Master Thesis
Electrical Engineering
Thesis no: MEE 10:93
November 2010

Evaluation of EAP-methods
Performance testing on IEEE 802.1x

(Peter Lorentzen, Johan Lindh)

School of Computing
Blekinge Institute of Technology
37179 Karlskrona
Sweden
This thesis is submitted to the School of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information

Authors:

Peter Lorentzen
E-mail: pelf005@gmail.com

Johan Lindh
E-mail: lindh.johan@gmail.com

External Advisor:

Björn Mattsson
Blekinge Tekniska Högskola

University advisor:

Mr Karel de Vogeleer, Ph.D. Student
COM/BTH

School of Computing
Blekinge Institute of Technology
371 79 KARLSKRONA SWEDEN
Internet: www.bth.se/com
Phone: +46 455 385000
SWEDEN
Abstract

Network administrators typically employ different methods for authenticating and authorizing the access to their networks. A flexible and scalable network access method is needed to combat the ever increasing network ubiquity brought on by technological advancements. The IEEE 802.1x Port-Based Network Access is a technology that allows transparent authentication to a network. It uses EAP-methods in order to authenticate against a server. There are a lot of different EAP-methods to choose from, and they vary in complexity and security.

This report will bring up the differences between the most commonly used authentication methods regarding the authentication time depending on different delay and network load.

Results showed that EAP-methods that are less complex take less time to perform authentication than their counterparts. When there is no delay, or a very small delay, this might not matter, but when the delay is higher complex EAP-methods take significantly longer time to perform the authentication process. This is very negative considering the nature of transparent authentication, and could lead to users becoming annoyed. A general formula for determining how long time an EAP-authentication process will take is presented.

**Keywords:** 802.1x, authentication, EAP-methods, RADIUS
Acknowledgements

We would like to thank our supervisor Karel de Vogeleer for his support and guidance. We also want to extend thanks to Björn Mattsson, for inspiring us to do this thesis, Fredrik Erlandsson for providing equipment and Charlott Lorentzen for her support and advice.
Contents

Abstract ii

Contents iv

List of Figures vii

List of Tables viii

Introduction 1

1 Introduction 2
   1.1 Scope .................................................. 3
   1.2 Research Questions .................................... 3
   1.3 Thesis Outline ......................................... 4

Technical background 4

2 Technical introduction 6
   2.1 IEEE 802.1x Port-Based Network Access .................. 6
       2.1.1 EAP and EAPOL ..................................... 8
       2.1.2 RADIUS protocol .................................. 9
       2.1.3 EAP Authentication Process ....................... 11
       2.1.4 EAP-Methods ..................................... 13

Methodology 16

3 Methodology 18
   3.1 Methodology ........................................... 18
# List of Figures

2.1 RADIUS packet Structure ........................................ 10  
2.2 Authentication Procedure ........................................ 12  

5.1 MD5 authentication time ........................................... 31  
5.2 TTLS-PAP authentication time ...................................... 32  
5.3 TTLS-CHAP authentication time .................................... 32  
5.4 TTLS-MD5 authentication time ...................................... 33  
5.5 TTLS-MSCHAPv2 authentication time ............................... 34  
5.6 PEAP-MSCHAPv2 authentication time ............................... 34  
5.7 Supplicant Processing Time ........................................ 35  
5.8 Authenticator Processing Time ..................................... 36  
5.9 Server Processing Time ............................................ 37  
5.10 Processing Time .................................................. 38  

6.1 Comparison of approximated and actual authenticaton time . 43
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>number of transmissions between the authenticator and the authentication server</td>
<td>30</td>
</tr>
<tr>
<td>6.1</td>
<td>$m$ and $k$ for different methods</td>
<td>42</td>
</tr>
<tr>
<td>6.2</td>
<td>number of transmissions</td>
<td>42</td>
</tr>
</tbody>
</table>
Glossary

AAA - Authentication, Authorization and Accounting.
BTH - Blekinge Institute of Technology.
DSL - Digital Subscriber Line.
EAP - Extensible Authentication Protocol.
EAPOL - Extensible Authentication Protocol over LAN.
IEEE - The Institute of Electrical and Electronics Engineers.
IETF - Internet Engineering Task Force.
LAN - Local Area Network.
MSCHAP - Microsoft Challenge-handshake Authentication Protocol.
NAC - Network Access Control.
OSI - Open Systems Interconnection.
PAP - Password Authentication Protocol.
PBNAC - Port-Based Network Access Control.
PEAP - Protected Extensible Authentication Protocol.
PPP - Point-to-Point Protocol.
RADIUS - Remote Authentication Dial In User Service.
TLS - Transport Level Security.
TTLS - Tunneled Transport Level Security.
VLAN - Virtual Local Area Network.
Introduction
Chapter 1

Introduction

Campus network administrators typically employ different methods for authenticating and authorizing the access to their network. A flexible and scalable network access method is needed to combat the ever increasing network ubiquity brought on by technological advancements.

Traditionally, the use of a web based log-in mechanism, where users have to provide user-name and password, is used to determine a user’s identity and allow or deny network access. Typically, every time a device loses its connection to the network, the users have to re-authenticate and provide their identity in order to be allowed, or denied, access. Instead of using this method for authentication and authorization (AA), it would be more user friendly to use some kind of automated process that authenticates and assigns devices to the correct Virtual Local Area Network (VLAN), based on some credentials associated with individual devices.

According to [21] and [29] manageability, scalability and security are the main issues for common access control methods. The IEEE 802.1x Port-Based Network Access Control [15] is a technology that tries to address these problems “by providing a modular, scalable and centralized design for port-based network access control.” [13]. It enables devices to automatically authenticate themselves to the network without having the user to take any action, thus making the authentication process transparent. However, the authentication process takes time, depending on factors such as network load and the complexity of the chosen authentication method, therefore it might take a few seconds before the authentication is completed. There are several kinds of authentication methods available with different characteristics such as security level, ease of use, compatibility, etc. Björn Mattson, one of the network experts at BTH, asked us to investigate what different types of authentication methods were available in the IEEE 802.1x technology. More specifically, he wanted help evaluating which method, or methods, would be most suitable to use in a campus environment such as BTH.

Since the process of authenticating a device should be transparent, the
time needed to complete the authentication must not be too long, on the other hand, a fast authentication method with very low security is not a good solution. This report will present a comparison between several different authentication methods in terms of authentication time depending on network delay.

IEEE 802.1x provide many interesting functionalities that are sought after by many network administrators [13, 14, 31, 12], this is because they provide means of relatively easy granting access to people whom want to connect to the network. This, combined with the ease of administrating and configuration, makes it a very lucrative technology. There has been a lot written about the suitability and viability of different authentication and encryption methods used in 802.1x [29, 13, 14, 12, 19], however, there has not been a lot published about the performance of different methods. We felt that this is also an important factor to consider when deciding which method is more suitable to use and implement in a network. As such, this work focuses on the technical aspects, and more specifically, on the aspects regarding the performance of different EAP-methods used in 802.1x.

\section*{1.1 Scope}

This work will focus on evaluating the performance of different EAP-methods, e.g. how the different methods respond to delay and what the processing time is for the whole authentication process (including processing time for supplicant, authenticator and authentication server).

\section*{1.2 Research Questions}

These are the research questions (RQs) that will be answered in our study:

- **RQ1:** Which metrics are important to consider in order to determine which credential is most suitable?
  - **SQ1.1:** Are there any available metrics that can be used to determine suitability or do new metrics have to be created?
  - **SQ1.2:** What metrics can be used during these specific circumstances that are suitable for an evaluation framework?

