therefore resulting simple configuration when compared to IPsec. On the other hand, IPsec is supported by many hardware devices and software manufacturers. Due to the fact that IPsec is just a standardized protocol suite, many implementations provide additional features or security addons which are not defined by the standard. Further details of strongswan or OpenVPN such as the extensibility will be presented in the requirements analysis and subsequent chapters if necessary. Table 2.1 presents an overview of the technologies OpenVPN and IPsec (strongswan is considered as IPsec implementation).

[11] The corresponding lines of code can be found in file *multi.c* on line 1635 of version 2.0.9.

Property	OpenVPN	IPsec (strongswan)
Easy to install and configure	Yes	No
Can be seen as stable and secure	Yes	Yes
Widely supported by many manufacturers	No	Yes
Implementation is Open Source	Yes	Yes
Official Standardized (e.g. RFC or IEEE)	No	Yes

Table 2.1: Properties of OpenVPN and IPsec

2.4 Weaknesses of VPN Solutions

Modern VPN solutions prove the authenticity and provide secure authentication of VPN clients. As a result, only endpoints holding valid authentication credentials are able to connect to the VPN. Unfortunately, an attacker can also easily access the network if he either controls an authenticated endpoint by using a backdoor or if he obtain valid authentication credentials. A VPN software is not designed to protect the network against these threats.

The reason therefore is that the VPN software does not consider the system integrity when authenticating the endpoints. For example, an user holding a valid certificate can connect to the protected network over the VPN although his computer is infected by malware. Verifying the integrity of network endpoints is out of scope for VPN solutions and is the domain of *Network Access Control* technologies such as *Trusted Network Connect*. An integration concept which combines the use of VPNs and TNC in order to solve the described problem will be introduced in chapter 6.

3 Trusted Network Connect

The introduction to this thesis has already mentioned what Network Access Control solutions are and that *Trusted Network Connect (TNC)* claims to be a secure alternative compared to its competitors by offering secure detection of lying endpoints. This chapter gives an introduction to TNC and explains several parts of the architecture in detail which belong to the development of an integration concept. However, this chapter expects basic knowledge of Trusted Computing and the Trusted Platform Module. A good introduction to Trusted Computing and TNC is available under [10].

3.1 Introduction

Trusted Network Connect (TNC) is an open, manufacturer-independent Network Access Control (NAC) specification developed by the Trusted Computing Group. The focus is on the trustworthy integrity measurement of the endpoints on the basis of the prospects offered by the Trusted Computing. TNC avoids a significant weakness of many NAC implementations – the so-called lying endpoints. Furthermore, TNC offers good interoperability with existing standards and technologies.

The term lying endpoint refers to an endpoint which is transmitting a forged integrity status to the verifying system. If an endpoint is able to generate arbitrary measured values (forgeability), without the verifying entity being able to detect it, the NAC solution is vulnerable and attackers can gain access to the network. For example, the malware changes e.g. important system files or stops the anti-virus scanner. The system changes have to result in a new integrity value which afterwards will be detected by the Policy Decision Point. However, the attacker is still able to forge and transmit integrity values to the PDP, but these values are being detected reliably as being false.

But the use of Trusted Network Connect only for the protection of a network does not solve the problem. TNC has indeed be developed based on the functions of the Trusted Computing, but doesn't obligatory require a Trusted Platform Module on the endpoints. Thereby TNC can also be applied in networks where not all devices

dispose of a TPM. Without a TPM, the PDP is not able to detect a lying endpoint reliably. A detailed introduction to this topic in available under [10].

3.2 Architecture

TNC classifies the architecture in three logic layers, the Network Access Layer, the Integrity Evaluation Layer and the Integrity Measurement Layer as illustrated in figure 3.1. Within the scope of this master thesis, only the Network Access Layer and the corresponding protocols are being presented in detail, since they are important for the further understanding of this master thesis. The entire TNC architecture is being described in detail in [1].

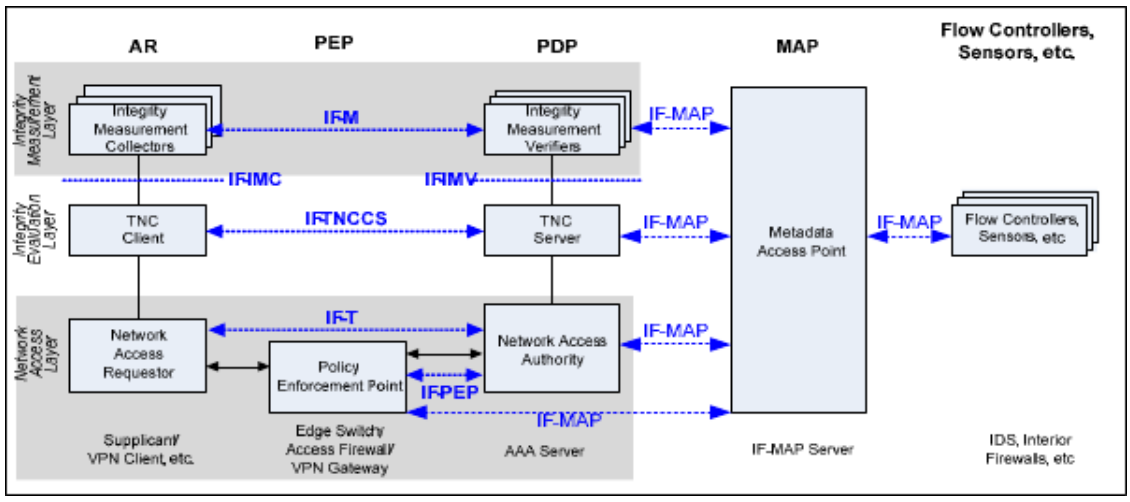

Figure 3.1: TNC architecture (Source: [1])

Understanding the essentials of the TNC architecture is of great importance for the comprehension of the requirements and the development of an integration concept.

3.2.1 Entities and Components

TNC defines three entities, which respectively perform a concrete task within the TNC architecture. These entities may be composed of several components, according to the respective layer as shown by figure 3.1. The components communicate on the same layer with a respective component of the destination entity.

3.2.1.1 Access Requestor (AR)

An Access Requestor (AR) is an endpoint, which wants to gain access to a TNC protected network. The Access Requestor is composed of the Network Access Requestor (NAR), the actual TNC client (TNCC) and the Integrity Measurement Collectors (IMCs). IMCs are individual modules, which each perform a particular integrity measurement. For example, an IMC could measure the patchlevel of the operating system while an other one measures the signature database of the anti-virus software. The IMCs are described in detail in the corresponding specification[11].

The TNC client has the task to execute these individual IMCs and to collect as well as to transmit the measured integrity values to the Policy Decision Point. The NAR provides functions for the secure transport of the TNC data on network level. In a VPN scenario the VPN client would be a part of the Access Requestor.

3.2.1.2 Policy Enforcement Point (PEP)

The Policy Enforcement Point is an optional entity in the TNC architecture. The PEP may generally be considered as entry point into a TNC protected network. The PEP receives connection requests of the Access Requestor and forwards them to the PDP for verification. The PEP has to be the entry point into the protected network since it is responsible for the enforcement of isolation recommendations by the Policy Decision Point and therefore permits or denies access to the network. In case of a VPN integration, the Policy Enforcement Point corresponds to the VPN gateway.

3.2.1.3 Policy Decision Point (PDP)

The task of the Policy Decision Point is the verification of the transmitted integrity measurements from the Access Requestor. The Network Access Authority (NAA) component receives packets on the network layer, extracts the TNCCS[1] messages and forwards them to the TNC server component. The TNC server invokes the appropriate Integrity Measurement Verifiers (IMVs). An IMV is the counterpart to an IMC on the Access Requestor and decides whether the transmitted measurement meets the policy requirements or not. The IMVs are described in detail in the corresponding specification[13]. After the integrity verification of an endpoint is finished, the PDP sends an appropriate recommendation to the Policy Enforcement Point.

[1] IF-TNCCS[12] is responsible for the message exchange on the Integrity Evaluation Layer between TNC client and TNC server.

3.2.2 Phases

The entire process from the connection request, integrity measurement and transmission of the TNCCS messages from the Access Requestor to the Policy Decision Point, to the verification and decision on its integrity, is called *assessment phase*. Possible results of this phase would be to grant network access, to deny network access or to grant only partial access to the network where the endpoint gets the possibility to remediate.

The *isolation phase* is responsible for an endpoint which failed the integrity verification and was therefore isolated. In the *remediation phase* the endpoint has the chance to adapt its integrity to the required policy, for example via installation of required operating system updates or updating the anti-virus signatures. Afterwards the endpoint can pass through the assessment phase anew.

3.2.3 Isolation Strategies

As already mentioned in section 3.2.2 above, TNC considers not only that access to the network will be either granted or denied but also limited access to only certain systems or services is considered. TNC defines three isolation strategies at the moment:

Binary-based: Either access will be granted or denied to the network. Binary-based isolation is the most stringent type of endpoint isolation. In case of a failed integrity verification no access to the network is granted.

VLAN-based: If the endpoint failed the integrity verification, it will be jailed into a certain VLAN which only grants access to limited systems. This limited access can be used for remediation.

Filter-based: If the endpoint failed the integrity verification there will be additional filter rules deployed (ACLs or packet filter rules) on the PEP. For example, these rules may grant access to only certain systems or certain services. Similar to VLAN-based isolation, this limited access can be used for remediation by an endpoint.

3.2.4 Interoperability of TNC

One aim concerning the design of Trusted Network Connect is to achieve a high interoperability with other technologies and a seamless integration in existing network infrastructures. For example, there is already an integration with Microsoft's

Network Access Protection (NAP)[2] which enables the communication between a TNC client and a NAP server and vice versa.

In addition to that, TNC considers several transport protocols, authentication solutions, as well as different opportunities for the network access. However, many of them are destined for the future. More information regarding the interoperability and support of further technologies is available under [1].

TNC and IEEE 802.1X

At the moment, TNC is usually used in conjunction with IEEE 802.1X. Port authentication according to IEEE 802.1X allows in contrast to a normal switch configuration no network traffic via the connected switch port, without prior authentication of the endpoint. Without authentication, a port is in the *unauthorized* state and accepts only 802.1X specific traffic. IEEE 802.1X uses the Extensible Authentication Protocol (EAP)[14] for communication and authentication. Related EAP traffic is being transferred from the switch to the Authentication Server (AS)[3] for verification. The Authentication Server sends the result of the authentication to the switch. Only in case of successful authentication the switch opens the port for arbitrary network traffic.

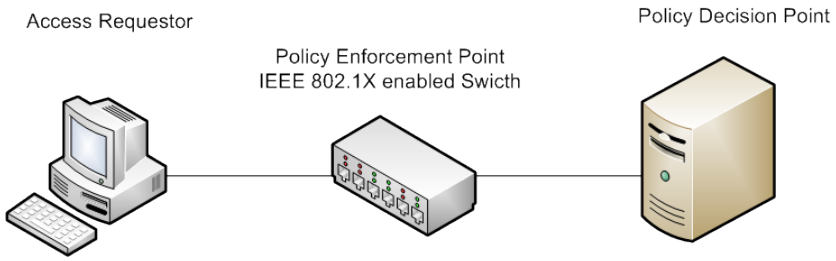

Figure 3.2: TNC and IEEE 802.1X

Trusted Network Connect defines an EAP-TNC method which can be used by IEEE 802.1X in a way that also a verification of the system configuration of the endpoint will be performed. For that purpose the client transmits in addition TNC messages to the authentication server. The switch-port is being opened only when the authentication according to IEEE 802.1X is successful and the integrity of the endpoint corresponds to a valid system configuration as well. In this case the 802.1X authentication server corresponds to the Policy Decision Point and the authenticator

[2] http://www.microsoft.com/windowsserver2008/en/us/nap-main.aspx
[3] Typically a RADIUS server is being used as authentication server.

to the Policy Enforcement Point of the TNC specification. Figure 3.2 illustrates the use of TNC in an IEEE 802.1X environment.

3.3 Network Access Layer

The Network Access Layer is the lowest of the three layers and is responsible for the transport of the messages on network level. The TNC Work Group has currently two different protocol bindings specified on this level each with an own focus. Both bindings will be introduced in the following sections.

3.3.1 IF-PEP

IF-PEP is responsible for the communication between the Policy Decision Point and the Policy Enforcement Point. In case of an invalid system configuration, the PDP sends a respective isolation recommendation for the endpoint to the PEP by using IF-PEP. As a result, IF-PEP must support the transport of all parameters regarding the isolation strategies considered by TNC (see section 3.2.3).

Furthermore the communication between the PEP and the PDP has to be protected, so that at least authenticity of the transmitted messages is warranted. If an attacker can interfere in the communication between PDP and PEP he will be able to send directly a kind of “Access Accepted” message to the PEP. In this case the attacker must just spoof the IP address of the PDP.

As already mentioned in section 3.2.4, TNC supports several authentication services and therefore has to be able to implement several protocols (like e.g. RADIUS) via IF-PEP. For this reason TNC defines special protocol bindings where the corresponding mapping is defined. A detailed introduction to IF-PEP and especially to the protocol binding for RADIUS is available in the corresponding specification under [15].

3.3.2 IF-T

The IF-T protocol is responsible for the transport of the TNCCS messages. Similar to IF-PEP, IF-T considers several protocol bindings which are responsible for the actual transport of the messages. At the moment there are two bindings specified for IF-T as introduced in the following sections. Understanding these protocol bindings is important for the development of a concept regarding the integration of TNC and VPN in the subsequent chapters of this thesis.

3.3.2.1 IF-T Binding to Tunneled EAP Methods

The TNC Work Group first specified a binding for the support of the Extensible Authentication Protocol (EAP). EAP is a generic protocol for the purpose of authentication of users and is being supported by many other protocols for authentication. An essential characteristic of EAP is the modular structure. EAP itself predefines no authentication method, but accepts several methods due to a generic design. Further information on EAP and the modular structure is available in RFC 3748[14].

EAP supports several methods for protecting the authentication against common attacks. EAP supports so-called tunneled EAP Methods, whereas EAP packets are being transmitted through an encrypted tunnel (e.g. TLS). For this reason the binding is called *IF-T Binding to Tunneled EAP Methods* since for security reasons only tunneled EAP methods are being supported.

Applying EAP enables the integration of TNC in many other protocols which offer support for EAP. For example, chapter 5 analyzes a possible integration of TNC and VPN on basis of IKEv2 (see chapter 2.3.1) and EAP. However EAP has also some drawbacks when the endpoint has already access to the network and the Internet Protocol (IP) can be used. The IF-T Binding to Tunneled EAP Methods is specified in [16].

3.3.2.2 Binding to TLS

The TNC Work Group released recently a binding based on TCP and TLS for the message transport. That binding should serve the advantages of an already existing network access where the Internet Protocol can be used for transmitting messages. The specification of *IF-T binding to TLS* explicitly suggests using EAP for the initial assessment and TLS for all further ones in order to benefit from the advantages of a TCP connection. Using TLS for the initial assessment is discouraged since there may be some additional risks when the endpoint has already access to the network.

The procedure of the IF-T binding is quite simple. The Access Requestor and the Policy Decision Point (more precisely the NAR and NAA components) first establish a TLS encrypted tunnel. In a second step some parameters like the supported version numbers will be negotiated before the IF-T messages will be transferred between both entities.

The TLS binding offers several advantages over the EAP binding as mentioned in the specification. The following enumeration of items is directly quoted without any changes from [17]:

- **Full Duplex connectivity** – can send multiple assessment messages prior to receiving a response including sending of asynchronous messages (e.g. alerts of posture changes)
- **High Bandwidth** – potentially much higher bandwidth than other transports (e.g. 802.1X) allowing more in-band data (e.g. remediation, verbose posture information)
- **Reliability** – IF-T messages sent will not be lost in transit (acknowledged by underlying protocol)
- **In-order Delivery** – IF-T messages can be sent knowing they won't be received prior to earlier messages
- **Large Messages** – ability to send very large IF-M messages without directly fragmenting them (underlying carrier protocol may introduce fragmentation)
- **Bi-directional** – TNC Client and TNC Server can initiate an (re)assessment
- **Multiple Roundtrips** – TNC Client and TNC Server can exchange numerous messages without fear of infrastructure timeouts. However, the entire exchange should be kept as brief as possible in case the user has to wait for its completion.

The TLS binding seems to be an interesting alternative to EAP regarding the aspired integration since the Internet Protocol is usually available if a connection shall be established with a VPN. Unfortunately, there is currently no free and open implementation of that specification available. The IF-T binding to TLS is specified in [17].

4 Requirements Analysis

The preceding chapters explained the fundamentals of the technologies Trusted Network Connect and Virtual Private Networks, each considered as a stand-alone solution. This chapter builds the bridge between TNC and VPN and defines several requirements which must or at least should be fulfilled by any integration approach of both technologies.

Maybe some of the introduced requirements can not be achieved because of technical restrictions or limitations of concrete implementations. However, if they are relevant and provide an added value for the integration they will be mentioned in this chapter. In order to increase the readability of the current and the following chapters, the result of an integration of Trusted Network Connect and Virtual Private Networks is shortly referred to as *TNC-VPN*. At the end of this chapter a table summarizes all requirements.

4.1 General Requirements

There are many general requirements concerning maintenance, security and configuration which shall be met by a TNC-VPN integration. Especially, the security requirements are of great importance since if it is easy for an attacker to circumvent these protection mechanisms the increased security is only fictitious. The following sections give an overview about these requirements.

4.1.1 Maintenance

The effort for setting up an initial TNC-VPN environment from scratch as well as the continuous effort for administrating and updating the environment must be kept at low level. If the effort for configuration and operation of an TNC-VPN environment is too complex, it will consume to much time and thus increases the probability for configuration mistakes. Section 4.1.4 introduces a central facility which can be also deployed in order to decrease the effort for maintaining complex or large environments.

Requirement: The complexity of maintaining a TNC-VPN environment should be remain reasonable.

4.1.2 Stability

As already mentioned VPNs are commonly used to connect one or more business units in order to realize a distributed LAN over several offices or offering access to the network from the outside. If the VPN is not available (e.g. caused by a software crash) the employees may not be able to work properly. For that reason, the stability represents a very important matter of any crucial application, especially like a VPN solution.

Independently of how the integration will be done, the stability must not be jeopardized. As a result of decreasing stability the availability of the VPN is in danger. Operations which endanger the stability such as manipulating the source code should be avoided whenever possible. Section 4.1.3.1 and 4.3.3 introduce further reasons why manipulating the source code of a VPN solution is discouraged.

Requirement: The stability of the deployed VPN software must not be jeopardized by a TNC-VPN integration.

4.1.3 Security

Besides maintenance and stability, the security is another important matter of any security related technology and is therefore an absolutely mandatory requirement which should gain particular attention. The following sections are covering basic requirements regarding security and introduce some typical security threats to a TNC-VPN environment.

4.1.3.1 Updating vulnerable applications

This section considers a TNC-VPN integration by adding TNC capabilities to an existing VPN software such as OpenVPN due to adapting the VPN software.

Updating vulnerable software is a crucial process regarding the overall system security. As far as a security vulnerability is found, the administrator is encouraged to install an update as soon as possible. For example, if the VPN gateway is vulnerable to remote attacks, an update of the VPN software is absolutely essential. In best cases, the VPN gateway is just vulnerable to Denial of Service attacks[1], but in worst cases an attacker gains full access to the protected network behind the gateway.

[1] More information regarding Denial of Service attacks can be found at http://www.cert.org/tech_tips/denial_of_service.html

Most Linux or BSD operating systems provide a packet manager which allows performing fully automated updates not only of operating system related components but also of third party applications like VPN software. That is one reason why most system administrators prefer using software available in the repositories of the packet manager instead of downloading and installing them manually (although often newer versions are available).

The most complicated case for an administrator is manually patching and compiling an application in order to get the desired functionalities. That means, in case of a security vulnerability, the administrator must download the latest version of the source code and furthermore needs also a corresponding patch for that specific version. In certain cases the latest available patch will not work for the newest version of the software. That is a pitfall because the administrator must now decide whether he still wants to use the vulnerable software until a corresponding patch is available or use the software without extended functionality provided by the patch. However, the administrator can backport the security fix to the actual deployed version but in general this is not possible.

As a result of still using the vulnerable software the entire security of the protected network may be in danger and should be avoided when security is a crucial requirement. As already mentioned above, not every security vulnerability results in access to the protected network but rather endangers the availability of the VPN (Denial of Service attacks). In the second case where the software gets updated without applying the third party patch there may be some fundamental functionalities missing and therefore the integrated VPN gateway may not be useful anymore.

Requirement: These situations should be prevented whenever possible by avoiding manual patching of the VPN software. As a result, updating the deployed software in the TNC-VPN environment must be possible.

4.1.3.2 IP-Spoofing to bypass TNC Assessment

There is another attack which can be used to bybass the isolation enforcement. In order to perform this attack, an attacker needs two different hosts, for example host *A* and host *B*. The system configuration of host *A* meets the requirements of the policy database and will be used to pass the integrity verification. Host *B* offers a system configuration which is explicitly forbidden (e.g. installed network sniffer or other security tools) and is used by the attacker to attack further systems.

In the first part, the attacker is using host *A* for establishing a VPN connection and passing the TNC assessment. After that the attacker uses host *B*, changes the IP address of host *B* to the address of host *A* and only authenticates at the VPN. Of course, the TNC assessment will be skipped since it would fail for that host. Now,

the attacker will gain access to the VPN with a potentially dangerous endpoint if the TNC-VPN integration provides no protection against IP spoofing.

Requirement: Bypassing the TNC assessment by an attacker due to spoofing the IP address of an already verified endpoint must be securely prevented by the integration concept.

4.1.3.3 IP-Spoofing to break out of VLAN Jail

TNC supports VLAN based isolation for those endpoints failing the TNC assessment (see section 4.2.5). A VLAN can be used to jail certain hosts into a separated network area. VLANs can be implemented on different layers of the OSI reference model. More information regarding VLANs and VLAN security is available at [18].

In a typical VPN scenario VLANs are implemented on layer three, since lower layers are not available over internet connections[2]. For that reason VLANs are realized by using the local client IP address available through the VPN.

Now, the attacker must only know one IP address of a successfully verified endpoint and spoofs that address in order to gain access to the VPN. This attack is very similar to that one described above in section 4.1.3.2. For that reason it is of great importance that the concept defeats attacks based on IP spoofing.

Requirement: Breaking out of a VLAN jail due to spoofing the IP address of an already authenticated endpoint must be securely prevented by the integration concept.

4.1.3.4 Rogue Access Requestor

The concept of using rogue hosts such as rogue WLAN Access Points (AP) or a HUB in order to circumvent network access control solutions is well known by security engineers. In an IEEE 802.1X based NAC architecture, an attacker is connected to the closed switch port and deploys an access point on his machine and awaits the first client request. As far as the first regular client starts with the 802.1X authentication, the *Rogue Access Requestor* (the attackers endpoint) just forwards the EAP traffic to the switch. After the port authentication has been finished successfully, the switch port is opened and the attacker has access to the protected network. This kind of attack is not only limited to IEEE 802.1X but will also apply for architectures where IEEE 802.1X is deployed in conjunction with Trusted Network Connect (see chapter 3.2.4).

[2]Some VPN technologies support forwarding of Ethernet frames through the VPN such as broadcasts, ARP etc.). However, in order to reduce network traffic this feature is disabled by default.

Requirement: The integration concept must consider and analyze attacks based on Rogue Access Requestors and whether they also apply to a TNC-VPN environment.

4.1.3.5 Bad VPN Gateways

This attack is quite similar to the *Rogue Access Requestor*. However in this case, a policy conform endpoint will be used in order to connect malicious endpoints with the network. The attacker is considered as a Road Warrior by the VPN gateway and as a consequence, the gateway expects only one IP address for packets from that endpoint.

The concept must prevent that the attacker uses a successful verified endpoint as a gateway for further malicious endpoints to the protected network (e.g. the attacker enables IP forwarding and Network Address Translation for that endpoint). Due to using Network Address Translation it becomes difficult to determine that the packets are sent by a malicious host behind the verified endpoint.

Requirement: An attacker must not be able using a policy conform endpoint as a gateway for malicious endpoints to the network.

4.1.4 Central Configuration Management

While the last sections have given an overview of common security threats regarding a TNC-VPN integration, this section covers a topic in order to reduce the effort for maintaining such an environment. In order to serve the requirements of enterprise environments or large networks the integration must consider any kind of facility which provides centralized configuration management. There are several use cases which will benefit from such a centralized repository:

VPN Authentication: The authentication credentials are stored on a central repository instead of storing them directly on the VPN gateway. This allows re-using these information in Single-Sign-On (SSO) architectures or on further VPN gateways (several VPN gateways increase the availability and scalability of large-scale VPNs).

IEEE 802.1X configuration: There are many RADIUS implementations available which support the use of a central configuration facility (e.g. FreeRADIUS). In that case, the configuration required for deploying IEEE 802.1X can be stored on a central repository. This is an important point when considering an integration of TNC-VPN in existing TNC-802.1X environments (as covered by section 4.4).

VLAN Assignment: The information which VLAN will be used for certain clients is no longer stored directly on the Policy Decision Point but also on a central repository. This is an essential requirement if more than one single PDP is in use (e.g. in order to increase the availability). Without a centralized configuration facility two or more configurations must be maintained for each PDP. That process is not only extensive but also error-prone.

Allowed Endpoint Policies: Such a central facility can also be used to hold the policy information used by the TNC server in order to verify endpoint configurations.

One common and widely supported protocol implementing the above described use case is the Lightweight Directory Access Protocol (LDAP)[19]. LDAP is the tool of choice when it comes to authentication and central configuration management and is widely supported by many systems.

For example, there is an additional authentication plugin available which extends OpenVPN by the support for a LDAP directory server[3]. Furthermore LDAP is also supported as configuration backend by many RADIUS servers. That point becomes important when considering an integration in existing IEEE 802.1X environments.

Requirement: A central facility such as a LDAP server should be considered by the TNC-VPN concept as configuration and authentication backend in order to meet the requirements of enterprise environments.

4.2 TNC Requirements

The following sections present some general requirements regarding TNC components and transport protocols which must be fulfilled by a TNC-VPN integration.

4.2.1 Transport Protocol (IF-T)

As already mentioned in chapter 3.3.2, there are currently two bindings specified for the transport protocol (IF-T). The usage of the right protocol is of great importance to guarantee good scalability and performance especially for large-scale environments. However, both protocols were designed with different use cases in mind. The advantages of using EAP or TLS for the message transport in a TNC-VPN environment are shown below:

Advantages using EAP: The Extensible Authentication Protocol is widely supported by many applications and other protocols. EAP can also be used when

[3] http://code.google.com/p/openvpn-auth-ldap/

IP is not available since EAP does not rely on the Internet Protocol and therefore it can be used in IEEE 802.1X architectures. As mentioned in chapter 2.3.1.1 version 2 of the IKE protocol even defines an integrated support for EAP and may represent an approach for the integration of both technologies.

Advantages using TLS: Using the binding to TLS offers very good quality of service features since it is directly placed on top of a TLS encrypted TCP connection. Chapter 3.3.2.2 introduced the advantages of using the TLS protocol instead of EAP for the TNCCS message exchange. Furthermore, using an existing TLS session of another application is considered in the IF-T TLS specification and may offer additional advantages when considering OpenVPN as VPN solution.

IPsec and OpenVPN based VPNs are realized on top of IP and thus the Internet Protocol can be used by the endpoint and the advantages of the binding to TLS come to play. In case of IEEE 802.1X, IP is not available for the initial assessment.

Requirement: If the concept does not consider the integration method based on IKEv2 and EAP, the concept should consider using the IF-T binding to TLS as message transport protocol.

4.2.2 IF-PEP

As mentioned in chapter 3.3.1 the purpose of IF-PEP is handling the communication between the Policy Decision Point and the Policy Enforcement Point. At the moment there is only the IF-PEP binding to RADIUS specified by the TNC Work Group. RADIUS is a very powerful protocol and considered by many other systems and protocols since it offers many useful features. For example, in IEEE 802.1X environments RADIUS can be used for authentication as well as for VLAN assignment when an endpoint fails the integrity verification.

Requirement: The IF-PEP binding to RADIUS should be considered by the integration concept since it is the only specified binding for IF-PEP at the moment.

4.2.3 Policy Enforcement Point

The aim of the Policy Enforcement Point is to ensure that only those endpoints allowed by the Policy Decision Point can access the protected network, however with one exception. In case of an unauthenticated endpoint the PEP must enable that the Access Requestor can communicate with the PDP in order to perform the TNC handshake while blocking all other network access.

In case of an integrated TNC-VPN architecture the Policy Enforcement Point is responsible for applying an isolation recommendation for valid VPN clients when they failed the TNC assessment. Every Road Warrior or remote VPN gateway use this machine as entry point to the network. If the PEP is deployed on any device behind the VPN gateway, enforcement of isolation recommendation is much more difficult to achieve. For that reason, housing the PEP and VPN gateway on the same machine will simplify that task.

4.2.3.1 Supporting multiple Isolation Strategies

The TNC specifications define several isolation levels as already described in chapter 3.2.3. As a result to a TNC-VPN environment, the VPN gateway must be capable of enforcing different TNC isolation recommendations. The PEP must support at least one of these isolation levels in order to be specification compliant[15]. However in order to meet the requirements of more complex environments, the concept should consider several isolation strategies. Section 4.2.5 defines the requirements regarding the isolation levels.

Requirement: Multiple isolation strategies shall be supported by the integration concept.

4.2.3.2 Enforcement of Reassessments

Reassessments are used to recheck endpoints which have already passed the platform authentication at regular intervals. This is a necessary requirement since the integrity of an endpoint may change over time while the system is up and running (e.g. a system gets infected by some kind of malware after the initial integrity verification). Not only server systems with long uptimes are candidates for reassessments but also usual workstations.

Requirement: In order to detect malicious endpoints within the VPN as far as possible, support for reassessments is a mandatory requirement and must be covered by the concept.

4.2.4 Policy Decision Point

The Policy Decision Point is the core of any TNC solution. Most of the functionalities provided by the Policy Decision Point are not specific to a IEEE 802.1X or VPN integration, since the whole TNC architecture is divided into three different layers each with its own focus. However, the integration concept must consider potentially changes to the Network Access Layer (NAL) as well as to the Integrity Evaluation Layer (IEL).

4.2.5 Isolation and Remediation

Trusted Network Connect supports three isolation levels at the moment as defined by [15]. The binary-based isolation where access to the protected network is either granted or denied can be seen as the most stringent type of isolation. In fact, this isolation level protects the network from potentially dangerous endpoints but it is too inflexible to be used in many enterprise environments.

One advantage of providing VLAN or filter-based isolation is that endpoints get partial access to only certain network resources. This can be useful for two scenarios.

- Granting access to less critical systems by VPN clients regardless whether they passed the TNC assessment or not.
- Giving an endpoint the chance of remediation when it failed the TNC assessment. However, granting partial access to the network is actually not required for remediation, since the client has already an existing internet connection for updating the system.

Requirement: Besides binary-based isolation, at least VLAN or filter-based isolation shall be covered by the TNC-VPN integration concept.

4.3 VPN Requirements

The following sections describe some general requirements regarding the VPN part of a TNC-VPN integration.

4.3.1 VPN Authentication

The VPN authentication process should be kept separate from the TNC authentication and platform integrity verification. That increases the security if one of both authentication operations is vulnerable against a security flaw. At least the authentication process of the well known VPN solutions such as OpenVPN or IPsec can be seen as secure and stable. Manipulating these algorithm can decrease the entire security of the VPN gateway.

Requirement: The TNC assessment should not be directly integrated into the VPN authentication process since that may jeopardize the security and reliability of the VPN authentication process.

4.3.2 Generic Integration Approach

In practice, there are several VPN technologies available and many different implementations are used in production environments. Many of these network environments are in use over several years and changing them to another VPN software is a very complex process which will be avoided by most administrators. Therefore, developing an integration concept which considers many different VPN implementations is absolutely mandatory in order to support as many production environments as possible.