- **RQ2:** Which parameters are important to consider when creating the framework used to evaluate network access classification?
  - **SQ2.1:** Is it possible to do classification based on the data transmitted during authentication?
• **RQ3:** How do we verify that a computer fulfills the security restrictions required by the different groups using the 802.1x framework?

• **RQ4:** What parameters are to be considered when deciding on which VLAN to assign to the authenticated computer?

• **RQ5:** How do we, using the aforementioned framework, evaluate which type of network access control (NAC) policies are best suited?

### 1.3 Thesis Outline

In chapter two, we give a general introduction to IEEE 802.1x and its features. It explains what port-based network access is and how it works, as well as giving an overview of some of the available authentication methods used in IEEE 802.1x. We follow this chapter with an explanation of the research methodology that was used in this study. We motivate why we thought the chosen methodology had been best suited for this particular study. Chapter four depicts the laboratory environment used to conduct the experiments as well as a description of the experiments and how they were carried out. Following this, we present the results that we gathered from the experiments. Chapters six and seven are comprised of discussion around the outcome of the experiments and show the authors’ own conclusions around the work and its results, aside from providing some thoughts around possible future work.
Technical background
Chapter 2

Technical introduction

This section will give a rundown of the different technologies we used in this study. It will explain what Port-Based Network Access (IEEE 802.1x) is and how it works. All entities that are used will be shown and their purposes will be explained. Illustrations will be included to more easily describe how the different parts are connected and how they inter-operate. Furthermore, the main protocols that are used will be explained to give an overview of how they work.

2.1 IEEE 802.1x Port-Based Network Access

The IEEE 802.1x is a standard for Port-Based Network Access Control (PBNAC), it provides authentication on layer 2 of the OSI-model. As such, it can be used in all IEEE 802 networks. Simply put, the port users connect to when they want to access the network is closed by default, or in an “unauthorized” state. This means that even though a link layer connection has been established between the user and the switch no traffic is allowed to go through to the network. To gain access to the network, users have to authenticate themselves: once a successful authentication has been achieved, the port changes to open, or an “authorized” state, and network traffic is allowed to pass through the port.

Client/Supplicant

The client, or the supplicant as it is referred to from here on in this document, is the device that has to authenticate itself in order to gain access to the network. It can be any kind of device that supports IEEE 802.1x and can make use of the network e.g. PCs, mobile phones or PDAs. The supplicant requests access and provides the authenticator with its credentials. This communication takes place using EAP over LAN, EAPOL, on layer 2. Since EAPOL operates on layer 2, no IP address is required to initiate the authentication process.
CHAPTER 2. TECHNICAL INTRODUCTION

Authenticator

The authenticator is the device that controls the access to the network. It sets the state, either open or closed, of the ports that devices are connected to, based on information received from the authentication server. The authenticator relays information between the supplicant and the authentication server. This data will only be relayed until the authentication process is finished; once the process is finished, the authenticator will be notified by the authentication server whether or not the authentication has been successful. Depending on the outcome of the authentication process, the authenticator will then set the port state to either “authorized” or “unauthorized”.

Authentication Server

The authentication server processes and validates the supplicant’s credentials and then notifies the authenticator whether or not the supplicant is authorized to use the network. It also specifies which authentication method that shall be used between the supplicant and the authentication server. Furthermore, it can send optional parameters to the authenticator such as which VLAN the Supplicant is supposed to belong to and other user policies [23].

To be able to communicate with each other, the devices use three different protocols. The credentials, which are used for authentication, are sent from the supplicant to the authentication server, via the authenticator, using one of several available EAP-Methods. This data is then encapsulated from the supplicant to the authenticator using EAP. The EAP-messages are further encapsulated between the supplicant and authenticator using EAPOL, and between the authenticator and authentication server using RADIUS.


2.1.1 EAP and EAPOL

The Extensible Authentication Protocol (EAP) is an extension to the Point-to-Point Protocol (PPP) [28]. It is an authentication framework defined in [3] providing for the transport and usage of keying material and parameters generated by EAP methods. It is important to point out that EAP is not a wire protocol; instead, it is only a framework defining message formats. Each protocol that uses EAP defines a way to encapsulate EAP messages within that protocol’s messages.

EAP was developed in response to an increasing demand for remote access user authentication that uses third-party security devices [13]. EAP provides a standard mechanism for support of additional authentication methods within PPP. By using EAP, support for a number of authentication schemes might be added.

Since EAP is part of PPP it is not suited to use on 802.3 networks. In response to this a new protocol was defined; EAP over LAN, or EAPOL, specified in [22]. EAPOL is a very simple protocol and is used to encapsulate EAP-messages in Ethernet frames, as such it’s used for transport over wired and wireless Ethernet LANs. EAPOL enables communication between the supplicant and the authenticator. Communication consists of two types of packages: the authenticator sends ’Request identity’ packages and the supplicant sends ’Response’ packages to the authenticator.
CHAPTER 2. TECHNICAL INTRODUCTION

2.1.2 RADIUS protocol

Remote Authentication Dial-In User Service (RADIUS), described in [26] and [24], is commonly used to provide centralized authentication, authorization, and accounting for dial-up networks, virtual private networks, wireless and wired networks. It was originally developed for dial-up remote servers but is now also supported by authenticating Ethernet switches, wireless access points, Digital Subscriber Line (DSL) access, Virtual Private Network (VPN) servers, and other network access types.

RADIUS is a client/server protocol that runs on the application layer of the OSI model. The RADIUS server has three main tasks.

1. authenticate users or devices before granting them access to a network
2. authorize users or devices for certain network services
3. account for usage of authorized services.

The RADIUS server receives a RADIUS Access Request message from the device, controlling the network access requesting authorization to grant access via the RADIUS protocol. The request includes access credentials, the form of these credentials is described later in this chapter. But typically there is a user-name and a password. Additionally, the request may contain other information which the access point knows about the user such as MAC-address or information about the user’s physical point of attachment.

The RADIUS server checks that the credentials are correct and verifies the user’s identification. RADIUS servers check the user’s information against a locally stored flat file database or against external sources - commonly SQL, Kerberos, LDAP, or Active Directory servers - to verify the user’s credentials.

RFCs 2865 and 2866 define the following RADIUS message types:

- **Access-Request.** Sent by a RADIUS client to request authentication and authorization for a network access connection attempt.

- **Access-Accept.** Sent by a RADIUS server in response to an Access-Request message. This message informs the RADIUS client that the connection attempt is authenticated and authorized.

- **Access-Reject.** Sent by a RADIUS server in response to an Access-Request message. This message informs the RADIUS client that the connection attempt is rejected. A RADIUS server sends this message if either the credentials are not authentic or the connection attempt is not authorized.

- **Access-Challenge.** Sent by a RADIUS server in response to an Access-Request message. This message is a challenge to the RADIUS client that requires a response.
- **Accounting-Request.** Sent by a RADIUS client to specify accounting information for a connection that is accepted.

- **Accounting-Response.** Sent by the RADIUS server in response to the Accounting-Request message. This message acknowledges the successful receipt and processing of the Accounting-Request message.

Each RADIUS message consists of a RADIUS header and RADIUS attributes. A RADIUS attribute specifies a piece of information about the connection attempt, e.g., there are RADIUS attributes for the type of service requested by the user, the user ID, the user password and the IP address of the access server. RADIUS attributes are used to convey information between RADIUS clients, RADIUS proxies, and RADIUS servers. For example, the list of attributes in the Access-Request message includes information about the user credentials and the parameters of the connection attempt. In contrast, the list of attributes in the Access-Accept message includes information about the type of connection that can be made, connection constraints, and any vendor-specific attributes (VSAs).

the RADIUS attributes are described in [26, 24, 34, 36, 25, 2].