In order to provide an integration solution which is well accepted and future-proof there are at least three conditions which must be covered by the concept:

Supporting different technologies: Only supporting one VPN technology is not sufficient, since there are many different technologies used in practice. IPsec and OpenVPN are very popular solutions and therefore the concept should be realizable with IPsec as well as with OpenVPN based VPNs.

Supporting different implementations: IPsec is a standardized protocol suite with many implementations available on the market. For that reason, the concept should not only consider multiple VPN technologies but also many different implementations of one technology in order to increase acceptance.

Supporting proprietary implementations: The concept should also be realizable with proprietary VPN solutions, although supporting proprietary software is not a major requirement.

Requirement: Developing an integration concept tailored for exactly one VPN implementation is not convenient and should be avoided. Instead of this, the concept should consider a generic approach which works with many different VPN technologies and implementations. As a result, the effort of implementing the concept for certain VPN implementations should not be too complex.

4.3.3 No Source Code Manipulation

As mentioned in section 4.1.3.1, it is highly recommended to not manipulate the source code of any VPN implementation directly. This results not only in trouble when it comes to updating the affected applications but furthermore has a negative effect regarding the general security and scalability of the application.

Unfortunately, not every VPN implementation provides a plugin or extension mechanism where the developer gets a well defined interface to extend the VPN software. Furthermore, if such a plugin facility is provided by the software it must

get analyzed if it offers all required functions which are necessary in order to integrate Trusted Network Connect functionalities.

Requirement: Manipulating the source code directly without using any kind of plugin facility is absolutely discouraged and must be avoided.

4.3.4 Support of Site-to-Site VPNs

Site-to-Site VPNs are widely used to connected distributed network segments through a VPN gateway on each side (see chapter 2.2.1). The VPN gateways are responsible for encapsulating and encrypting the packets before sending them to the opposite VPN gateway. The respective gateway then decapsulates and decrypts the packets and forwards them to the specified destination.

The whole process is completely transparent so that endpoints do not get notice of the physical detachment of both networks. Exactly this behavior makes it very difficult to hold the requirement that only verified endpoints are allowed to access the network, since only the VPN gateways get verified but not the endpoints behind that VPN gateway.

Requirement: The integration concept should analyze and consider how to support and integrate Site-to-Site VPNs securely.

4.4 Integration in existing IEEE 802.1X Architectures

Another major requirement regarding the TNC-VPN integration concept is the integration in existing TNC environments based on IEEE 802.1X. In practice, setting up a TNC-VPN-only environment may rarely occur because an existing environment will rather be extended in order to enforce TNC assessment for VPN clients.

Requirement: The concept must cover an integration of TNC-VPN in an existing TNC-802.1X environment.

4.5 Catalog of Requirements

Category	Requirement	Relevance
General Requirements		
	Maintenance	Important
	Stability	Important
	Central Configuration Management	Important
Security		
	Updating vulnerable Applications	Mandatory
	IP-Spoofing to bypass TNC Assessment	Mandatory
	IP-Spoofing to break out of VLAN Jail	Mandatory
	Rogue Access Requestor	Important
	Bad VPN Gateway	Important
TNC		
	IF-T binding to TLS	Recommended
	IF-PEP binding to RADIUS	Recommended
	Enforcement of Reassessments	Mandatory
	Binary-based Isolation	Mandatory
	VLAN- or filter-based Isolation	Important
VPN		
	Secure Authentication	Mandatory
	Generic Integration Approach	Important
	No Source Code Manipulation	Mandatory
	Supporting Road-Warrior VPNs	Mandatory
	Supporting Site-to-Site VPNs	Important
Integration in existing Environments		
	TNC - IEEE 802.1X	Recommended

Recommended:	Support would be good although it is not required.
Important:	Support is important for an integration, however not absolutely necessary.
Mandatory:	Support is mandatory, not fulfilling these requirements result in a K.O. criteria.

Table 4.1: TNC-VPN integration requirements summary

5 Evaluating Integration Methods

There are several methods available of how to integrate Trusted Network Connect and Virtual Private Networks. For that reason, it is necessary to shortly introduce and evaluate these different integration methods before start developing the concept. Restricting to the essential aspects, five methods remain which will be presented in this chapter. The result of this evaluation provides the basis for the next chapter.

5.1 Method 1: TNC starts VPN

This approach divides the steps of establishing a VPN tunnel and the TNC integrity verification in two independent phases. In the first phase the Access Requestor communicates with the Policy Decision Point and performs the TNC assessment. For example, this can be done by using the new IF-T binding to TLS. In that phase, no VPN related data transfer is included. In case of a successful integrity verification reported by the PDP the Access Requestor starts with the VPN phase. In that phase, the VPN client just establishes a usual VPN connection to the VPN gateway. That procedure offers the major advantage, that there is no need to adapt or extend the VPN or TNC software.

Due to the fact that the VPN gateway does not know anything about TNC, the attacker can quite easily access the network as it starts the VPN client directly without performing the TNC phase. Since the VPN gateway is not modified and thus not capable of verifying whether the platform authentication for that endpoint has been performed and with which result, an attacker can easily gain access to the network.

The solution for preventing access without doing the TNC assessment is the use of a packet filter on the VPN gateway. The packet filter assures that only network traffic of successful verified endpoints can enter the network. For that reason, the PDP is responsible for manipulating (either directly or by sending a corresponding message to the PEP) the packet filter rules.

The packet filter can also be used to realize the requirements regarding isolation strategies, since binary-based and filter-based isolation can be quite easily realized by using a packet filter.

Evaluation

This trivial scenario requires no adaptions of the VPN software and functions therefore as well with proprietary VPN technologies and fulfills all requirements regarding the VPN and TNC part of the requirements analysis. But unfortunately, this scenario has one major shortcoming. It does not protect the network from attacks based on IP spoofing as mentioned by the requirements analysis at section 4.1.3.2. That way, an attacker can e.g. use a valid endpoint to generate appropriate packet filter rules for that client IP address and uses afterwards another (not verified) endpoint with the same IP address and connects itself to the VPN. According to the time interval between the reassessments, the attacker can access the network unhindered.

The PDP and the VPN gateway have to be able to verify reliably whether the platform authentication as well as the VPN authentication have been performed for the same endpoint. Section 5.4 describes a quite similar approach where solely the order of the authentications is different.

5.2 Method 2: TNC is integrated in VPN Client and Gateway

Method 2 represents a strongly coupled integration of both technologies. The entire TNC assessment is being directly integrated in the VPN software. As a result, the IF-T message exchange will be done by the VPN client and gateway as a part of the authentication process (in addition to the regular VPN authentication). That procedure offers the advantage that an attacker will not be able to bypass the TNC assessment, since it is directly integrated in the VPN software and a part of the entire authentication process.

But only integrating the TNC assessment into the VPN client and gateway is not sufficient to meet the requirement exposed in section 4.2.3. For example, the VPN gateway has to verify continuously the time of the last integrity measurement (reassessment) and initiate a new assessment if required. As a result, the VPN software must be either capable to handle reassessment by itself or it must contact another instance like a PDP. Another important requirement is the support of at least two isolation strategies such as binary-based as well as VPN- or filter-based isolation. VLAN- or filter-based isolation becomes important for environments where access to certain systems or services should granted, although the endpoint has not passed the assessment. For being able to implement all these requirements, it is necessary to modify the VPN software deeply.

Evaluation

Unfortunately, the direct integration into the VPN implementation has several grave disadvantages as already exposed by the requirements analysis. There is a continually high developing effort for each supported VPN implementation and furthermore the future user has to patch and compile the software manually. Due to patching the VPN implementation, restrictions in regard to security, stability or performance may occur and the gained security is in danger. The sections 4.1.3.1 and 4.3.3 of the requirements analysis explain the resulting problems in detail.

In addition, such a patch has to be generated respectively for each VPN implementation which shall be supported and in case of proprietary systems this is not possible. However, lots of VPN implementations indeed provide a plugin concept but these ones are often not designed for an enhancement in such a dimension. For example, OpenVPN plugins offer no mechanism for transmitting data from the client to the server but only offering options for extending the VPN server.

This integration method results in many disadvantages so that it will not be considered anymore for developing the concept.

5.3 Method 3: TNC is only integrated in VPN Gateway

The third integration method can be seen as a combination of the first two methods. The integration is divided into a TNC and VPN phase very similar to method 1. The significant difference is that the VPN gateway gets extended only in that way, that it contacts the PDP in order to find out whether the platform authentication for the enquiring endpoint has been already performed and with which result. That procedure ensures that an attacker can not bypass the TNC assessment just by starting a VPN client.

The VPN gateway has to detect reliably whether the VPN authentication and the TNC platform authentication has been performed by the same endpoint to defeat attacks based on IP spoofing as described in section 4.1.3.2. Furthermore, the extended VPN gateway has to contact the PDP regularly in order to find out the time of the last assessments and if necessary disconnect the VPN tunnel for that endpoint to meet the requirements of section 4.2.3.

The TNC integration takes place only in the VPN server (e.g. via a plugin) and the VPN client remains untouched. For that reason, a direct manipulation of the mainline source code is not required.

Evaluation

Using this method for integrating TNC and VPNs offers the advantage that the TNC assessment is part of the VPN authentication but does not require extensive manipulation of the VPN software since it can be realized by using a plugin.

Due to the fact that no TNC messages between VPN client and gateway has to be exchanged, a server-side plugin mechanism of the VPN software can be used for that purpose. Unfortunately, without using an appropriate plugin facility, this integration approach has the same disadvantages as the second method.

5.4 Method 4: VPN starts TNC

The fourth method is quite similar to the first one since the process is divided into two phases and no adaptions to the VPN software are required. However, the significant difference compared to the first method is the order of authentications. In this integration method the VPN authentication takes place first before the TNC platform authentication is being performed. Similar to the first method, a packet filter is deployed on the VPN gateway to ensure that an endpoint can not access the network until the TNC assessment finishes successfully.

Due to this fact, the TNC connection between Access Requestor and Policy Decision Point can be done through the right before established VPN tunnel which offers several advantages. For example, method 1 requires that the Policy Decision Point is accessible from the outside since the VPN connection will be established not until the TNC handshake has finished. Furthermore, some VPN implementations such as strongswan or OpenVPN can reliably prevent IP spoofing of the VPN clients and therefore all network traffic which is sent through the VPN tunnel is also protected.

Evaluation

This method is a general integration approach which functions for almost every VPN software (like the first variant). No source code for a certain VPN software has to be written or adapted and is therefore also possible in case of proprietary VPN implementations. In addition, this approach offers a high flexibility, since e.g. packet filter requirements can be simply added or adapted by the administrator.

The shortcoming of the first method was that an attacker was able to start the VPN client directly, spoof the IP address of a valid endpoint and therefore gain access to the network. However, in this case an attacker can not simply change his IP address since some VPN technologies such as strongswan or OpenVPN can detect when the VPN client sends packets with a different IP address.

5.5 Method 5: Integration based on IKEv2 and EAP

An integration based on IKEv2 and EAP sounds to be a promising approach for the first thought. The major advantage of an integration based on EAP would be the seamless integration in existing IEEE 802.1X environments. Furthermore there is no manipulation of the VPN software necessary and should work with all IKEv2 enabled IPsec implementations.

However, using EAP in order to integrate both technologies has one essential disadvantage. The specification of IKEv2 allows only the usage of one authentication method, but not using multiple authentication methods one after the other as it would be required for this integration method. One authentication method is required to authenticate the VPN client and the second one is responsible for the actual TNC assessment. However, RFC 4739[20] extends IKEv2 by the ability of handling multiple authentication exchanges.

A detailed description of an integration based on IPsec VPNs and TNC by using EAP was done by Gérard Pelka at his Bachelor thesis[21]. His thesis covers an integration based on *TNC@FHH*, *strongswan* and *FreeRADIUS*[1]

Evaluation

Considering an integration based on IKEv2 and EAP for developing the concept has many downsides which result from practical limits. IKEv2 is sparely supported by most of the IPsec implementations yet and moreover, only supporting IKEv2 is insufficient since also the extension for handling multiple authentication exchanges is necessary in order to integrate both technologies. Integrating other VPN technologies such as OpenVPN is almost impossible. As a consequence, this integration method cannot be seen as a generic integration approach as suggested by section 4.3.2.

[1] http://freeradius.org/

5.6 Evaluation of Integration Methods

The evaluation of the above presented methods should be done on the basis of the mandatory requirements mentioned by the catalog of requirements at table 4.1. Not fulfilling these requirements can be seen as a K.O. criteria.

Method 1 fulfills most of the mandatory requirements. Updating the software without manually patching and compiling is possible since neither the TNC nor the VPN software will be extended. For the same reason, there is almost every VPN technology and implementation supported. Method 1 represents therefore a generic integration approach and the stability of the VPN software is not affected. However, it does not protect the network against attacks based on IP spoofing and thus cannot fulfill two mandatory requirements (IP spoofing prevention).

An integration based on method 2 will be not considered any longer since it can not fulfill two mandatory requirements. Due to the necessary manipulation of the VPN client and server source code, security updates require patching and compiling the affected application and the stability may also be jeopardized. Furthermore, this is not a generic approach and many VPN implementations will be out of scope.

Method 3 is similar to method 2 but does not require a modification of the VPN client and thus a plugin facility can be used. However, the development of plugins represents also no generic approach and may also jeopardize the stability. However, installing security updates will be possible since the source code will not be modified. Although method 3 can detect IP spoofing and therefore fulfills all mandatory requirements, there is still the grave disadvantage that only certain VPN applications can be used and an appropriate plugin must be developed for each implementation anew.

Method 4 offers the same advantages like method 1 but can further fulfill all mentioned security requirements and has therefore no disadvantages.

Method 5 also fulfills all mandatory requirements and does also not require the development of any source code related to the VPN implementation. But unfortunately, its use is only applicable to IPsec implementations which supports IKEv2 as well as the extension which allows multiple EAP authentication exchanges. As a result, the integration method does not represent a generic approach.

Table 5.1 presents a summarized overview of the five integration methods containing the major properties regarding each integration method.

Property	**1**	**2**	**3**	**4**	**5**
Extending VPN client is required	N	Y	N	N	N
Extending VPN server is required	N	Y	Y	N	N
High development effort and complexity	N	Y	Y	N	N
Adaptions are required for each VPN implementation anew	N	Y	Y	N	N
Supports many VPN implementations out of the box	Y	N	N	Y	N
Support of proprietary VPN solutions	Y	N	N	Y	Y
PDP muss be accessible by the outside	Y	N	N	N	N
TNC assessment is part of the VPN authentication	N	Y	Y	N	Y
TNC assessment can be bypassed by an attacker	Y	N	N	N	N

Y: Yes

N: No

Table 5.1: Properties of the integration methods

After comparing the five introduced and evaluated methods, the *VPN starts TNC* approach (method 4) offers the most advantages including development effort, maintenance, flexibility and security. Method 5 offers also many advantages however due to the limitation on IPsec and the practical limits it will not be considered by the concept.

For that reason, the integration concept presented in the next chapter is based on the concepts introduced by method 4

6 Integration Concept

The last chapter has introduced and evaluated several integration methods in order to find out the approach which fulfills most of the requirements and should therefore be used as basis for the integration concept. Furthermore, the requirements exposed by chapter 4 will be used as some kind of guideline within this chapter.

The integration concept is developed step by step and when adequate refers to a small example scenario for a better illustration. For that reason, before introducing the conceptual details the example scenario will be presented.

6.1 Example Scenario

In the example scenario a sales-man needs access to a web-application which is only available within the internal network of the company. In order to get access to the network from the outside, the sales-man[1] must pass the VPN authentication as well as the TNC assessment.