Figure 2.1: RADIUS packet Structure

<table>
<thead>
<tr>
<th>Code</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Access-Request</td>
</tr>
<tr>
<td>2</td>
<td>Access-Accept</td>
</tr>
<tr>
<td>3</td>
<td>Access-Reject</td>
</tr>
<tr>
<td>4</td>
<td>Accounting-Request</td>
</tr>
<tr>
<td>5</td>
<td>Accounting-Response</td>
</tr>
<tr>
<td>11</td>
<td>Access-Challenge</td>
</tr>
<tr>
<td>12</td>
<td>Status-Server (experimental)</td>
</tr>
<tr>
<td>13</td>
<td>Status-Client (experimental)</td>
</tr>
<tr>
<td>255</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Figure 2.1 shows the packet structure for RADIUS. The attribute value pair is where the additional information is sent along.

RADIUS messages are sent using User Datagram Protocol (UDP) messages, port 1812 is the standard port for RADIUS authentication messages, port 1813 for RADIUS accounting messages. To ensure security in the trans-
missions between the RADIUS client and the RADIUS server they are configured with a common shared secret.

2.1.3 EAP Authentication Process

In IEEE 802.1x enabled networks, a successful authentication is required for a port to change its state from “unauthorized” to “authorized” to let traffic pass through.

The authentication process can be started either by the supplicant or the authenticator. In the case when the supplicant initializes the authentication process, it starts by sending an EAPOL-Start message. The authenticator responds with an EAP-Request/Identity message back to the supplicant, the supplicant responds with its identity in the form of an EAP-Response/Identity. Once this packet reaches the authenticator, it has to be re-encapsulated into a format the authentication server can understand, e.g. RADIUS. If the authenticator initializes the authentication process, the first step is skipped.

The authentication server then processes this information in order to determine whether or not the supplicant is allowed access. If the identity of the supplicant is correct, an EAP-Success message is sent to the supplicant, relayed by the authenticator, and the port changes to “authorized” and traffic can pass through freely. If, however, the identity is incorrect, an EAP-Failure message is sent instead and the port remains in an “unauthorized” state and the supplicant is denied access and cannot send traffic through the port.

Depending on which EAP-Method is used, there might be additional transmissions, e.g. setting up secure tunnels, but this is the general flow of the authentication process. The important thing to notice is that the communication between the supplicant and the authenticator uses EAPOL whereas the authenticator and the authentication server typically communicates using RADIUS.

EAP message exchange

1. The supplicant connects to the network and tries to access information from the network.
2. The authenticator responds by requesting its identity (ID) using EAPOL.
3. The supplicant responds with its Identity (ID) using EAPOL.
4. The authenticator forwards the ID of the client to the server, using the protocol agreed on by the server and authenticator. In the EAP standard there is support for multiple back-end authentication servers.
5. The authenticator server responds back to the authenticator with a challenge. There are several EAP-methods available for this challenge which are described below.

6. The authenticator then relays the challenge to the supplicant. The EAP-Method is encapsulated using EAPOL.

7. The supplicant checks whether it has support for the requested EAP-method. If the supplicant does not support the requested EAP-Method, it will reply back to the server with a NAK, the server will then try to negotiate a different method. When a supported method is requested the Supplicant replies back to the server with its credential using the requested method encapsulated using EAPOL.

8. The authenticator receives the response from the supplicant and decapsulates it from the EAPOL and then re-encapsulates it using RADIUS and sends it on to the server, depending on what method used the validation of the supplicant can need multiple transmissions between the server and the supplicant.

9. If the supplicant’s credentials are valid, the server informs the authenticator that it should grant access to the supplicant. It can also send additional information regarding access restrictions, the supplicant might only be allowed to access some parts of the network. If the credentials are invalid or the supplicant and server could not negotiate which method to use, the server will send a fail to the authenticator and the supplicant will be blocked from the network.
CHAPTER 2. TECHNICAL INTRODUCTION

2.1.4 EAP-Methods

EAP by itself does not provide a secure authentication process. In order to protect the authentication messages, it is necessary to use an EAP-authentication protocol (EAP-method). Even though there are a large amount of methods available, only three, MD5-challenge, OTP and GTC, are specified in the original RFC [5]. There are a few proprietary methods available, but these methods may not be supported by all vendors.

MD5-Challenge

EAP-MD5 was standardized along with EAP in [5]. It uses ID and password as the authentication method; the authentication server stores a database of user-IDs and their corresponding passwords. It is a challenge-response handshake protocol, this means that the authentication server generates a random challenge string to the supplicant. To answer the challenge, the supplicant calculates an MD5-hash from the challenge and the user password. This hash is then sent back to the authentication server as a challenge response. Once the authentication server receives the challenge response it does the same process as the supplicant; it calculates the MD5-hash of the challenge and the password corresponding to the user, if the resulting hash matches the challenge response the authentication is successful, if it does not match the client is denied. It is important to note that the password is never sent between the two entities.

EAP-MD5 suffers from a few drawbacks, the main security risk is that all user passwords are stored in plain text on the authentication server [13, 19]. Man-in-the-middle attacks as well as session hijacking are security risks when using EAP-MD5 in wireless networks, this is because EAP-MD5 does not provide mutual authentication. EAP-MD5 will not be useful to network managers as a tunneled authentication method; support for MS-CHAP-v2 is more common. It is more likely that EAP-MD5 will be used in non-wireless environments outside of TTLS or PEAP.

TLS

EAP-Transport Level Security, EAP-TLS, was first defined in [1]. Unlike EAP-MD5, it is not a password-based authentication method, instead it uses certificates for the authentication method. It requires the use of a public key infrastructure, PKI, in order for mutual authentication between the authentication server and the supplicant to function. This is because EAP-TLS requires both server and client certificates. EAP-TLS provides strong security [13, 19], a TLS tunnel is used between the supplicant and the authentication server in order to protect the EAP-messages. Even though TLS is considered one of the most secure EAP-methods, it’s rarely used because certificate roll out and maintenance are so complex due to the requirement
of a functioning PKI infrastructure. Another thing to note is that even though a tunnel is used to protect EAP-messages, the user identity is still sent in clear text before the certificate exchange begins.

The EAP-TLS authentication process is described as follows:

1. The supplicant sends an EAP-Start message to the authenticator.
2. The authenticator replies with an EAP-Request/Identity message.
3. The supplicant sends its user-name to the authenticator in an EAP-Response message.
4. The authenticator forwards the identity to the authentication server encapsulated in a RADIUS Access Request message.
5. The authentication server responds to the supplicant with its digital certificate.
6. The supplicant validates the authentication server’s digital certificate.
7. The supplicant replies to the authentication server with its digital certificate.
8. The authentication server validates the supplicant’s credentials against the supplicant’s digital certificate.
9. The supplicant and authentication server derive encryption keys.
10. The authentication server sends the authenticator a RADIUS-ACCEPT message, including the supplicant’s WEP key, indicating successful authentication.
11. The authenticator sends the supplicant an EAP-Success message.
12. The authenticator sends the broadcast key and key length to the supplicant, encrypted with the supplicant’s WEP key.