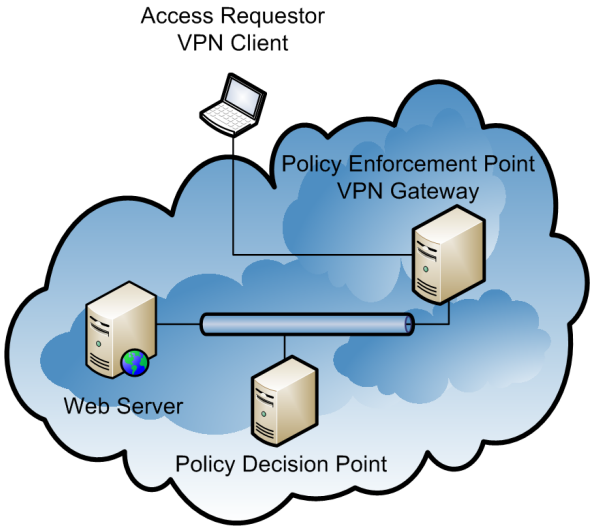

Figure 6.1: Example scenario

[1]The sales-man equates to the Access Requestor of the TNC architecture.

The Policy Decision Point is located in the internal network and not accessible from the outside. As a result, the sales-man must first establish a VPN tunnel to the VPN gateway in order to start with the TNC assessment in the second phase. That procedure follows the selected integration method as presented in the last chapter. Figure 6.1 illustrates the example scenario.

The details of the overall authentication process and of the integration are described in the following sections. The example scenario is just used for a better demonstration of the individual authentication steps. However, before starting with introducing the TNC-VPN related entities and protocols the concept will first explain the communication flows within the architecture on the basis of the TNC phases.

6.2 TNC-VPN Phases

The TNC architecture specification[1] has defined three phases as described by the TNC chapter at 3.2.2. Each of these phases must be covered and implemented by the integration concept.

6.2.1 Assessment Phases

The concept follows the *VPN starts TNC* approach as described by chapter 5.4 since that approach fulfills most of the mentioned requirements out of the box. Besides other benefits of following that approach, the main advantage is that neither the VPN nor the TNC software needs to be extended by developing a certain plugin or even by manipulating the source code directly in order to bring both technologies in line.

The assessment of the endpoints is separated into two phases, each with its own technology and focus. Dividing the process of VPN authentication and verification of the system configuration into two completely independent processes has the advantage that none of both technologies need to get enhanced by functionalities of the other technology. For this reason many different VPN or TNC implementations will be supported. However, the PEP must absolutely ensure that connected VPN clients can not access the network until their system configuration was verified as to be valid, although they have already an established VPN tunnel. Both phases are described in detail in the following two sections.

6.2.1.1 Phase 1: VPN Authentication

The responsibility of the first phase is just establishing a secure VPN tunnel between the Access Requestor and the VPN gateway. This process is exactly the same as without using any NAC solution. For example, the OpenVPN server authenticates the client by using a certificate. Of course, the client should also authenticate the VPN server in order to prevent Man in the Middle attacks[2]. How the VPN connection establishment will be done in detail depends on the VPN technology and is not of interest for understanding how this concept works.

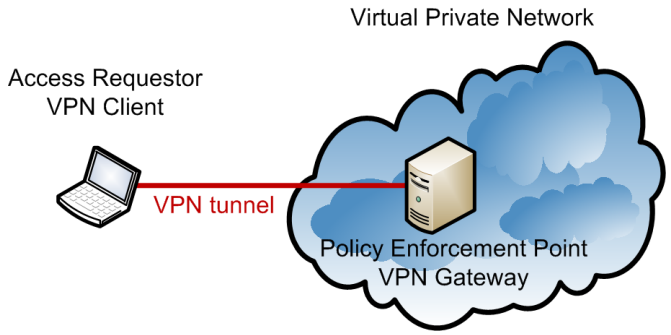

Figure 6.2: Phase 1: Establishment of the VPN tunnel

After phase 1 has finished successfully, the sales-man's notebook as mentioned by the example will now be able to communicate with the Policy Decision Point through the VPN tunnel. Without an established VPN tunnel the Access Requestor will not be able to communicate with any endpoint located in the protected network including the PDP (of course, the PDP can be deployed on a machine accessible by the outside but that example assumes that the PDP is only available within the local network).

However, the sales-man's notebook must not be able to communicate with other hosts but the PDP (such as the web server). This is achieved by deploying a packet filter on the PEP which only accepts packets destined for the PDP unless phase 2 finishes successfully.

6.2.1.2 Phase 2: Endpoint Assessment

The aim of phase 2 is measuring the system integrity of the endpoint which requests access to the network. In this phase the TNC client of the Access Requestor directly communicates with the Policy Decision Point via any IF-T protocol supported by TNC client and PDP.

[2]More information regarding SSL / TLS Man in the Middle attacks can be found at [2].

In the example scenario, the PDP verifies the integrity of the sales-man's notebook and sends an appropriate recommendation to the Policy Enforcement Point using IF-PEP. After that, the PEP adds or removes certain packet filter rules regarding the recommendation (e.g. access will be granted or denied).

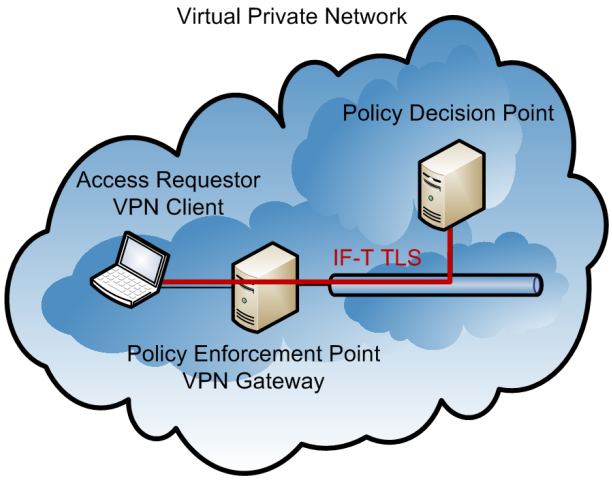

Figure 6.3: Phase 2: TNC assessment

In order to implement phase 2 there is only one additional component on the PEP required which receives and handles IF-PEP messages sent by the PDP. That component is described in section 6.3.2.3 in detail. The complete communication between Access Requestor, Policy Enforcement Point and Policy Decision Point including the respective processes is shown in listing 6.4.

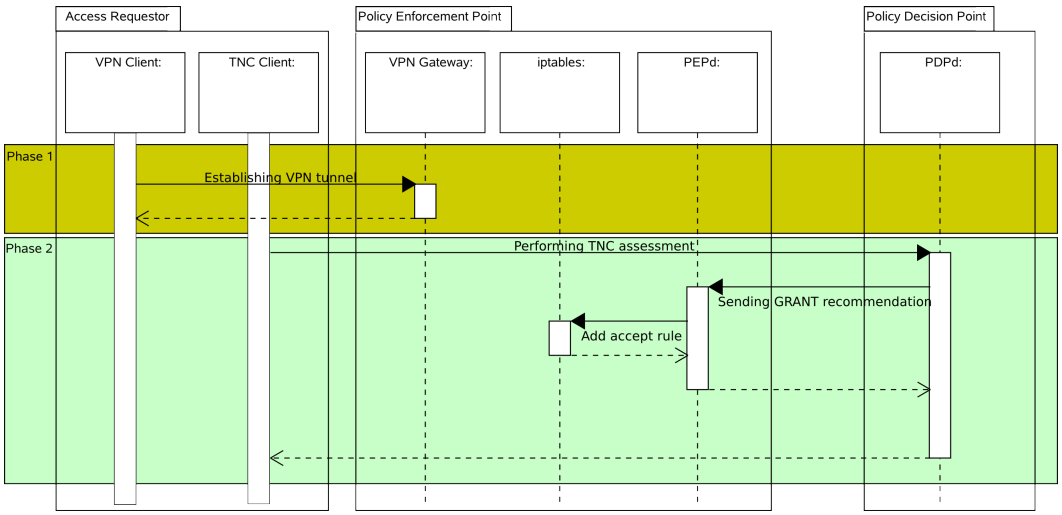

Figure 6.4: TNC-VPN assessment communication flow

However, if an endpoint fails the assessment phase the PDP will send an appropriate isolation recommendation to the PEP.

6.2.2 Isolation

As already explained, by default no access is granted to the protected network unless phase 2 finishes successfully. That prevents that an attacker gains access to the network only by establishing a VPN tunnel (e.g. the attacker holds valid VPN authentication data such as a signed certificate or a Pre-Shared Key).

This default behavior is very good from a security perspective since it represents a *deny-all* firewall strategy[3]. However, to meet the requirements given by the TNC specification (and also by many enterprise environments) the integration must support multiple isolation strategies. Which isolation strategies will be considered and how they will be realized by this concept is described in the following sections.

6.2.2.1 Binary-based Isolation

Binary-based isolation is the default isolation strategy used in this TNC-VPN integration. Either all requests (packets) will be dropped (access denied) or accepted (access granted) by the PEP depending on the endpoint's system configuration. The recommendation of the PDP creates only in case of an "access granted" recommendation additional rules for the packet filter because all packets will be dropped by default.

Listing 6.1 shows the required packet filter rule (using the syntax of the iptables command) in order to grant network access to the remote endpoint with the IP address *192.168.1.230*.

Listing 6.1: Granting access to the network using iptables

```
1 iptables -A FORWARD -s 192.168.1.230 -j ACCEPT
```

The deny rule presented at listing 6.2 is added in case of a reassessment. The job of that rule is removing a prior added allow rule for that endpoint.

Listing 6.2: Denying access to the network using iptables

```
1 iptables -D FORWARD -s 192.168.1.230 -j ACCEPT
```

[3]Deny-all firewall strategy means that every packet will be dropped by the packet filter unless it is explicitly allowed by a certain rule.

6.2.2.2 VLAN and Filter-based Isolation

VLAN-based as well as filter-based isolation offers the ability that endpoints which do not match the required policy get the chance for remediation. This requirement results from the use of TNC in conjunction with IEEE 802.1X. In that case when enforcing binary-based isolation, the endpoint has no chance updating installed software or anti-virus signatures because there is no access to the network granted.

Indeed, in a VPN scenario like in the example scenario above the sales-man has already access to the internet and can use that connection to update the system configuration according to the required policy. Nevertheless, there are further use cases than only accessing update servers and therefore the TNC-VPN integration concept will also introduce VLAN-based and filter-based isolation. Moreover, one of them will be implemented in addition to binary-based isolation.

VLAN-based isolation does not provide any further protection than filter-based isolation since it must also be implemented on layer three of the OSI reference model. VLANs realized on layer three or any other access control mechanism based on IP addresses may be insecure and therefore can be bypassed by an attacker due to using IP spoofing as already mentioned by the requirements analysis at section 4.1.3.3.

VLAN-based Isolation

As mentioned above, VLANs can be realized only on layer three because OpenVPN as well as IPsec establish a connection to the VPN gateway based on the Internet Protocol and thus lower layers are not available on the PEP. Although Linux offers support for VLANs and VLAN trunking based on IEEE 802.1Q[4][18] by default, implementing VLAN-based isolation on the Policy Enforcement Point is not that easy. The Access Requestor has already an IP address assigned which is used within the VPN and that IP address can not be mapped simply to a certain VLAN with a completely different subnet mask. That circumstance makes it difficult applying VLAN-based isolation to this integration concept.

However, when thinking of a TNC-VPN environment the typical use case of a VLAN might be granting access only to certain servers or even only to certain services (e.g. only FTP or SMB traffic to a certain file server will be allowed) within the VPN. Fortunately, that use case can be achieved by using filter-based isolation in a much simpler manner. For that reason, VLAN-based isolation will not be covered by this concept in favor of filter-based isolation. A good introduction on how to implement VLANs on Linux is given at [22].

[4]Most modern Linux distributions like Ubuntu 8.0.4 have built in support for VLANs and IEEE 802.1Q in their kernel.

Filter-based Isolation

The approach based on using packet filters is quite straightforward. Instead of assigning VLANs to an endpoint there are just some additional packet filter rules on the PEP required which grants access to certain servers or services without the need for passing the TNC assessment. Because of a *deny-all* firewall strategy there are special packet filter rules required for every resource in the network which should be accessible by remote VPN clients. However, this simple approach is only adequate when access to few resources shall be granted but nevertheless will apply for most environments.

Listing 6.3 shows appropriate iptables rules where access only to the endpoints *192.168.1.20* and *192.168.1.30* is granted. Furthermore only FTP traffic is allowed for host *192.168.1.20*.

Listing 6.3: Filter-based isolation with iptables

```
1 iptables -A FORWARD -d 192.168.1.20 -j ACCEPT
2 iptables -A FORWARD -d 192.168.1.30 -p tcp --dport 21 -j ACCEPT
```

These rules can be either defined as default enforcement behavior or only activated on a certain recommendation by the PDP. Default behavior means that an endpoint gains access to these network resources as far as phase 1 has finished successfully. If these rules should be applied only on a certain recommendation, the commands must be extended by the source IP address of the endpoint.

6.2.3 Remediation

The process of installing updates or performing some other kind of operation in order to meet the requirements regarding the system configuration is called remediation. As already explained by the section above, remediation is not that important for VPN clients as in case of local endpoints. Of course, this integration concept supports remediation and can be easily implemented by using filter-based isolation as described by section 6.2.2.2.

6.2.4 Handling Reassessments

Reassessments can be initiated by the Access Requestor or by the PDP. In general, whether the Access Requestor or the PDP initiates the reassessment does not make any difference for the concept. If noone initiates a reassessment within a specified time interval either the PEP or the PDP must enforce a new assessment by revoking access to the network for that endpoint.

Revoking Access to the Network

In order to revoke access to the network, the Policy Enforcement Point must either remove all rules from the packet filter which grants access for an endpoint or limit the access to certain resources. This depends on the isolation strategy (see 6.2.2) deployed in an environment. The task of determining whether the Access Requestor has performed a reassessment or not can be done by the PEP, by the PDP or by both. The next section covers the advantages and disadvantages of the different strategies.

Responsibility for Reassessments

In general, the Policy Decision Point contains all information regarding the endpoints including the timestamp of the last assessment and seems therefore the best place for determining whether a reassessment is missing by an endpoint. Unfortunately, from a security point of view it is not sufficient when only the PDP is capable of determining that.

Imagine an attacker who passes both assessment phases and gains full access to the VPN by using a proper endpoint. After that, he uses an exploit, gains full access to the endpoint and installs many tools prohibited by the security policy of the network. Normally, the PDP would detect these changes when performing the next (re)assessment. But what if the attacker is able to drive a Denial of Service (DoS) attack against the PDP? He can stay as long in the network as the responsible system administrator needs to redeploy the PDP.

In order to defeat these security threat, the PEP should also be capable of determining whether an endpoint must be locked out of the network. Unfortunately, locking endpoints out of the network when the PDP is not available can cause reasonable lack of work since employees can no longer access the network. Actually, this example demonstrates a common problem when it comes to a security related decision. What is more important to the business - security or availability? In order to serve high security requirements, the Policy Enforcement Point presented by section 6.3.2 of this thesis will be capable of determining the time of the last assessment and if necessary revoke access for that endpoint.

6.3 TNC-VPN Entities and Components

Section 3.2.1 of the TNC chapter has introduced the entities and components of the TNC architecture. This section covers the role of them within the TNC-VPN architecture.

6.3.1 Access Requestor

The Access Requestor does neither require any modifications regarding the TNC nor VPN client software. As a result, a combination of different TNC and VPN solutions will be possible as far as they are compatible to the deployed VPN and IF-T protocols. The implementation chapter presents in section 7.2.3 a small wrapper shell script in order to provide one single application for getting TNC-VPN access to the network.

6.3.2 Policy Enforcement Point

The Policy Enforcement Point can be seen as the critical component of this TNC-VPN integration concept. The PEP must ensure that only successfully verified endpoints are able to access protected network resources, although VPN traffic destined for the PDP is allowed. Since this concept does not demand any modifications regarding the VPN software, the PEP entity consists of three independent applications:

VPN Gateway: The VPN gateway is responsible for establishing the VPN tunnel between VPN Client and VPN gateway.

PEP Daemon: The PEP daemon (PEPd) is a special component which is responsible for processing IF-PEP messages by the PDP. This software will be developed in section 7.3.3 of the implementation chapter.

Packet Filter: The packet filter is responsible for granting or denying access to network resources depending on the isolation strategies and recommendations by the PDP.

In this concept the PEP daemon as well as the VPN gateway are housed on the same machine. Of course, other scenarios where the VPN gateway and the PEP are placed on different machine are also possible, but following this approach simplifies the architecture and focuses on the integration of TNC and VPN. The next sections introduces these three components in detail.

6.3.2.1 VPN Gateway

After several integration methods were evaluated in chapter 5, the concept follows a VPN technology independent approach. Among other implementations, strongswan and OpenVPN seems to be a good choice as VPN solution. The implementation of this integration concept considers using OpenVPN because the installation and configuration is quite simple and it also provides protection against IP spoofing.

6.3.2.2 Packet Filter

The task of the packet filter is ensuring that authenticated VPN clients can not access arbitrary network resources until their system configuration was declared to be valid. In case of binary-based isolation the packet filter drops all kind of network traffic unless one of the following two conditions is met:

- The traffic is adjusted to the VPN gateway. This kind of traffic deals with the authentication and other VPN related operations.
- The traffic source is an authenticated VPN client in the network and the destination is the address of the Policy Decision Point

The concept can be realized with any packet filter which is capable of accepting or dropping IP packets which should be forwarded to another host on the basis of IP addresses. The exemplary implementation considers using *iptables*[5]. A short introduction to iptables is available in section 7.3.2 of the implementation chapter.

6.3.2.3 Policy Enforcement Point daemon (PEPd)

The PEP daemon (PEPd) is small application deployed on the Policy Enforcement Point responsible for processing TNC recommendations sent by the PDP. The PEPd listens for incoming IF-PEP messages (of course, only on the internal interface) and depending on the recommendation of a certain IF-PEP message the PEPd adds or removes packet filter rules.

In order to defeat common security threats the communication between PEP and PDP will be done through a TLS tunnel whereas each endpoint authenticates the other one. Without authenticating at least the PDP, an attacker might be able to send counterfeit IF-PEP messages to the PEP by spoofing the IP address of the PDP and thus gains access to the network.

Reassessment Timer

The PEP does not need any complex logic in order to verify whether a reassessment is missing by an endpoint like described in section 6.2.4. There is only a table containing IP addresses of the endpoints and a timestamp of their last assessment required. That timestamp is determined by the last received recommendation from the PDP. The PEP checks regularly these timestamps and if the difference between current time and timestamp of last assessment is greater than a specified value the access will be revoked by the PEP.

[5] http://www.netfilter.org/

6.3.3 Policy Decision Point

The Policy Decision Point is responsible for processing incoming TNC messages from the Access Requestor and sending appropriate recommendations to the Policy Enforcement Point (more precisely to the PEP daemon on the PEP) after verifying the integrity of that endpoint. In detail, the PDP listens for incoming requests on the Network Access Layer (NAL) where either EAP or TLS can be used. The NAL is responsible for extracting the IF-TNCCS messages from the IF-T packets and passing them to the Integrity Evaluation Layer (IEL). The IEL is the actual TNC server and is responsible for verifying the integrity.

Unfortunately, there are some minor changes which must be made to the current version of the tNAC PDP in order to use it within this TNC-VPN integration concept. Section 7.4 explains the minor shortcomings and presents a redesign of the tNAC PDP which will solves the issues.

Determining the responsible PEP

When a VPN endpoint has successfully performed phase 1 of the TNC-VPN assessment (see 6.2.1.1), the endpoint can directly communicate with the Policy Decision Point using IP. However, the PDP must detect which Policy Enforcement Point is responsible for that endpoint in order to know where to send the recommendation.

In IEEE 802.1X environments this case is handled implicitly by IEEE 802.1X because the Policy Enforcement Point is the opposite communication partner since it encapsulates and forwards the EAP messages to the PDP[6]. There are several solutions available of determining the responsible Policy Enforcement Point for a certain endpoint as described below:

Enhancing IF-T message format: Extending the IF-T messages has the grave disadvantage that the implementation is no longer specification compliant and other TNC compliant clients will be out of scope.

Enhancing IF-TNCCS message format: Extending the TNCCS messages is quite similar as extending IF-T messages and has therefore the same disadvantages.

IP Subnet Masks: The PDP uses a configuration setting which determines whether the sender of an IP packet is an IEEE 802.1X PEP (e.g. a switch

[6]The communication between Access Requestor and Policy Enforcement Point is directly based on Ethernet (EAPoL). The PEP (e.g. a switch) encapsulates EAP packets in IP and sends them to the Policy Decision Point. More information regarding IEEE 802.1X is available at [23].

or access point) or a VPN client. There is no need to adapt the PDP configuration when adding new Road Warriors or remote VPN gateways. Local endpoints connected via 802.1X are not affected and there is no need to add any special configuration to the environment as long as only EAP is considered for local endpoints.

Considering IP subnet masks is quite easy to implement and requires no changes to a specification compliant TNC client. For that reason, the concept will follow the approach of using subnet masks as additional configuration parameter. Chapter 7.4 of the implementation covers the redesign of the PDP in detail.

6.4 TNC-VPN Protocols

The last sections have covered the phases and entities of the TNC-VPN architecture. However, there is one part of the TNC architecture left - the message transport protocols.

6.4.1 TNCCS Message Transport (IF-T)

As suggested by the requirements analysis at section 4.2.1, the new TLS based transport protocol offers many advantages when the Internet Protocol can be used by the endpoint. For that reason, the TLS binding will be considered by this concept in order to provide a fast, reliable and secure connection between Access Requestor and Policy Decision Point. Since there are no free and open implementations available yet, a prototype of this specification will be developed in section 7.5.1 of the implementation chapter.

6.4.2 PEP Message Transport (IF-PEP)

Currently, the TNC Work Group has only specified the IF-PEP binding to RADIUS as protocol between PDP and PEP. Unfortunately, RADIUS is a very powerful but also complex protocol and the development of a RADIUS enabled PEP daemon for exemplary purposes would be too complex. In order to keep the focus on the implementation of the TNC-VPN integration a strongly simplified version of the IF-PEP protocol will be developed in chapter 7.5.2. This protocol is considered for exemplary use and should be replaced by the official binding to RADIUS at a later point of time.

6.4.3 Switching the Transport Protocol

The specification of *IF-T binding to TLS* explicitly mentions the use case of using TLS for reassessment instead of EAP for local endpoints since it offers many opportunities as already explained. The reason why TLS should not be used for initial assessment lies on the fact that TLS requires TCP and thus IP must be available on the endpoint. However, if the client has already an IP address he can try to circumvent the TNC assessment and directly communicate with other network resources.

The fact that an endpoint can use EAP for the initial assessment and TLS for all further reassessments results in one problem when using server-side initiated reassessment as explained below:

server-side: The PDP initiates the reassessment. In order to start the assessment, the PDP requires the IP address of the endpoint otherwise the PDP will not be able to establish a TCP connection which is necessary when using TLS. Unfortunately, when EAP was used first, the PDP does not know the IP address of that endpoint.

client-side: The above described problem does not occur as long as the client knows the IP address of the Policy Decision Point.

However, this problem does not occur in a TNC-VPN architecture for two reasons. Using EAP first does not offer any additional security aspect since it is also transmitted by using IP through the VPN tunnel. For the same reason, the PDP knows the IP address of the sender.

6.5 Integration in existing IEEE 802.1X Environments

Another important requirement regarding this concept was the seamless integration in existing IEEE 802.1X environments. That means, access to the network is offered via VPN in case of external endpoints and via IEEE 802.1X in case of local endpoints. In both cases a TNC assessment should be accomplished.

Figure 6.5 shows how such an architecture can look like. There are two Policy Enforcement Points required, one PEP is required for handling endpoints connected by VPN and the second PEP handles local endpoints connected by a switch. The following sections introduce two important aspects regarding the integration between TNC-VPN and TNC-802.1X.

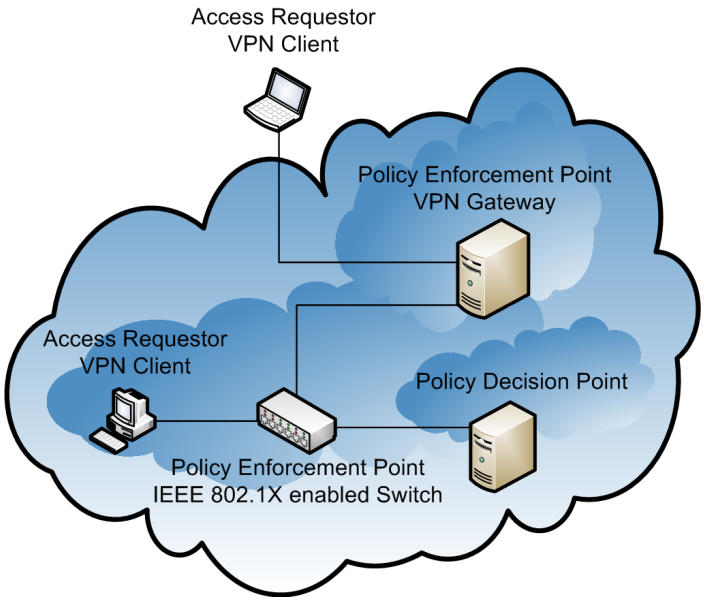

Figure 6.5: Real-world architecture considering TNC, VPN and IEEE 802.1X

6.5.1 Single Policy Decision Point

Although there are two Policy Enforcement Points required to handle local and remote Access Requestors, there is only one Policy Decision Point intended to be deployed in the environment. The reasons therefore are reducing complexity and minimizing administration effort. For that reason, the PDP must be capable of distinguishing between VPN and 802.1X connected endpoints and supporting multiple different Policy Enforcement Points each with different isolation strategies. Figure 6.5 illustrates the use of a single Policy Decision Point and two Policy Enforcement Points.

Fortunately, section 6.3.3 has already introduced a simple but powerful solution to find out the responsible Policy Enforcement Point for a certain endpoint. That approach must be extended by additional configuration parameters defining the available isolation strategies and supported IF-PEP protocols for that PEP.

6.5.2 Central Configuration Repository

A central configuration repository as mentioned by chapter 4.1.4 such as a LDAP service is very useful for larger environments with many endpoints or even several PEPs. Obtaining the data by a central repository is very useful and makes a lot of sense regardless of using TNC in conjunction with VPNs or not. For example, such

a central configuration repository can also be used to store the responsible Policy Enforcement Point for a certain subnet.

Using a LDAP server is recommended but does not belong directly to the topic of integrating TNC and VPN and is therefore not covered in more detail by this master thesis.

6.6 TNC in Site-to-Site VPNs

The simple Road Warrior use case as presented by the example scenario considers that the endpoint which demands access to the VPN is also the TNC endpoint. But in real-world VPN architectures there are not only single hosts connected to the VPN but whole networks (see chapter 2.2). Unfortunately, an integrity verification of all endpoints which are connected by a Site-to-Site VPN is difficult to achieve.

The problem lies at the desired transparency of the connected endpoints behind the VPN gateways. The endpoints should not get any notice of the VPN and as a result, as far as an endpoint gained access to the local network it also gained access to the VPN. Any additional authentication of the endpoints would contradict the meaning of a Site-to-Site VPN.

Actually, assuring that local endpoints within a Site-to-Site VPN do match a required system configuration is the job of local Policy Enforcement Points such as switches in the respective networks. The following enumeration introduces three approaches of how to deal with the endpoint integrity within Site-to-Site VPNs.

Contacting central PDP: The PEP checks for every IP packet whether the sender of an IP packet is already authenticated by contacting the PDP. The result of that request will be cached for gaining better performance. That approach is quite simple but has one disadvantage. It requires one central PDP for all connected network segment and as a result if that service is not available even local authentications via 802.1X are no longer possible.

PDP Synchronization Protocol: A special protocol is developed which is used for synchronizing the PDPs of the different network segments. After all PDPs have the same state regarding the endpoint information, the PEP can query the local PDP in order to check whether that endpoint was already verified.

Unfortunately, just synchronizing a list of already verified endpoints may not be sufficient since each PDP may have varying security policies.

Trust: Every endpoint which has access to the local network is also allowed to access resources of other network segments connected by a Site-to-Site VPN.

That approach prerequisites that each network segment does only grant local network access to verified endpoints.

When comparing these three approaches, the PDP Synchronization Protocol offers good security and does not risk the availability of all company units when the Policy Decision Point is unavailable. On the other hand, the Trust approach represents a good tradeoff between security, developing complexity and maintenance effort. However, since developing such a special synchronization protocol is a very complex task and needs further research on the topic, this concept will keep the focus on the TNC-VPN integration and therefore follows the Trust approach.

6.7 Security

The following sections cover the security requirements regarding the integration concept and introduce how they will be fulfilled.

6.7.1 Defeating IP Spoofing

The requirements analysis has introduced attacks based on IP spoofing such as in section 4.1.3.2 and 4.1.3.3. As already explained, the problem in TNC-VPN environments is that any access prevention mechanism is realized on layer three. Unfortunately, any access prevention solution relying on IP may be vulnerable to IP spoofing attacks. If an attacker is able to successfully spoof his IP address, he may usually bypass such restrictions.

However, many VPN solutions such as OpenVPN or strongswan offer solutions for preventing IP spoofing of the VPN endpoints. For example, OpenVPN prevents IP spoofing by comparing the source address of a packet with its internal association table for that client by default.

6.7.2 Defeating Rogue Access Requestors

The so called *Rogue Access Requestor* attack as described in section 4.1.3.4 does only work for IEEE 802.1X environments because the attacker just wants to open a switch open in order to gain access to the LAN. If the legitimate endpoint uses any encrypted EAP method (as required by the IF-T binding to EAP) the attacker is not able to read or manipulate the exchanged data but as far as the Policy Enforcement Point opens the port, the attacker can also access the network.

Fortunately, this kind of attack is not possible in a TNC-VPN environment since there is not just a port to be opened but an encrypted tunnel needs to be established

between Access Requestor and Policy Enforcement Point. If an attacker uses a hub or access point like described at 4.1.3.4 and just forwards VPN traffic between Access Requestor and PEP he will not gain any access until he can interfere in the encrypted communication.

In order to get this kind of attack working, the attacker must become a *real* Man in the Middle (MitM) so that he can influence the traffic and insert own packets. Without being a MitM, the attacker is just some kind of proxy forwarding encrypted traffic between Access Requestor and Policy Enforcement Point.

Man in the Middle attacks will be securely defeated by common VPN implementations like IPsec or OpenVPN since these attacks will not work when the client authenticates the VPN gateway before establishing the connection. Due to this fact, VPN authentication for both endpoints should be deployed if not activated by default.

6.7.3 Defeating bad VPN Gateways

A very similar attack to the Rogue Access Requestor is that an attacker uses a successfully verified endpoint in order to connect further malicious endpoints to the protected network as described in section 4.1.3.5. Other Network Access Control solutions are also vulnerable to this kind of attack like IEEE 802.1X.

Fortunately, this attack can be easily defeated due to using the features offered by Trusted Network Connect. There is only a special IMC / IMV pair on each endpoint required which checks whether the endpoint is capable of being used as a VPN gateway or not. Such an IMC / IMV pair must only check if the endpoint is capable of forwarding network traffic. Without the ability of forwarding network traffic the endpoint can not be misused by an attacker for this purpose.

6.8 Evaluation of the Concept

This chapter has presented a concept which enables a secure integration between Trusted Network Connect and Virtual Private Networks. The benefit of this concept is that only minimal development effort is required and many VPN solutions are supported without any need of adapting the concept.

Finally, this concept fulfills almost all requirements as mentioned by the requirements analysis and even the requirements which are not fulfilled completely by now can be added in future without any problems. Actually, there are only the following three items left which need further research or development:

Support of IF-PEP binding to RADIUS: The concept considers using a much more simple protocol for the purpose of transmitting recommendation from the Policy Decision Point to the Policy Enforcement Point. When considering the official binding to RADIUS, the development complexity and effort for realizing a prototype of this concept would be too extensive. However, there is only additional development effort required in order to support the binding to RADIUS and therefore to be compatible with TNC.

LDAP as central configuration service: As already explained by the requirements analysis as well as by the concept, there are many use cases available which would benefit of using a central LDAP service. Actually, the advantages offered by a LDAP service are not limited to a TNC-VPN integration but to the overall TNC architecture.