TTLS

EAP-Tunneled TLS, EAP-TTLS[11], was developed in order to eliminate the need for a PKI infrastructure. It does however still retain support for mutual authentication, but unlike EAP-TLS, it only requires the authentication server to authenticate itself to the supplicant using a certificate. Once the secure tunnel between the authentication server and the supplicant is established, the process of authenticating the supplicant begins. EAP-TTLS supports legacy protocols such as PAP, CHAP, MS-CHAP and MSCHAPv2 for supplicant authentication. The authentication process, using one of the authentication methods, takes place inside the secure tunnel.
PEAP

Protected EAP Protocol (PEAP) [22] is very similar to EAP-TTLS, the key difference is that PEAP does not allow legacy protocols for authentication inside the secure tunnel. Instead, it only allows the EAP variants of the legacy protocols, such as EAP-MD5 and EAP-MSCHAPv2. The authenticator only acts as a pass-through device when a supplicant using PEAP wants to authenticate itself. This means that the authenticator does not need to understand the specific EAP protocol used for authentication.

PEAP addresses the weaknesses of EAP by:
- Protecting user credentials.
- Securing EAP negotiation.
- Standardizing key exchanges.
- Supporting fragmentation and re-assembly.
- Supporting fast reconnections.

PAP

The Password Authentication Protocol, PAP, [17] was originally specified for use with PPP. PAP transmits the user name and password across the network un-encrypted. Therefore, it is insecure to be used over an open network that does not provide privacy protection. The PAP authentication process is fairly simple:

1. After the Link Establishment phase is complete, an Id/Password pair is repeatedly sent by the peer to the authenticator until authentication is acknowledged or the connection is terminated.

2. The authenticator keeps a table of user-id/password and compare the received ID and password pair with the ones stored in its database.

3. If there is a match the authenticator sends a ACK back to the client. If there is no match the authenticator sends a NAK and terminates the connection.

PAP is not an EAP method itself, but can be used as authentication method using TTLS.

CHAP

Like PAP, Challenge Handshake Authentication Protocol, CHAP, [27], was designed for use with PPP. But, as the name suggests, instead of sending the password openly over the network the authentication server issues a challenge to the supplicant. The client proves its identity by successfully responding to the challenge issued by the authentication server. To respond
to the challenge the supplicant has to calculate a hash of the shared secret. This is one of the major drawbacks of CHAP; even though the password is never sent over the network, the shared secret is stored in plain text on both the supplicant and the authentication server.

The CHAP authentication process is as follows:

1. After the Link Establishment phase is complete, the authenticator sends a "challenge" message to the supplicant.

2. The supplicant responds with a value calculated using a "one-way hash" function.

3. The authenticator checks the response against its own calculation of the expected hash value.

4. If the values match, the authentication is acknowledged; otherwise the connection is terminated.

CHAP is not an EAP method itself, but can be used as authentication method using TTLS.

MS-CHAP

Microsoft CHAP, MS-CHAP, [35] was designed by Microsoft to offer enhanced functionality for Windows systems. The difference between MS-CHAP and CHAP is that the former does not require that the shared secret is stored in clear-text. Instead, MS-CHAP calls for a particular one-way cryptographic hash of the password to be used to store the password on the server. The client can reproduce this hash because of the fact that it knows which hash method is used by the server. The resulting hash is what is used in the challenge/response handshake authentication.

MS-CHAP is not an EAP method, and is only supported by TTLS.

MS-CHAP has several identified security vulnerabilities [19, 13] and was obsoleted by MS-CHAPv2, which was initially introduced with Windows 2000 and documented in [33]. MS-CHAPv2 introduced new functionality such as mutual authentication, improved keying and key generation and it eliminated the weak encoding of passwords for older clients. MS-CHAP-v2 is widely supported by Microsoft clients, and is commonly supported and used as an inner authentication method with PEAP. MS-CHAP-v2 is defined both as a PPP method, which means it can be used "as-is" within TTLS, and as an EAP method, which means it can be used as "EAP-MS-CHAP-v2" tunneled within both TTLS and PEAP.
Methodology
Chapter 3

Methodology

This chapter will explain the processes used throughout the work of this thesis; including motivations of why we chose to use experiments for generating results and how we analyzed the generated results.

3.1 Methodology

To be able to answer the research questions a mixed research approach was chosen. This means that both qualitative and quantitative [8] approaches were taken in order to successfully answer the research questions.

The work started with a literature study to gain knowledge and get a good grasp of relevant work, both theoretical and practical, that had been done on the subject. This study also helped us get a good understanding of the inner workings of IEEE 802.1x. The methods that were used when conducting the systematic literature review are based on the framework created by Kitchenham et al [4].

In order to find relevant articles a number of different steps were taken. To start with queries using broad search terms were done on databases such as IEEE Xplore, ACM, Science Direct and Google Scholar. To minimize the amount of hits, and get more relevant papers, the search keywords were refined and databases with low relevance to the topic were excluded. The papers were then rated based on how well their topics reflected the search words. This first iteration gave a good insight into which keywords would give relevant results. With this knowledge in mind the keywords were refined once more and inclusion and exclusion criteria were constructed to help with the screening process.

Once the refined keywords were satisfactory another iteration was started. During this iteration the abstracts were read, and the inclusion and exclusion criteria applied, in order to determine whether or not the paper was of any relevance to our study. When a number of papers had been selected based on their keywords and their abstracts a quality assessment of the se-
LECTED PAPERS WAS DONE. THIS QUALITY ASSESSMENT DIFFERED FROM THE INCLUSION AND EXCLUSION CRITERIA IN THE SENSE THAT IT WAS MORE THOROUGH.

IN ADDITION TO THE LITERATURE STUDY WE ALSO HAD MEETINGS WITH BJÖRN MATTSSON AND CONSULTANTS FROM ATEA WHERE WE DISCUSSED FEATURES OF THE PORT-BASED AUTHENTICATION MECHANISMS. THESE MEETINGS WERE OF GREAT VALUE TO US BECAUSE THEY GAVE US A GOOD INSIGHT INTO HOW BJÖRN MATTSSON ENVISIONED THE IMPLEMENTATION AND FINAL RESULT. BEING ABLE TO ATTEND MEETINGS WITH EXPERTS IN THE FIELD OF NETWORKING EXCHANGING IDEAS AND QUESTIONS WAS VERY BENEFICIAL AS WELL.

THE GOAL OF THIS WORK WAS TO EVALUATE THE PERFORMANCE OF DIFFERENT EAP-METHODS, I.E., HOW DIFFERENT METHODS RESPOND TO DELAY IN A NETWORK, AND WHETHER OR NOT THE INTRODUCTION OF DELAY AFFECTS THE PROCESSING TIME FOR THE WHOLE AUTHENTICATION PROCESS. THE LATTER INCLUDES PROCESSING TIME FOR THE SUPPLICANT, THE AUTHENTICATOR AND THE AUTHENTICATION SERVER. GIVEN THIS, CONDUCTING EXPERIMENTS IN A SETTING WHERE ALL PARAMETERS COULD BE CONTROLLED TO BETTER REFLECT THE DESIRED CONDITIONS WAS, ACCORDING TO THE AUTHORS, THE BEST APPROACH. ALL DATA THAT HAD TO BE COLLECTED IN ORDER TO ANSWER THE RESEARCH QUESTIONS WERE GATHERED THROUGH LABORATORY EXPERIMENTS. THE LABORATORY ENVIRONMENT AND ALL OF ITS COMPONENTS ARE EXPLAINED IN THE NEXT CHAPTER.