Verifying endpoints behind a VPN gateway: The concept covers authentication and platform verification only of the VPN endpoints. However, in case of a Site-to-Site VPN, potentially dangerous endpoints may access the network of another unit without being verified. Section 6.6 has already introduced a solution which considers the development of a special synchronization protocol. However, the development of such a protocol will be really complex and requires further research on the topic.

Table 6.1 compares the developed concept with the catalogue of requirements presented by table 4.1.

Category	Requirement	Realized
General Requirements		
	Maintenance	Yes
	Stability	Yes
	Central Configuration Management	Considered
Security		
	Updating vulnerable Applications	Yes
	IP-Spoofing to bypass TNC Assessment	Yes
	IP-Spoofing to break out of VLAN Jail	Yes
	Rogue Access Requestor	Yes
	Bad VPN Gateway	Yes
TNC		
	IF-T binding to TLS	Yes
	IF-PEP binding to RADIUS	No
	Enforcement of Reassessments	Yes
	Binary-based Isolation	Yes
	VLAN- or filter-based Isolation	Yes
VPN		
	Secure Authentication	Yes
	Generic Integration Approach	Yes
	No Source Code Manipulation	Yes
	Supporting Road-Warrior VPNs	Yes
	Supporting Site-to-Site VPNs	Partial
Integration in existing Environments		
	TNC - IEEE 802.1X	Yes

Considered:	This requirement is considered by the concept but not covered in detail.
Partial:	This requirement is partially fulfilled by the concept, however further research is required.
Yes:	This requirement is fulfilled by the concept.
No:	This requirement is not fulfilled by the concept.

Table 6.1: Requirements summary

7 Implementation

Chapter 6 has presented a concept of how Virtual Private Networks and Trusted Network Connect can be brought in line. This chapter covers the exemplary implementation of the concept on the basis of the sales-man example as introduced by section 6.1. The result of this chapter will be a prototype of the TNC-VPN integration as well as the establishment of a testing environment for future research.

The corresponding source code, configuration files and other resources concerning this implementation are available on the DVD attached to this master thesis.

7.1 General Requirements

There are many general requirements regarding the implementation of the example scenario. The resulting implementation will be integrated into the tNAC project and therefore, there are several requirements which must be fulfilled by the implementation.

Using C or C++ for development: Most components of the tNAC project are developed using C++. In order to ensure an easy integration with existing components, the necessary components will be developed by using C++.

Preferring boost libraries: *Boost*[1] offers many high quality C++ libraries which offers stability, very good performance and platform independence. For that reason, boost libraries will be preferred instead of using self-developed solutions.

Platform independence: The resulting implementations should be run on all operating systems supported by tNAC.

Specification Compliant: All parts which will be developed should be compliant to official specification unless a simplified alternative is required in order to reduce the development complexity.

[1] Boost provides many free high quality C++ libraries, see http://www.boost.org for more information.

VPN Technology: As mentioned by the concept, OpenVPN as well as IPsec can be used to implement these concept. This thesis covers the implementation on a basis of OpenVPN in version 2.0.6 since it is less complex to configure when compared to common IPsec implementations such as strongswan.

The example scenario was already illustrated by figure 6.1 in the concept chapter. However, figure 7.1 is very similar but extends the example scenario by the internal IP addresses of the endpoints which are considered in this chapter.

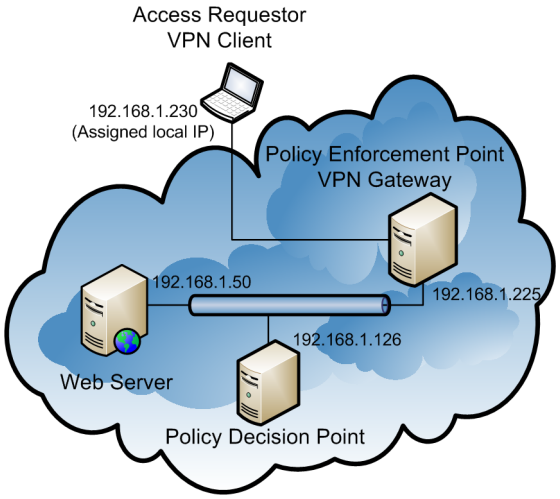

Figure 7.1: Example scenario with local IP addresses

7.2 Access Requestor

The Access Requestor consists of a TNC client as well as a VPN client. Each of them is responsible for performing one assessment phase (see assessment phases at section 6.2.1). The following sections covers the configuration of the OpenVPN client and the development of the TNC client application.

7.2.1 OpenVPN Client

The installation and configuration of OpenVPN is quite simple. After the installation has finished there is a only small configuration file and a signed certificate required. The certificate consists of a public and private key and is used by the VPN gateway to authenticate the client (see section 2.3.2). In practice, the certificates will be usually provided by the responsible network administrator. Listing 7.1 shows the content of the required configuration file.

Listing 7.1: OpenVPN client configuration file

```
1 client
2 dev tun
3 # 10.10.0.150 corresponds to the public IP of the VPN Gateway
4 remote 10.10.0.150
5
6 ca keys/ca.crt
7 cert keys/vpn-client.crt
8 key keys/vpn-client.key
```

The configuration of the VPN client is finished by providing this configuration file on the Access Requestor. Section 7.3.1 covers the installation of the OpenVPN gateway and shows that a VPN connection between Access Requestor and Policy Enforcement Point can be established.

7.2.2 TNC Client

Any TNC client which supports the IF-T binding to TLS can be used in order to handle phase 2 of the assessment. Unfortunately, there is neither an open and free client available which supports the TLS binding nor is there a general TNC client within the tNAC project available at the moment.

Since developing a full featured TNC client is out of scope for this master thesis there will be a prototype developed which is capable of serving the requirements concerning the integration concept. That implementation is based on the exemplary version of the IF-T TLS version developed in section 7.5.1. However, in order to demonstrate that the overall TNC-VPN architecture works as expected, the client must be capable of sending real measurements to the Policy Decision Point. For that reason, prior captured TNCCS message exchanges, where the client gained access to the network, will be replayed[2].

In detail, the client establishes a TLS tunnel to the Policy Decision Point and uses the IF-T binding to TLS implementation for the message exchange. The prior captured TNCCS messages contain valid measurements of the *IMCHostScanner*[3] and are stored in XML files. The content of these files will be loaded by the client application, encapsulated in appropriate IF-T messages and send to the PDP. In doing so, the client will gain access to the network as illustrated later in section 7.6 of this chapter.

[2] That proceeding is very similar to a typical replay attack whereas an attacker just replays prior captured messages.

[3] The IMCHostScanner is available at http://tnc.inform.fh-hannover.de/tnc/wiki/index.php/Download:_IMCHostScanner

The prototype of the TNC client is able to establish and communicate with the PDP by using the IF-T binding to TLS and only the IEL and IML layers are not implemented yet. The current version may be extended in the future to become a full featured TNC client.

7.2.3 Building Client Wrapper Script

At the moment there are two independent programs required which must be executed one after the other in order to gain access to the protected network. However, these process can be simplified by offering a simple shell script for that purpose as presented by listing 7.2.

Listing 7.2: Access Requestor startup script

```
1 #!/bin/sh
2 openvpn /etc/openvpn/client.conf
3
4 if [ $? -ne 0 ] ; then
5   echo "Error while starting OpenVPN";
6   return $?;
7 fi
8
9 access-requestor 192.168.1.126 12345
```

7.3 Policy Enforcement Point

The Policy Enforcement Point as described by the concept is divided into three parts each with its own focus. This section covers the configuration of OpenVPN and iptables as well as the development of the PEP daemon.

7.3.1 OpenVPN Gateway

This section describes the necessary steps for setting up a VPN between the network on one side and a Road Warrior (corresponds to the sales-man mentioned by the example scenario) on the other side.

In order to set up a secure OpenVPN environment, a *Public Key Infrastructure (PKI)* must be established first. For that purpose, OpenVPN offers a tool called *easy-rsa* which simplifies the overall procedure a lot. After OpenVPN is installed on the Policy Enforcement Point and the PKI was established there is only the configuration of the OpenVPN server left. The necessary server configuration file

is illustrated by listing 7.3. A detailed introduction regarding the configuration of OpenVPN and the establishment of a PKI by using *easy-rsa* is available at [24].

Listing 7.3: OpenVPN server configuration file

```
1 dev tun
2 mode server
3 tls-server
4
5 ifconfig 192.168.1.225 192.168.1.226
6 ifconfig-pool 192.168.1.230 192.168.1.251
7
8 route 192.168.1.224 255.255.255.224
9 push "route 192.168.1.0 255.255.255.0"
10
11 dh easy-rsa/keys/dh1024.pem
12 ca easy-rsa/keys/ca.crt
13 cert easy-rsa/keys/vpn-server.crt
14 key easy-rsa/keys/vpn-server.key
15
16 verb 4
```

Testing the Installation

The OpenVPN client should now be able to access the web server with the IP address *192.168.1.50* within the Virtual Private Network. Instead of using a browser, the procedure will be illustrated by sending *pings* destined to the web server, first without and later with established VPN tunnel.

Figure 7.2 illustrates that the sales-man is not able access the web server within the internal company network until OpenVPN will be started. In this case, OpenVPN was started due to using the corresponding runlevel script. Figure 7.2 also presents the manipulated routing table after OpenVPN was started on the client.

As a result, phase 1 of the TNC-VPN assessment is almost realized. However, the concept demands that an endpoint can only access the Policy Decision Point within the VPN until phase 2 has also finished successful.

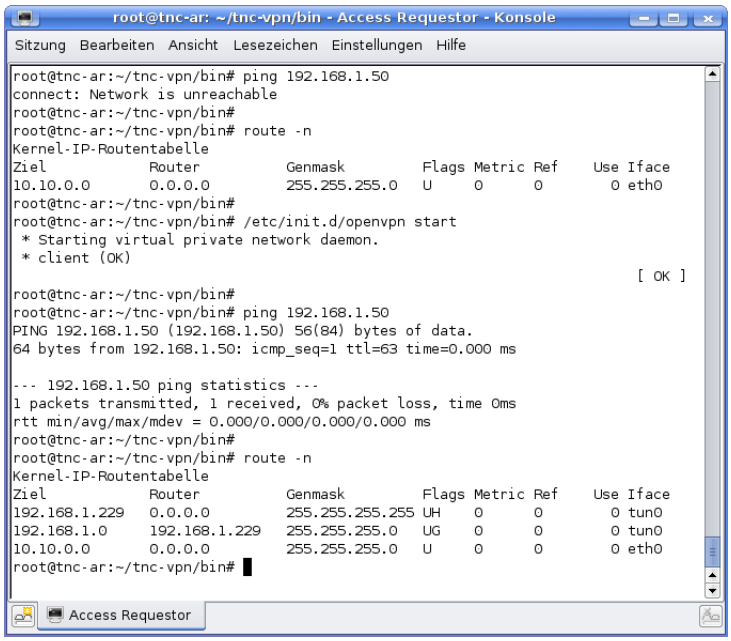

Figure 7.2: Testing OpenVPN client installation

7.3.2 iptables

The concept considers using a packet filter on the Policy Enforcement Point to prevent arbitrary access to the internal network. Iptables is small command line application which is used by almost all modern Linux operating systems for configuring a Linux based packet filter firewall. More precisely, iptables is a user space program which modifies the internal Netfilter modules of the Linux kernel. More information regarding iptables and how to use it can be found at [25].

Deploying a *deny-all* firewall strategy is considered by the concept whereas only access to the Policy Decision Point shall be granted. Listing 7.4 presents a shell script containing the required iptables commands in order to implement exactly that desired behavior on the PEP.

Listing 7.4: deny-all.sh - enabling deny all firewall strategy

```
1 #!/bin/sh
2
3 # Set default policy to drop
4 /sbin/iptables --policy FORWARD DROP
5
6 # Only PDP traffic is allowed by default
7 /sbin/iptables -A FORWARD -d 192.168.1.126 -j ACCEPT
8
9 # Enable stateful firewalling
```

```
10 iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

The rule at line 4 sets the default policy of the *forward* chain to *drop*. That has the result, that every packet which shall be forwarded by the Policy Enforcement Point will be dropped unless a certain allow rule will follow. Line 7 ensures that traffic destined to the PDP with IP address *192.168.1.126* will be allowed. Line 10 enables stateful firewalling which allows traffic forwarded back to the originator without explicitly specifying appropriate rules.

However, applying these rules manually after each system start is not recommended and error-prone since it opens the door to the network for every valid VPN client until these commands will be executed by the administrator. In order to defeat that vulnerability, listing 7.4 is already realized as a shell script which can be executed on the PEP before the OpenVPN server will be started.

Testing the Rules

After applying the iptables commands the Access Requestor will no longer be able to access the web server but only the Policy Decision Point. Figure 7.3 shows that the Access Requestor can only access the PDP after applying the *deny-all.sh* script (packets to the web server will be dropped). As a result, phase 1 of the TNC-VPN assessment is therefore fully supported by this implementation and the next step will be developing the mentioned PEP daemon.

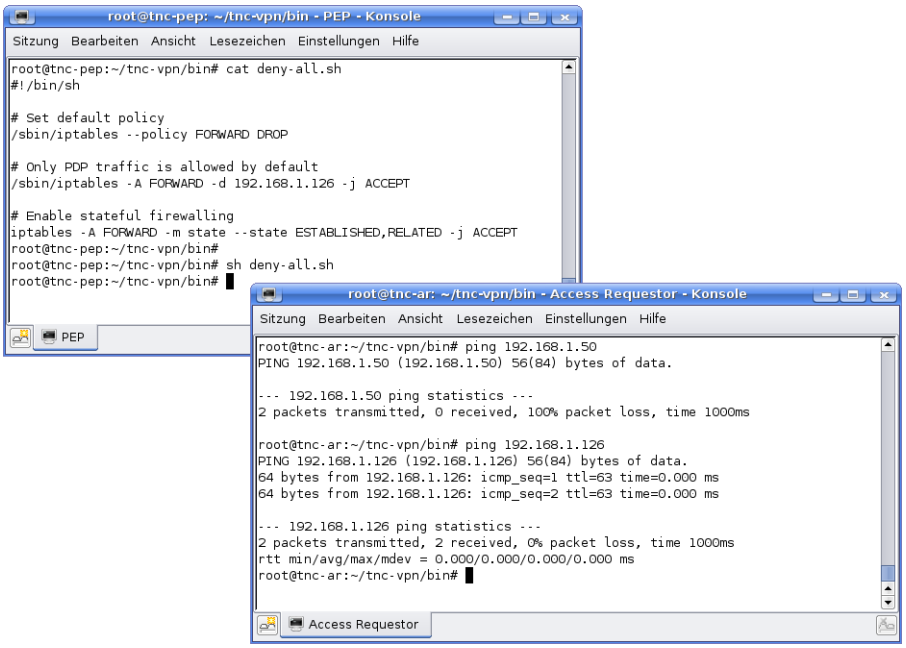

Figure 7.3: Only traffic destined to PDP is accepted

7.3.3 PEP daemon

The PEP daemon (PEPd) as introduced by the concept (see section 6.3.2.3) is responsible for processing IF-PEP messages sent by the PDP and manipulating the packet filter rules. The PEP daemon will be developed as an additional application to the tNAC project using C++ as programming language and the *boost* libraries for handling TLS based networking and threading. The PEPd consists of three parts each with its own focus. Each part will be explained apart in the following sections.

Networking

The job of the networking part is listening for incoming IF-PEP messages, extracting and parsing the payload and executing iptables commands according to the received recommendation. Actually, the PEP daemon executes shell script depending on the recommendation instead of directly invoking iptables. Using shell scripts for each use case such as granting or denying access offers the advantage, that system administrators can simply modify these iptables rules by using an editor. Furthermore, using scripts separates the PEP daemon from the used packet filter technology and decreases therefore the effort for porting the implementation to other operating systems. The corresponding shell scripts will be introduced in section 7.3.3.

At the moment the networking part is only capable of handling the simplified IF-PEP message format but is designed to get extended by full support of the official IF-PEP binding to RADIUS. The simplified IF-PEP message will be explained at 7.5.2.

Reassessment Scheduler

Reassessments are not actively performed by the PEP daemon in form of measuring the time of the last assessment and if necessary initiating a reassessment (server-side initiated). In doing so would require detailed logic regarding the endpoints and the TNC server and is therefore only implemented by the Policy Decision Point.

However, if only the PDP will be capable of revoking access to the network a security vulnerability may occur if the Policy Decision Point is unavailable. For that reason, the PEPd should also be capable of revoking access if no reassessment took place in a certain time interval. The application determines that by comparing the time of the last assessment with the current time. If the difference is greater than a specified value, the access to the network will be revoked. However, using this features requires the Policy Decision Point to send after each assessment a

recommendation message to the PEP also in cases where the endpoint isolation has not changed. The feature is implemented by using *boost threads*[4].

Shell Scripts

As already explained above, the PEP daemon invokes appropriate shell scripts instead of iptables directly. In doing so, an administrator can easily modify the set of commands which shall be executed depending on a certain recommendation. For example, the administrator can easily grant access to further resources within the VPN when the Access Requestor failed the TNC assessment.

This implementation of the concept only realizes binary-based isolation. For that reason, there are two shell scripts required. Listing 7.5 shows the *allow-client.sh* shell script which will be invoked to grant access to the network for an endpoint. That script expects the IP address of the endpoint as only parameter. However, implementing filter-based isolation as mentioned by the concept would just require minimal enhancements to the PEP daemon (includes extending the simplified IF-PEP protocol) and the development of additional shell scripts which cover the necessary iptables commands.

Listing 7.5: allow-client.sh - granting access to an endpoint