TO BE ABLE TO SEE IF DELAY HAD ANY AFFECT ON PROCESSING TIME IT WAS NECESSARY TO ESTABLISH A POINT OF REFERENCE; IN THIS CASE THE POINT OF REFERENCE WAS WHEN NO DELAY HAD BEEN INTRODUCED, I.E. THE NETWORK WAS “AS IS”. DATA OF THE PROCESSING TIME HAD TO BE COLLECTED TO GAIN KNOWLEDGE ABOUT HOW LONG THE TOTAL AUTHENTICATION PROCESS TOOK. IN ORDER TO ACQUIRE THIS INFORMATION DATA OF THE ENTIRE PROCESS HAD TO BE TAKEN, AND FROM THIS DATA EXTRACT THE INFORMATION. SINCE THE AUTHENTICATION PROCESS STARTS WHEN THE SUPPLICANT SENDS AN ‘EAPOL-START’ PACKET AND ENDS ONCE THE SUPPLICANT RECEIVES EITHER AN ‘EAPOL-SUCCESS’ OR AN ‘EAPOL-FAILURE’ PACKET IT WAS POSSIBLE TO MONITOR THE NETWORK FOR THESE TYPES OF PACKETS AND AS SUCH, A PICTURE OVER ONE AUTHENTICATION PROCESS CYCLE, I.E. FROM STARTING THE AUTHENTICATION PROCESS TO FINISHING IT. BY USING AN APPLICATION THAT CAN ANALYZE PACKETS BEING SENT OVER THE NETWORK, IN THIS CASE WIRESHARK, IT IS POSSIBLE TO EXTRACT TIME-STAMPS FROM ALL PACKETS.

IN WIRESHARK IT IS POSSIBLE TO APPLY FILTERS TO AFFECT, OR CONTROL, HOW THE OUTPUT IS SHOWN. BY APPLYING A FILTER SO THAT WIRESHARK ONLY OUTPUTS ‘EAPOL’ MESSAGES IT IS EASY TO MONITOR FOR BOTH THE START, ‘EAPOL-START’, AND THE ENDING, ‘EAPOL-SUCCESS’ OR ‘EAPOL-FAILURE’, PACKETS. ONCE THESE MESSAGES HAVE BEEN LOCATED IT IS POSSIBLE TO DETERMINE THE TOTAL TIME IT TAKES FOR THE AUTHENTICATION PROCESS TO FINISH. BY SUBTRACTING THE TIME-STAMP ON THE ‘EAPOL-START’ WITH THE TIME-STAMP ON THE ‘EAPOL-SUCCESS’ PACKET THE TOTAL TIME CAN BE OBTAINED. THE REASON FOR SUBTRACTING THE START-TIME FROM THE FINISH-TIME IS BECAUSE IT IS NOT POSSIBLE, TO THE AUTHOR’S KNOWLEDGE, TO CONFIGURE WIRESHARK TO START ITS TIMER ON SPECIFIC PACKETS.

ONCE DATA HAD BEEN COLLECTED FOR THE DIFFERENT EAP-METHODS WHEN THE
network was “as-is” a point of reference could be established. With the reference data being established it was possible to start gathering data about the authentication process when a delay had been introduced in the network. Delay was introduced by using an application named DummyNet.

Once all data from experiments were gathered we also ran tests with large delays; such as 200 up to 1000 milliseconds just to observe if any irregularities would appear, or if the data would continue showing fairly linear results; it proved to be the latter.

The data, gathered for each case, was then analyzed to see if any patterns between processing times for the different entities and network delay would emerge. Each case was also compared to the reference case as well as with each other. Statistical analysis on the resulting data was performed in order to determine the validity of the results. How this was done is also explained in the coming chapter.
Experiments
Chapter 4

Implementation and experiments

This section will focus on how the experiments were conducted and which applications and devices were used. First a description of the laboratory environment will be given, after that the experiments will be depicted and explained.

4.1 Lab Environment and Network

The lab environment consists of four different entities; a client, a switch, a traffic shaper and a RADIUS-Server. These entities and their respective roles will be explained below. A number of applications were used for specific purposes, these applications and their purposes will also be described.

4.1.1 Client - The Supplicant

As we explained in an earlier chapter a supplicant is a device supporting IEEE 802.1x, that wants to connect to the network.

For this environment a PC was used that was running Microsoft Windows XP as its operating system (OS). One major problem we encountered with Windows XP was that it’s natively limited to very few EAP-Method types for use in the authentication process. In order to be able to measure performance for a wider array of EAP-Methods we used XSupplicant for connecting to the network instead of using Windows XP’s native client. XSupplicant supports all recognized EAP-Methods.

The PC was a Dell with the following system specifications:

- CPU: 2.0 GHz
- RAM: 2 GB
- OS: Microsoft Windows
- Supplicant Software: XSupplicant
4.1.2 Switch - The Authenticator

The role of the authenticator is to relay packets between the supplicant and the authentication server, and vice versa. It is also responsible for monitoring for “Success” and “Failure” packets and to set the port state accordingly. For the role of authenticator a Cisco Catalyst 2950 switch with 48 ports running IOS 12.1(22)EA13. To emulate the live environment Björn envisioned as closely as possible the switch was configured to have five different VLANs. These five VLANs correspond to groups with different access privileges that devices which successfully authenticated devices will be placed in depending on the authentication credentials used.

Since the switch was to act as an authenticator it was configured to support IEEE 802.1x and to communicate with the RADIUS-server. A single port was configured as a monitor port to observe all traffic that passes through the switch.

The running configuration of the switch is listed in appendix A.

4.1.3 RADIUS-Server - The Authentication Server

The authenticator server handles all the processing of the authentication. Its purpose is to determine whether the devices wanting to connect are to be granted or denied access to the network. For this role we chose to use a PC running Debian 4.1. There are a number of different software that can be used to enable the device to act as an authentication server. However, we decided to go with an open source application called freeRADIUS [10]. The reasons for choosing this over other applications are that it is free and very well documented.

The PC was a Dell with the following system specifications:
- CPU: 2.0 GHz
- RAM: 2 GB
- OS: Debian 4.1.2-25
- Software: freeRADIUS

4.1.4 Traffic Shaper

Since a major part of this thesis revolves around measuring performance when a delay has been introduced in the network, we needed to emulate this phenomenon. This was done by introducing yet another entity in our environment; the traffic shaper. The purpose of the traffic shaper in the lab environment is to introduce delay between the authenticator and the authentication server. For the traffic shaper to function properly it had to have two network interface cards (NICs) installed, because it has to be able to forward all traffic between the two end-points, i.e., act as a bridge.

A Debian Lenny system running brctl [6] and DummyNet [9] was used as a traffic shaper. Configuration of the two NICs to act as a bridge was done
using brctl, which is part of the standard repository in Debian Lenny. To achieve the shaping functionality we used an open source application named DummyNet.

The PC acting as traffic shaper had the following system specifications:
- CPU: 2.0 GHz
- RAM: 2 GB
- OS: Debian Lenny 4.1.2-25
- Software: DummyNet, brctl

### 4.1.5 Tools

This section will give a brief explanation of the tools that were used in our experiments.

**XSupplicant**

XSupplicant [20] is an open source application that implements the IEEE 802.1x protocol and allows a computer to authenticate with a RADIUS server with EAP. It works for both wired and wireless LAN connections. XSupplicant has support for the following EAP types:

- EAP-MD5
- LEAP
- EAP-MSCHAPv2
- EAP-OTP
- EAP-PEAP (v0 and v1)
- EAP-SIM
- EAP-TLS
- EAP-TNC
- EAP-TTLSv0 (PAP/CHAP/MS-CHAP/MS-CHAPv2/EAP)
- EAP-AKA
- EAP-GTC
- EAP-FAST
WireShark

Wireshark [32] is an open source application used to monitor and analyze network packets, it is able to identify a variety of different packets from different networking protocols, and is able to display encapsulation and packet fields. This makes it a very valuable tool for network troubleshooting, analysis and protocol development.

TCPdump

TCPdump [30], like Wireshark, is a packet analyzer and provide means to analyze performance and network traffic behavior etc. TCPdump is completely command-line based. It works on most Unix-like operating systems, there is also a version for windows called WinDump, and is distributed under the BSD license.

DummyNet

DummyNet [9] has a wide scope of use, including testing network protocols and bandwidth management. With DummyNet it is possible to emulate multi-path effects, delays and packet loss in the target network. Thanks to its wide area of use it has become popular in the research community [7].

DummyNet is a free software that runs on FreeBSD, Linux, Windows and OpenWRT.

brctl

brctl [6] is an application that is used to create, maintain and inspect Ethernet bridge configuration in the Linux kernel. It aggregates the NICs so that it appears to be one physical connection to all outside devices.

4.2 Experiment 1: Measuring of Authentication Time

The purpose of the first experiment is to determine if there are any differences between different EAP-Methods regarding the total time for authenticating the supplicant. Measurements were taken from the point the supplicant started the authentication process by connecting to the network using XSupplicant until it has been successfully authenticated and received a ‘Success Packet’ from the authenticator.

Method

We started by configuring XSupplicant to use the EAP-Method we wanted to measure. Since this experiment is about determining if there are any dif-
ferences in processing time between different EAP-Methods a way to record the time from the starting of the process to the ending of the process had to be determined. To do this we used Wireshark to observe the traffic being sent between the supplicant and the authenticator. This allowed the whole process to be monitored and the useful data to be extracted. By comparing the time-stamp of the EAPOL-Start packet leaving the supplicant with the time-stamp on the EAPOL-Success that arrived at the supplicant we were able to get the time it took to complete one authentication. By subtracting the start time from the end time the total time for the authentication process could be determined. This means that the total time can be explained like this:

\[ t_{\text{total}} = t_{\text{success}} - t_{\text{start}} \]

where \( t_{\text{success}} \) is the time-stamp on the incoming ‘EAPOL-Success’ packet and \( t_{\text{start}} \) is the time-stamp on the ‘EAPOL-Start’ packet.

Once the ‘EAPOL-Success’ message had arrived at the supplicant we terminated the connection using XSupplicant; when closing an active session the supplicant issues an ‘EAPOL-Logoff’ message to the authenticator. Once the session had been terminated we repeated the process. This was performed 40 times and from the recorded data a mean value was calculated:

\[ t_{\text{mean}} = \frac{t_{\text{total}}}{n} \]

where \( n \) is the number of times the process was run, which in our case was 40.

This process was repeated for all different EAP-Methods that were to be compared.

DummyNet, that was installed on the traffic shaper, was used to introduce a delay in the network. It was configured to apply a delay of 10 millisecond. Once the delay had been properly configured we did the steps explained earlier again to get data corresponding to a 10ms delay in the network. After we had collected this data for all different EAP-Methods the delay was incremented in steps of 10 until we had measurements for all EAP-Methods from 0 milliseconds up to 100 milliseconds.

### 4.3 Experiment 2: Evaluation of Processing Time

As part of the performance evaluation processing times for the different entities; supplicant, authenticator and authentication-server, were examined. This was done to determine how much of the total time was due to the processing time of the different entities and to see if there was a pattern between processing time and the complexity of the EAP-Method.

**Method**

Different authentication methods have different complexity; some, like EAP-MD5, require no setup of a secure tunnel between the supplicant and the authenticator while other methods, like EAP-TLS, base their security around
this. Because of this we were interested in evaluating if there were any significant differences between the processing times, and if there was any pattern between the processing time and the complexity of the EAP-Method at the three entities. To be able to measure the processing times at the different entities we used a similar approach as in the first experiment. By monitoring specific packets at the different entities it was possible to calculate the time it took for the supplicant, the authenticator and the authentication server to process requests.

Similar to how the first experiment was conducted we used the time-stamps on the packets to determine the time it took for each entity to process the requests. Wireshark was setup to monitor for different packets depending on which entity was being examined. In the case of the authenticator Wireshark monitored for the first packet that arrived from the supplicant, and until that packet had been encapsulated and relayed to the authentication server. By comparing the time-stamps on these two packets it was possible to derive the total amount of processing time on the authenticator. On the authentication server TCPdump was used to monitor and record RADIUS packets, from the first packet arriving from the authenticator until the authentication server had processed the request and sent its response back to the authenticator.

By using Wireshark to extract these types of packets it was possible to compare the time-stamps on the incoming packet with the outgoing packet and calculate the total processing time for the authentication process.

4.3.1 Validity Threats

By introducing yet another entity, the traffic shaper, in the topology we added a somewhat unnatural element in the system. This might have negative impact on the accuracy of the system. However, since this element is present during all measurements, even when no delay was introduced, we consider this to be neglectable. Furthermore, because of the fact that all measurements were carried out manually instead of being automated we settled for a smaller number of batches, as opposed to if we had used an automated approach. However, according to [18] 40 batches are enough, if the confidence interval and the standard deviation are small enough. Since our results conform with this statement, i.e. both the confidence interval and the standard deviation are in range, the decision was taken to not use more batches.

Wireshark was used in order to capture packets and record the time-stamps, however, there is a chance that the packets don’t always arrive at Wireshark first: some packets may arrive at XSupplicant before Wireshark and vice versa. This means that the exact time on the captured packets may vary slightly. However, since this variation is quite small we do not think it has a huge impact on the results.
Results
Chapter 5

Results

In this section the data collected in the experiments will be presented and explained. All calculations used will be explained in detail.

5.1 Evaluation of authentication

Here will follow the results for the first experiment measuring the authentication time. The results will be presented first for each method individually then there is a comparison of the methods visualizing the differences between the regression line for each authentication method.

The results are presented in the form of a graph where the $x$-axis represents the introduced delay and the $y$-axis is the mean authentication time. For each data point there is also an error bar representing the 95% confidence interval which shows in what interval 95% of all data samples are. There is also a regression line showing the linearity of the mean.

The regression line will have the equation

$$y = kx + m$$  \hspace{1cm} (5.1)

where

$$m = \overline{y} - k\overline{x}$$  \hspace{1cm} (5.2)

and where $\overline{x}$ and $\overline{y}$ are the mean value of all known $x$ and $y$ respectively, $m$ is the point where the regression line crosses the $y$-axis and $k$ shows the slope of the regression line where

$$k = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sum(x_i - \overline{x})^2}$$  \hspace{1cm} (5.3)

Table 5.1 shows $k$ and $m$ for the regression line of each authentication method in the graph, together with value of the coefficient of determination.
CHAPTER 5. RESULTS

\[ r^2, \] which is a statistical measure of how well a regression line approximates real data points; an \( r^2 \) of 1.0 (100\%) indicates a perfect fit. The formula for \( r \) is:

\[
r(x, y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}} \tag{5.4}
\]

\( r^2 \) for all methods is between 99.6\% and 99.8\%, which means that the regression lines are good approximations for all methods.

Table 5.1: number of transmissions between the authenticator and the authentication server

<table>
<thead>
<tr>
<th>Method</th>
<th>( m )</th>
<th>( k )</th>
<th># of transmissions</th>
<th>( r^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAP-MD5</td>
<td>0.11</td>
<td>5.96</td>
<td>6</td>
<td>0.996</td>
</tr>
<tr>
<td>EAP-TTLS CHAP</td>
<td>0.25</td>
<td>11.87</td>
<td>12</td>
<td>0.996</td>
</tr>
<tr>
<td>EAP-TTLS PAP</td>
<td>0.25</td>
<td>11.85</td>
<td>12</td>
<td>0.997</td>
</tr>
<tr>
<td>EAP-TTLS MSCHAPv2</td>
<td>0.28</td>
<td>13.73</td>
<td>14</td>
<td>0.998</td>
</tr>
<tr>
<td>EAP-TTLS MD5</td>
<td>0.29</td>
<td>13.77</td>
<td>14</td>
<td>0.997</td>
</tr>
<tr>
<td>PEAP MSCHAPv2</td>
<td>0.31</td>
<td>18.26</td>
<td>18</td>
<td>0.995</td>
</tr>
</tbody>
</table>
5.1.1 MD5

Figure 5.1 shows the mean authentication time for EAP-MD5, the added regression line has \( k \) equal to 5.96 and \( m \) equal to 0.11. \( r^2 \) is equal 0.996 giving a fit of 99.6% which means that the regression line is a good approximation for the authentication time for MD5.