```
1 #!/bin/sh
2 /sbin/iptables -A FORWARD -s $1 -j ACCEPT
```

Listing 7.6 presents the *deny-client.sh* shell script which is the corresponding counterpart and will be executed in order to remove former granted access for an endpoint. Similar to *allow-client.sh*, this script expects also the IP address of the endpoint as only parameter.

Listing 7.6: deny-client.sh - revoking access to an endpoint

```
1 #!/bin/sh
2 /sbin/iptables -D FORWARD -s $1 -j ACCEPT
```

The presented iptables commands fulfill the requirements regarding binary-based isolation as demanded by the official TNC specification as well as by the integration concept. However, these commands can be extended by additional parameters in order to restrict further access. For example, the *allow-client.sh* can restrict the accept rule only to the VPN interface (e.g. tun0).

[4] http://www.boost.org/doc/libs/1_40_0/doc/html/thread.html

7.4 Policy Decision Point

This section covers the required changes and extensions to the tNAC Policy Decision Point. The PDP implementation as provided by the tNAC project represents a very good basis but there will be a redesign necessary in order to serve the requirements of the TNC-VPN architecture.

7.4.1 Redesigning the PDP

At time of this writing the tNAC Policy Decision Point is available in version 0.5.0. However, in that version the PDP is realized as an extension to the FreeRADIUS server. In detail, a plugin was developed which adds EAP-TNC capabilities to FreeRADIUS and builds therefore the bridge between FreeRADIUS and the TNC server. However, that design has the disadvantage from the TNC-VPN integration point of view that only the Extensible Authentication Protocol can be used for the TNCCS message exchange.

Unfortunately, there is a strong coupling between the FreeRADIUS specific plugin and the TNC server which makes it difficult to add a further NAL module to the PDP. However, such an additional NAL module will be required in order to add IF-T TLS capabilities to the TNC server. Redesigning the current version of the Policy Decision Point offers many advantages not only limited to the aspired TNC-VPN integration as shown below:

Loosing strong coupling to FreeRADIUS: Due to the strong coupling between TNC server and FreeRADIUS many scenarios are either not realizable or result in a high development effort.

Multiple IF-PEP bindings can be deployed: At the moment there is only the IF-PEP binding to RADIUS specified. However, additional IF-PEP bindings might be specified in the future. Adding custom versions like the simplified IF-PEP protocol can also be added easily. This might be interesting for further integration scenarios where RADIUS as protocol can not be used.

Multiple IF-T bindings can be deployed: Similar to the item directly above, the PDP should be capable of supporting all specified bindings for IF-T instead of only EAP.

Abstraction Layer

All the above mentioned recommendations can be achieved just be implementing an abstraction layer between the Network Access Layer and the Integrity Evaluation

Layer (see chapter 3.2 for more information regarding these layers). As a result of inserting an abstraction layer, multiple transport protocol bindings may be added without re-implementing the TNC server capabilities. Unfortunately, developing an universal approach which allows using a NAL and an IEL component of two different vendors is not possible because there is no interface specified between both layers and is therefore vendor dependent. Figure 7.4 illustrates the structure of the PDP with multiple NAL modules.

Redesigning the PDP will not just allow adding multiple NAL implementations but furthermore the deployment of a stand-alone version without the need of FreeRADIUS.

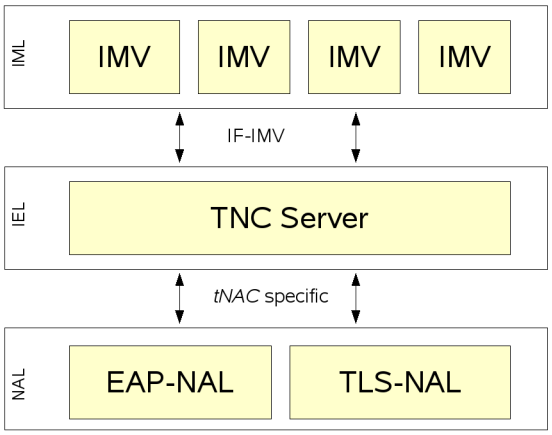

Figure 7.4: PDP after Redesign

The overall redesign of the Policy Decision Point was done in collaboration with the tNAC developers and is officially available since version 0.6.0. The following section describes the development of the new NAL component which offers support for the IF-T binding to TLS.

Developing TLS-NAL Component

The aim of the TLS-NAL component is offering support for IF-T binding to TLS, processing the corresponding packets and forwarding the TNCCS messages to the TNC server. Section 7.5.1 of this chapter introduces the details of the development of the IF-T binding to TLS.

Very similar to the PEP daemon, the NAL component is developed by using C++ as programming language and the boost libraries. Figure 7.5 illustrates the processing steps from receiving a new message to sending a appropriate recommendation to the PEP daemon. That diagram shows not each processing step but a simplified

version of the overall process. The first part shows the initialization of the TNC Server (*Coordinator* class) and the TLS server.

The second part begins with the *accept* call which indicates that a new client connection was established. After that a *message_handler* object for the *tls_session* will be constructed. The *message_handler* is responsible for handling any kind of IT-T message and depending on the message type forwards the TNCCS payload to the *Coordinator*. Before processing any message, a connection id must be generated by the TNC server. The connection id is used as an unique identifier for an assessment for one endpoint by the *Coordinator*.

The *do_operations* method of the *tls_session* checks whether there are new incoming messages or messages which should be send back to the Access Requestor. Within the *do_operations* method, the corresponding send or receive operations are invoked asynchronous in order to realize full duplex connectivity.

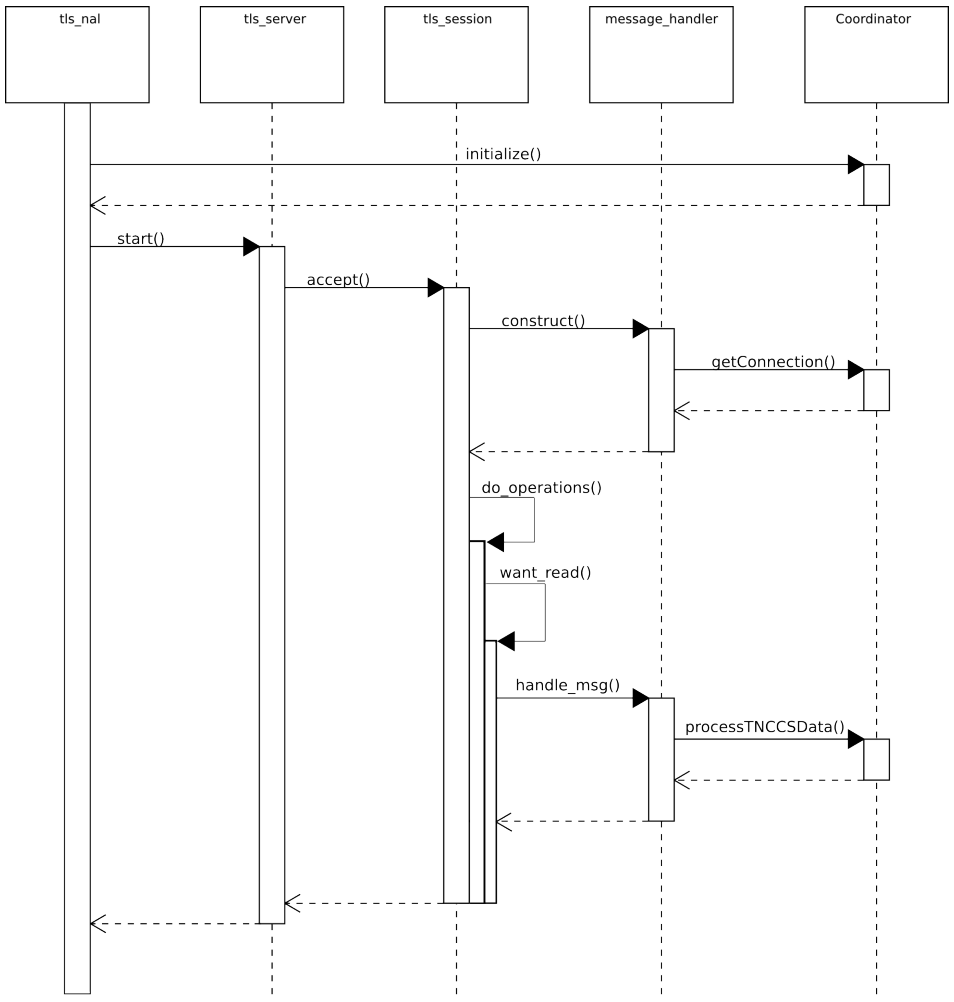

Figure 7.5: Processing steps of incoming message

The *message_handler* extracts the TNCCS message and forwards it to the *Coordinator* by using the *processTNCCSData* method. The *Coordinator* represents the only interface between this NAL module and the TNC server component.

Extending Data Structures

The Policy Decision Point data structures must be extended in order to know which PEP is responsible for a certain endpoint. The concept has introduced in section 6.3.3 several approaches and proposed a solution based on IP subnet masks. Listing 7.7 presents the corresponding part of the configuration file. There is one item for each Policy Enforcement Point required.

Listing 7.7: Subnet masks in the PDP configuration file