Figure 5.1: MD5 authentication time

![MD5 authentication time graph](image)

5.1.2 TTLS-PAP

Figure 5.2 shows the mean authentication time for TTLS-PAP, the equation for the regression line is different from MD5, \( k \) is 11.85 and \( m \) is 0.25, so the authentication time of TTLS-PAP is almost twice as high as for MD5. The \( r^2 \) value for this method is also very high; 0.997 indicates a fit of 99.7%.

5.1.3 TTLS-CHAP

The graph for TTLS-CHAP, shown in Figure 5.3, is very similar to the one for TTLS-CHAP. Actually both \( k \) and \( m \) are the same for the two methods, namely 11.85 and 0.25 respectively, which indicates that the two methods are very similar. Also here is \( r^2 \) very high; 0.996 indicates a fit of 99.6%.
CHAPTER 5. RESULTS

Figure 5.2: TTLS-PAP authentication time

![TTLS-PAP authentication time graph]

\[ y = 11.852x + 0.2502 \]
\[ R^2 = 0.9968 \]

Figure 5.3: TTLS-CHAP authentication time

![TTLS-CHAP authentication time graph]

\[ y = 11.869x + 0.2517 \]
\[ R^2 = 0.9958 \]
5.1.4 TTLS-MD5

Figure 5.4 shows the mean authentication time for TTLS-MD5. Just as previous graphs the regression line seems to have a very good fit; \( r^2 \) here is 0.997, giving a fit of 99.7%. \( k \) for the regression line is 13.77 and \( m \) is 0.29.

![Figure 5.4: TTLS-MD5 authentication time](image)

\[
y = 13.768x + 0.2919 \\
R^2 = 0.9972
\]

5.1.5 TTLS-MSCHAPv2

In figure 5.5 the mean authentication time for TTLS-MSCHAPv2 can be seen. \( k \) for the regression line is 13.73 and \( m \) is equal to 0.28. Just as for the previous graphs; \( r^2 \) for the regression line is very high. In this case it is 0.998 which indicates a fit of 99.8%.

5.1.6 PEAP-MSCHAPv2

The mean authentication times for PEAP/MSCHAPv2 is shown in figure 5.6, even though there are some fluctuation on the mean values the graph is still has a linearity and the \( r^2 \) for the regression line is still very close to 1 namely 0.995 giving a fit of 99.5% \( k \) is in this case 13.73 and \( m \) is 0.28.
CHAPTER 5. RESULTS

Figure 5.5: TTLS-MSCHAPv2 authentication time

\[ y = 13.73x + 0.2837 \]
\[ R^2 = 0.9976 \]

Figure 5.6: PEAP-MSCHAPv2 authentication time

\[ y = 18.263x + 0.3094 \]
\[ R^2 = 0.9947 \]
5.2 Evaluation of Processing Time

This part presents the results of the second experiment. First there will be a presentation of the process time for each of the devices then the total process time for each method will be presented.

5.2.1 Supplicant Processing Time

Figure 5.7 shows the mean processing time depending on the EAP-method at the supplicant. The y-axis shows the mean processing time. And on the x-axis we have the different methods. It also displays the standard deviation for the population.

We can see that the process time for EAP-MD5 is fairly much lower than the other methods. But except for that it is difficult to find any pattern in the data.
5.2.2 Authenticator Processing Time

Figure 5.8 shows the mean processing time for the authenticator. The processing time is on the y-axis and on the x-axis is the methods. The standard deviation is a bit lower in this case than the previous table, but still it is quite high. However, it is noticeable that the mean time for TTLS-CHAP and TTLS-PAP are almost the same, and so is the case for TTLS-MD5 and TTLS-MSCHAPv2 as well. As in the previous graph the processing time for MD5 is also lower than the rest.

![Figure 5.8: Authenticator Processing Time](image-url)
5.2.3 Server Processing Time

The mean processing time for the authentication server is presented in Figure 5.9. The y-axis and x-axis represent the same as in previous figures. The standard deviation for the different populations in this measurement is relatively low. In this measurement there is a even clearer difference between EAP-MD5 and the other methods.

![Figure 5.9: Server Processing Time](image-url)
Figure 5.10 shows the results for the different entities added together to a total mean time. The dark blue part is the mean processing time for the authenticator. The lighter blue is for the supplicant and the lightest top part is the server. Here it is visible that the processing time for the server is significantly lower than for the supplicant and the authenticator.
Discussion
Chapter 6

Discussion

6.1 Discussion

To be able to control the access to a network, one could implement port based access control (802.1x), doing so will give the network administrator the possibility to control what devices or persons are allowed to access the network and resources on it. The implementation of 802.1x requires use of one or more authentication methods to authenticate the devices trying to access the network. When deciding what type of authentication process to adopt there are a number of parameters to consider. One needs to consider what level of security is to be achieved, are there access points (wired or wireless) that are physically accessible for outsiders. What is the cost of deployment. Is it necessary to purchase any new equipment or is it possible to use the existing one. How to keep track of the credentials, is there already an AD in the system, how are the credentials stored. Is it necessary to purchase any third party software. Are there a number of clients that need to be configured and if so, how do we do that or can the users do it themselves.

Considering transparency there are a number of questions raised: should the authentication be visible for the user? Should they have to authenticate manually or should it be an automated process? If the authentication is automated what is the time that the authentication may take? How long will it take before users, unaware of the ongoing authentication process, will give up and disconnect?

Even though this report will not answer all of these questions, it should be able to cast some light on how one can compare different methods and the authentication time based on a few parameters.

Looking at the graphs presented earlier in this report one can see that the mean authentication time for each of the methods follow a linear pattern. For the regression lines applied to the data of the methods, all have an $r^2$ above 99,5%, thus indicating a very good fit.
The equation of the linear regression line, has the form

\[ y = kx + m \]  \hspace{1cm} (6.1)

which is the equation for a straight line. The regression line for the different methods have different values for the \( k \) and \( m \) value and at first glance it might seem as if there is no connection between the different regression lines. However, if we analyze the coefficient \( k \) and \( m \), we should be able to come to a different conclusion.

In our system, we had control of the delay and knew all the time what amount was added. But in real life one would not know the absolute delay in a system and therefore would have to have a reference \( x_{ref} \) for this delay, so that \( x \) can be expressed in the form \( x = x_m - x_{ref} \), where \( x_m \) is the total delay for the measured time. The delay in the system at a point in time used as reference is denoted by \( x_{ref} \), in this case it is when no delay has been added to the system by the traffic Shaper. During our experiments \( x_m \) is equal to \( x_{ref} \) plus the delay added by the TS, and \( x \) is the time introduced at the TS.

then

\[ x_m = x_{ref} + \Delta x \]  \hspace{1cm} (6.2)

for

\[ 0 < x_m \]

This since there can be no total delay less than 0, that would mean that the receiver receives the data before it has been sent.

In a real life scenario, one would not know the value of \( x \), but one could calculate it knowing \( x_m \) and \( x_{ref} \) by using equation 6.2. It is worth pointing out that \( \Delta x \) could be negative since there can be less delay on the network compared to the reference case.

If we then look at the equation 6.1 knowing the equation of a straight line we know that \( m \) in the equation represents the value \( y \) have when the line crosses the y axis, i.e. when

\[ x_{ref} = x_m \]

Table 6.1 compares \( m \) for the regression lines for each method with the mean authentication time without delay added. It also shows the mean process time for each method. It can be seen in the table that there is a small difference between the processing time and the authentication time. This indicates that in this case, with no delay added, the transmission time is a minor part of the authentication time. Still, the authentication time is a good approximation of \( m \), even the total processing time might be a
A good approximation for $m$. However, further research would be necessary to find out if there is any correlation between the processing time/transmission time and $m$.

### Table 6.1: $m$ and $k$ for different methods

<table>
<thead>
<tr>
<th>Method</th>
<th>$m$</th>
<th>Process time</th>
<th>Auth time</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAP-MD5</td>
<td>0.11</td>
<td>0.11</td>
<td>0.12</td>
</tr>
<tr>
<td>EAP-TTLS CHAP</td>
<td>0.25</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td>EAP-TTLS PAP</td>
<td>0.25</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td>EAP-TTLS MSCHAPv2</td>
<td>0.28</td>
<td>0.23</td>
<td>0.27</td>
</tr>
<tr>
<td>EAP-TTLS MD5</td>
<td>0.29</td>
<td>0.25</td>
<td>0.27</td>
</tr>
<tr>
<td>PEAP MSCHAPv2</td>
<td>0.31</td>
<td>0.28</td>
<td>0.26</td>
</tr>
</tbody>
</table>

$k$ in the equation 6.1 is the slope of the regression line. This is by which rate $y$ is increasing (if $k$ is negative $y$ is decreasing) depending on $x$, this corresponds to how many times the delay is added to the total authentication time. Since the delay is added one time per round trip, we could expect that the $k$ value would correspond to the total number of round trips taking place during the authentication process.

Table 6.2 shows $k$ for the authentication methods together with number of transmissions, and here we can see that $k$ from the regression line corresponds very well with the number of transmissions during the authentication process.

### Table 6.2: number of transmissions

<table>
<thead>
<tr>
<th>Method</th>
<th>Transmissions</th>
<th>$k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAP-MD5</td>
<td>6.00</td>
<td>5.96</td>
</tr>
<tr>
<td>EAP-TTLS CHAP</td>
<td>12.00</td>
<td>11.87</td>
</tr>
<tr>
<td>EAP-TTLS PAP</td>
<td>12.00</td>
<td>11.85</td>
</tr>
<tr>
<td>EAP-TTLS MSCHAPv2</td>
<td>14.00</td>
<td>13.73</td>
</tr>
<tr>
<td>EAP-TTLS MD5</td>
<td>14.00</td>
<td>13.77</td>
</tr>
<tr>
<td>PEAP MSCHAPv2</td>
<td>18.00</td>
<td>18.26</td>
</tr>
</tbody>
</table>

If then $k$ is replaced with the number of transmissions, and $m$ with mean authentication time with no delay added, we have a way to approximate the expected authentication time based on the relative increase of delay.

Figure 6.1 show the approximated values for authentication time, the lines, calculated using the equation 6.1 together with the calculated values from the experiment (dots).

In the figure it can be seen that the approximated regression lines fit well, which would indicate that the calculation is a good approximation for the authentication time.
Figure 6.1: Comparison of approximated and actual authentication time
Conclusions
Chapter 7

Conclusions and future work

This report shows a study of performance on different EAP authentication methods. Experiments were carried out on different authentication methods to find out their sensitivity to delay. By adding delay to the system, the mean authentication time was calculated for different delays. Based on the data gathered we found a pattern; we saw a very clear linear relation between the mean authentication time and the delay in the system. From these results, a general way of approximating the authentication time based on differences in delay was established. The linear relation between delay and authentication time can be approximated by using the number of transmissions and the processing time. This way it can be numerically determined how a specific EAP authentication method responds to a certain increase in delay.

For a company, knowing the approximate delay for low and high traffic intensity on the network, it can be highly valuable to know the reaction of the authentication method before deploying it. If the network often has high traffic intensity resulting in high delays, the company might have to compromise by using an authentication method with lower security level, but that is less susceptible to growth in delay.

Even though there are some consistent results from these studies, there are more work that can be done in this area. Maybe the most interesting studies would be how the authentication time is experienced by users. Will users, when unaware of the authentication time, get annoyed and wonder if there is something wrong?

Furthermore, it is interesting to see how users react to the differences in delay if they are aware of the authentication process but not have to take part in it themselves, i.e. they know it happens but the process is automated so they don’t have to take any actions to start the process or to authenticate. and compare it with how users react to the differences in time if they have to initialize the authentication process themselves.

There are more authentication methods available than those covered by
this report, some of which even require the use of additional hardware, e.g. card reader, cellular phones, etc. Naturally, this study could be extended to see if the approximation established in this report also applies for those methods.
Appendix
Appendix A

Apendix

Current configuration : 9189 bytes
!
version 12.1
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname leSwetch_1
!
aaa new-model
aaa authentication dot1x default group radius
aaa authorization network default group radius
enable password kallejo
!
username peter password 0 kallejo
ip subnet-zero
!
!
spanning-tree mode pvst
no spanning-tree optimize bpdu transmission
spanning-tree extend system-id
no spanning-tree vlan 1,10,101,201,301,666,801
dot1x system-auth-control
dot1x guest-vlan supplicant
!
!
interface FastEthernet0/1 - 10
switchport access vlan 10
switchport mode access
spanning-tree portfast
!
interface FastEthernet0/11
switchport access vlan 101
spanning-tree portfast
!
interface FastEthernet0/12
switchport access vlan 201
spanning-tree portfast
!
interface FastEthernet0/13
switchport access vlan 301
spanning-tree portfast
!
interface FastEthernet0/14
switchport access vlan 666
spanning-tree portfast
!
interface FastEthernet0/15
switchport access vlan 801
spanning-tree portfast
!
interface FastEthernet0/16
!
interface FastEthernet0/17 - 48
switchport mode access
dot1x port-control auto
dot1x timeout reauth-period 20
dot1x guest-vlan 801
dot1x reauthentication
dot1x auth-fail vlan 666
spanning-tree portfast
!
interface GigabitEthernet0/1
!
interface GigabitEthernet0/2
!
interface Vlan1
no ip address
no ip route-cache
shutdown
!
interface Vlan10
ip address 194.47.131.201 255.255.255.0
no ip route-cache
!
interface Vlan101
ip address 10.0.1.2 255.255.255.0
no ip route-cache
shutdown
!
interface Vlan201
ip address 10.0.2.2 255.255.255.0
no ip route-cache
shutdown
!
interface Vlan301
ip address 10.0.3.2 255.255.255.0
no ip route-cache
shutdown
!
interface Vlan801
ip address 10.0.8.2 255.255.255.0
no ip route-cache
shutdown
!
ip http server
no cdp run
radius-server host 194.47.131.195 auth-port 1812 acct-port 1813 key abc
radius-server retransmit 1
radius-server timeout 2
radius-server key abc
radius-server vsa send authentication
!
line con 0
logging synchronous
line vty 5 15
!
!
monitor session 1 source interface Fa0/3, Fa0/33 - 48
monitor session 1 destination interface Fa0/16
Bibliography


[21] Bruce Potter. 802.1x What it is, How it’s broken, and How to fix it. 2002.