```
1 # PEP 192.168.1.225 is responsible for the network 192.168.1.224/27
2 192.168.1.225 = 192.168.1.224/27
```

As already considered by the requirements analysis as well as by the concept, such configuration settings can be stored in a central configuration facility such as a LDAP server. However, an integration between the tNAC PDP and a LDAP server is not covered by this implementation and may be realized in future work.

7.4.2 Reassessments

Reassessments are currently not supported by the tNAC Policy Decision Point. For that reason, reassessment must be initiated by the Access Requestor. The *Reassessment Scheduler* as described above in section 7.3.3 is not affected since that component works on the basis of received recommendations by the PDP.

7.5 Transport Protocols

The following sections describe the development of the IF-T binding to TLS as well as the simplified version of the IF-PEP protocol.

7.5.1 IF-T Binding to TLS

The IF-T binding to TLS transports TNCCS messages on top of a TCP connection in order to offer good quality of service properties. The message transport is protected by an encrypted TLS tunnel. The overall process of establishing the TLS connection, negotiating several properties and exchanging TNCCS messages is divided into three phases by the specification[17]:

TLS Setup: The task of the TLS Setup phase is establishing the TLS encrypted TCP connection between Access Requestor and Policy Decision Point.

IF-T Pre-Negotiation: The Pre-Negotiation phase is responsible for exchanging several capabilities of both endpoints such as the supported IF-T TLS version numbers. This phase is always performed before any TNCCS data will be transported.

IF-T Data Transport: In the Data Transport phase there will be primarily IF-TNCCS messages exchanged between Access Requestor and Policy Decision Point.

The implementation of the IF-T binding to TLS specification is only for exemplary purposes within this master thesis but the overall development is designed with future use and development in mind.

Developing the Prototype

Meeting the requirements of the TLS Setup phase is quite easy since only a TLS session between Access Requestor and Policy Decision Point must be established. Using *asio*[5] of the boost library for that purpose simplifies the development from loading password protected certificates over establishing a TLS encrypted TCP connection to handling the data exchange a lot.

The Pre-Negotiation phase is started directly after the TLS session was established. At the moment there is only the *version request* and *version response* implemented, but adding support for negotiating further parameters is considered by the implementation. After the Pre-Negotiation phase has finished the TLS connection can be used to exchange IF-T messages. As far as the TNC client and server have finished sending and receiving messages, the TLS session will be closed. The whole procedure of a IF-T TLS based communication session is shown at figure 7.6.

The prototype of this specification was developed as a shared object using C++ so that the same code base can be used on client and server side. Furthermore, the prototype is capable of performing all three phases as considered by the IF-T binding to TLS specification. However, that implementation is just for exemplary purposes and there are only few parts missing which are requested by the official specification such as the support of negotiating additional parameters.

[5] http://www.boost.org/doc/libs/1_39_0/doc/html/boost_asio.html

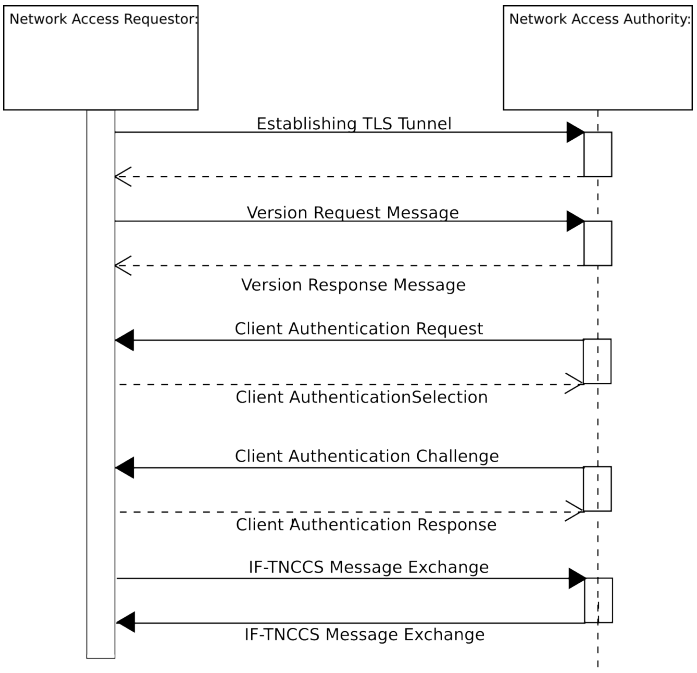

Figure 7.6: IF-T TLS message flow

7.5.2 IF-PEP

Implementing the IF-PEP binding to RADIUS for the communication between PDP and PEP daemon in this integration scenario would be too complex just for an exemplary use case. For that reason, a simplified version of the protocol was developed as already considered by the concept in section 6.4.2.

That basic protocol should only support those features which are required by the PEP daemon in order to realize binary-based isolation for a certain endpoint. As a result, the simplified IF-PEP protocol will only be capable of granting or denying (includes revoking) access for a certain endpoint to the network.

The structure of such packets is quite simple as illustrated by listing 7.7. The packet header starts with a three byte long fixed literal, followed by a one byte IP version identifier and a one byte recommendation flag. The version identifier indicates whether the last four bytes of such a packet contain an IPv4 address or the last twelve bytes contain an IPv6 address. The IF-PEP messages will be send through an encrypted TLS tunnel in order to secure the connection and to authenticate the parties.

This plain protocol is planned to get replaced by the offical IF-PEP binding to RADIUS in the future.

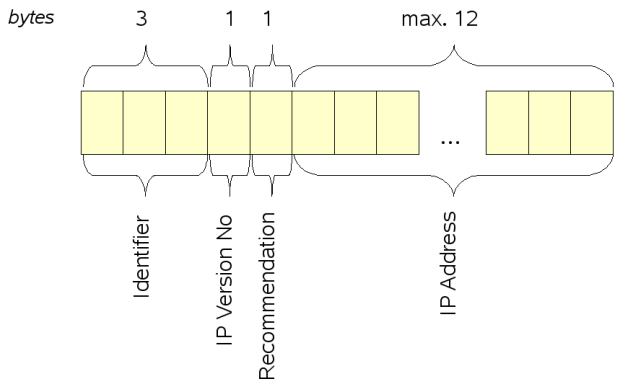

Figure 7.7: Simplified version of IF-PEP

7.6 Testing the Example Scenario

The development and configuration of all three entities including the required protocols is finished and the overall example scenario can be tested. Section 7.3.3 has already shown that the Policy Enforcement Point is capable of establishing VPN connections and preventing arbitrary access to the network unless the Policy Decision Point is the receiver of these packets.

However, the sales-man from the example scenario should now be capable of performing phase 2 of the TNC-VPN assessment due to starting the TNC client application after the VPN tunnel was created. The client will establish a TLS tunnel to the Policy Decision Point and will perform the TNC handshake. The Policy Decision Point verifies the measurements and sends a recommendation to the Policy Enforcement Point via the simplified version of IF-PEP.

The following three examples expect that phase 1 of the TNC-VPN assessment was already performed and therefore the sales-man has already a VPN connection to the company network as illustrated by figure 7.3.

7.6.1 Successful TNC Assessment

Figure 7.8 illustrates the overall interaction between Access Requestor, Policy Decision Point and Policy Enforcement Point. The Access Requestor has first tried to *ping* the web server with the IP address *192.168.1.50* but these packets were dropped by the PEP since the AR has not performed the TNC assessment yet.

For that reason, the sales-man initiates the TNC assessment by starting the *access-requestor* application. The PDP prints several log messages which indicates that the endpoint with the IP *192.168.1.230* (the sales-man's address) was successfully

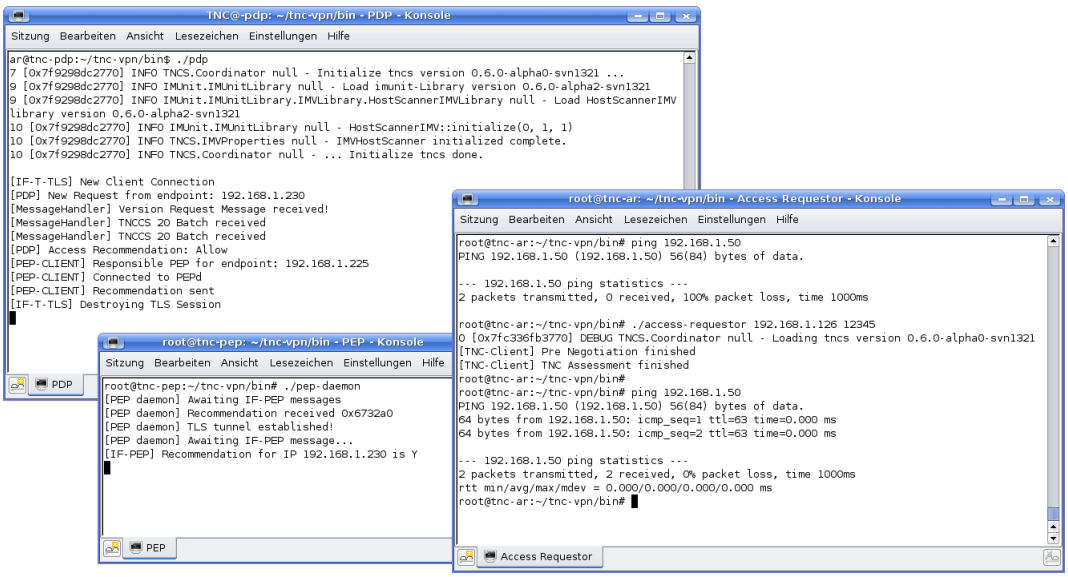

Figure 7.8: Successful TNC assessment

verified and that a corresponding IF-PEP message was sent to the responsible Policy Enforcement Point. The PEP daemon also prints a log message which indicate that the sales-man is now able to access the network.

That behavior is demonstrated due to executing the *ping* command again on the Access Requestor. The sales-man is now able to communicate with the web server.

7.6.2 Unsuccessful TNC Assessment

This section demonstrates that in case of a malicious endpoint the access will be denied. For that reason, the prior captured messages must be modified in such a way, that the Policy Decision Point (more precisely the IMVHostScanner) detects the endpoint as potentially dangerous. The replayed payload was modified so that port 22 is now in state open[6].

Figure 7.9 illustrates that the Policy Decision Point has securely detected that the endpoint may be dangerous according to the policy and has sent a corresponding access deny message to the PEP daemon. The endpoint is not able to *ping* the web server.

[6]The IMVHostScanner verifies the ports which are in state *open* on the Access Requestor.

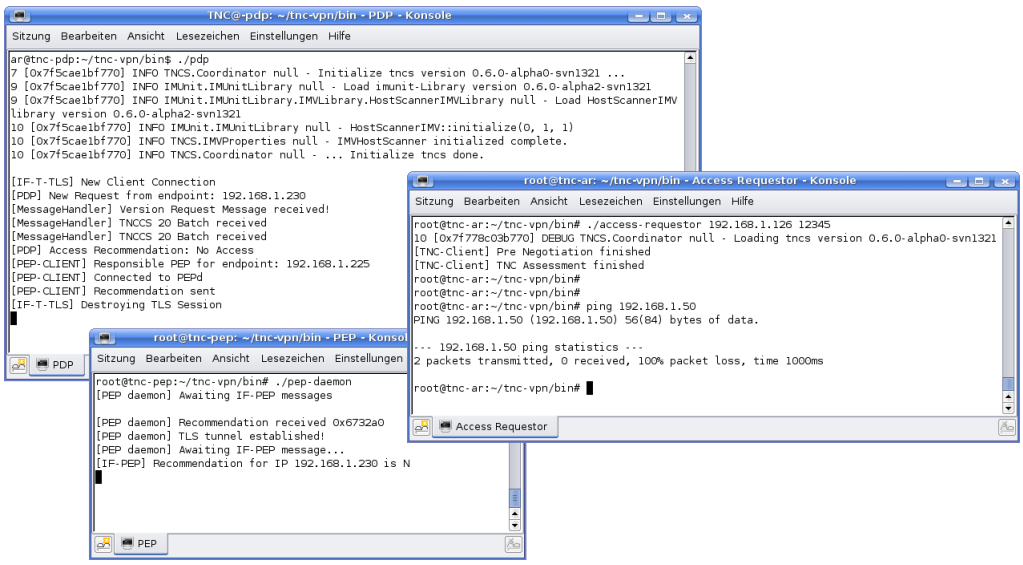

Figure 7.9: Unsuccessful TNC assessment

7.6.3 Defeating IP Spoofing

Section 2.3.2.2 has already explained that OpenVPN provides capabilities which can be used to prevent that an attacker changes his assigned IP address. The requirements analysis has introduced an attack where the attacker tries to circumvent the TNC assessment by changing his IP to the address of an already authenticated endpoint (see section 4.1.3.2). For example, an attacker gained access to the sales-man's notebook and installed a backdoor. Furthermore, the attacker knows that another endpoint with IP address *192.168.1.235* is already authenticated and corresponding access for that IP will be granted by the packet filter.

Figure 7.10 illustrates the attempt of an attacker by using IP spoofing to bypass the TNC assessment. First he tried to access the web-server directly by using the *ping* command. These access attempt was dropped by the packet filter since there is no corresponding accept rule for his IP address defined. In a second attempt, the attacker tries to access the web server by spoofing the source IP address of the packets to *192.168.1.235*. This was achieved by using the *hping3*[7] application. Changing the IP address for the local VPN device (e.g. tun0) would result in a shutdown of the OpenVPN connection.

Figure 7.10 shows that OpenVPN has securely detected that the attacker has changed his assigned IP address. These packets were dropped and a corresponding log message was written to syslog. As a result, the attacker is not able to circumvent the TNC assessment by spoofing his IP address.

[7] http://www.hping.org/

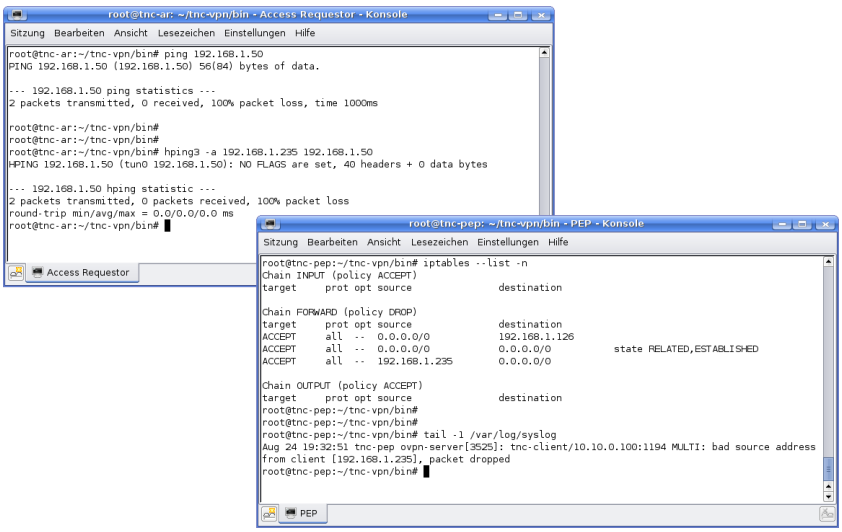

Figure 7.10: IP spoofing was prevented by OpenVPN

7.7 Conclusion

The introduced implementation offers a possibility to enforce TNC assessment for remote endpoints connected by a VPN client. The three examples have shown that the prototype of the integration concept works as expected. The sales-man has only gained access to the company network when he passed the VPN authentication as well as the TNC assessment successfully. Furthermore, bypassing the security restrictions by changing the IP to an address of an already authenticated endpoint was also prevented.

8 Conclusion and future Prospects

This master thesis has analyzed and developed an integration concept of the technologies Trusted Network Connect and Virtual Private Network. The aim of this thesis was the analyzes and the development of an integration concept, as well as the implementation of an exemplary prototype of the concept. A testing environment for future research on the topic was also established.

Before the development of the concept was started, a detailed requirements analysis was done in order to give an overview and to evaluate several requirements regarding their relevance. These analyzes reveals that an integration concept should do without manipulating the VPN software because otherwise there may occur problems in regard to security and stability. In addition, when adapting a solution either due to modifying the source code directly or by using a plugin facility limits the concept only to certain VPN implementations.

As a result of the requirements analysis, the concept has selected an integration method whereas the VPN and TNC technologies may remain untouched. In that proceeding, there are almost all VPN implementations supported without the need for any special configuration regarding Trusted Network Connect or even any adaptions to the VPN software itself. Furthermore, the concept has shown that a secure integration between two technologies will not always result in a high complexity and a large development effort. In this case, a secure integration was achieved due to the combination of both technologies and a deployment of a packet filter firewall.

A prototype of the concept was developed as part of this master thesis in order to illustrate that the integration works as desired. The prototype of the integration was implemented by considering OpenVPN as VPN technology and the tNAC TNC solution. The exemplary implementation has shown that the sales-man was only able to access the web server within the VPN after a successful verification of his system configuration was performed. The example has presented that the prototype is not vulnerable to attacks based on IP spoofing and therefore an attacker will not be able to bypass the TNC assessment. Changing the IP address to an already authenticated endpoint in order to gain access to the network without performing a TNC assessment was reliably defeated.

Moreover, there was a prototype of the new IF-T binding to TLS specification developed and successfully deployed for the message exchange between Access Requestor and Policy Decision Point.

The master thesis has also discovered topics for further research in the fields of how to connect two or more TNC protected networks via a Site-to-Site VPN or how to integrate a central configuration facility in an entire TNC architecture. In order to connect several networks securely, there is some kind of synchronization protocol required. Such a protocol should not only synchronize the verified endpoints of each segment but furthermore consider that each network segment may have a varying security policy. Another topic for future research may be the integration of a central configuration facility into a TNC architecture. Indeed, such an integration should not only consider certain parts such as the configuration of the Policy Decision Point, but the entire TNC architecture including TNC, VPN and RADIUS.

Bibliography

[1] Trusted Computing Group. *TNC Architecture for Interoperability*. 2009. https://www.trustedcomputinggroup.org/; visited on August 28th 2009.

[2] SANS Institute. *SSL Man-in-the-Middle Attacks*. 2002. http://www.sans.org/reading_room/whitepapers/threats/ssl_maninthemiddle_attacks_480; visited on August 28th 2009.

[3] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall International, 2002.

[4] Markus Feilner. *OpenVPN - Building and Integrating Virtual Private Networks*. 2006.

[5] P. Wouters and K. Bantoft. *Building and Integrating Virtual Private Networks with Openswan*. Packt Publishing, 2006.

[6] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409 (Proposed Standard), November 1998. Obsoleted by RFC 4306, updated by RFC 4109.

[7] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306 (Proposed Standard), December 2005. Updated by RFC 5282.

[8] N. Ferguson and B. Schneier. *Cryptographic Evaluation of IPsec*. 2003. http://www.schneier.com/paper-ipsec.pdf; visited on August 28th 2009.

[9] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), April 2006. Obsoleted by RFC 5246, updated by RFCs 4366, 4680, 4681.

[10] Ingo Bente. *Trusted Computing and Trusted Network Connect in a Nutshell*. 2008. http://trust.inform.fh-hannover.de/joomla/index.php/projects/67; visited on August 28th 2009.

[11] Trusted Computing Group. *TNC IF-IMC*. 2006. https://www.trustedcomputinggroup.org/; visited on August 28th 2009.

[12] Trusted Computing Group. *TNC IF-TNCCS*. 2006. https://www.trustedcomputinggroup.org/; visited on August 28th 2009.

[13] Trusted Computing Group. *TNC IF-IMV*. 2006. https://www.trustedcomputinggroup.org/; visited on August 28th 2009.

[14] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). RFC 3748 (Proposed Standard), June 2004. Updated by RFC 5247.

[15] Trusted Computing Group. *TCG Trusted Network Connect TNC IF-PEP: Protocol Bindings for RADIUS*. February 2007. https://www.trustedcomputinggroup.org/; visited on August 28th 2009.

[16] Trusted Computing Group. *TNC IF-T: Protocol bindings for Tunneled EAP Methods*. May 2007. https://www.trustedcomputinggroup.org/; visited on August 28th 2009.

[17] Trusted Computing Group. *TCG Trusted Network Connect TNC IF-T: Binding to TLS*. May 2009. https://www.trustedcomputinggroup.org/; visited on August 28th 2009.

[18] IEEE Computer Society. *IEEE 802.1Q - Virtual Bridged Local Area Networks*. 2003. Available online at http://standards.ieee.org/getieee802/download/802.1Q-2003.pdf; visited on August 28th 2009.

[19] J. Sermersheim. Lightweight Directory Access Protocol (LDAP): The Protocol. RFC 4511 (Proposed Standard), June 2006.

[20] P. Eronen and J. Korhonen. Multiple Authentication Exchanges in the Internet Key Exchange (IKEv2) Protocol. RFC 4739 (Experimental), November 2006.

[21] Gérard Pelka. *Entwicklung eines IPsec Security Gateways mit Trusted Network Connect Unterstützung*. German Bachelor Thesis; only available within the scope of the tNAC project.

[22] Paul Frieden. *VLANs on Linux*. 2004. http://www.linuxjournal.com/article/7268; visited on August 28th 2009.

[23] IEEE Computer Society. *IEEE 802.1X - Port-Based Network Access Control*. 2004. http://standards.ieee.org/getieee802/download/802.1X-2004.pdf; visited on August 28th 2009.

[24] OpenVPN. *OpenVPN Howto*. http://openvpn.net/index.php/open-source/documentation/howto.html; visited on August 28th 2009.

[25] Gregor N. Purdy. *Linux Iptables Pocket Reference*. O'Reilly Media, 2004.