Bachelor's Thesis Computing Science

Radboud University Nijmegen

Trusting Eduroam

Analysing Eduroam's security

Author:
Niek van den Kieboom
s1106321

First supervisor/assessor:
dr. A.A. Westerbaan

Second assessor:
dr. ir. X.J.G. de Carné de Carnavalet

August 21, 2025

Abstract

Eduroam plays an important role in allowing users to access an institute's Wi-Fi network, namely the authentication step of the process. Before connecting to Eduroam, your operating system may prompt you to check a server certificate, without any guidance on how to determine if this certificate is valid. Radboud's website* states that you should click on "Trust" to accept this certificate. In this thesis, our aim is to figure out if this poses a security risk. We will see that when the certificate is not properly checked an attacker may retrieve user passwords in some scenarios using an 'evil twin' access point. This is due to some security issues that are present in a protocol called 'MSCHAPv2' which is commonly used in Eduroam setups. We find that there are solutions to these problems such as configuration tools (CAT and geteduroam) and alternative protocols, but these have not yet been widely adopted.

*https://www.ru.nl/en/services/campus-facilities-buildings/ict/wi-fi/setting-up-wi-fi-eduroam/set-up-wi-fi-eduroam-on-campus

Contents

Chapter 1

Introduction

Eduroam, originally created by SURFnet, plays a critical role in enabling secure and seamless authentication to wireless networks across educational institutions. Since its small inception in 2002, releasing only in a few European countries, it has grown into a globally used service that is available in more than 39,000 locations in 106 countries [7], reaching 8.4 billion authentications in 2024*. With so many locations and active users, it is important that the network protocols used in Eduroam are secure.

Eduroam stands for "education roaming", and aims to provide users access to Wi-Fi, not only at their their own educational institute, but also when they visit another institute which uses Eduroam, much like data roaming across country borders in mobile telephony. This is achieved through a hierachrical network of **RADIUS** servers, through which the user communicates with its own institute's RADIUS server.

In some cases, **certificate**s provided by institutes' RADIUS servers to set up a TLS connection have to be manually checked by users to make sure that they are communicating with their own institute's RADIUS server. Contrary to the web public key infrastructure, this can not be done automatically. This puts the burden of user authentication on the users themselves by requiring them to check the server certificate, which already assumes that an average user knows what they have to look for to ensure they are connecting to a real Eduroam service point. The reality, however, is that most users are unwilling to spend time manually checking a server certificate, let alone know what a certificate is, which could result in them connecting to a rogue network. Radboud states on their official website that you should accept this certificate, without any exceptions [16]. How problematic is this and, if it is, what has and can be done about it?

Whilst the process of accessing an Eduroam network is relatively easy, the process of authenticating a user in Eduroam behind the screens is extensive and involves several protocols, where the choice of protocols used partly lies

*https://eduroam.org/eduroam-hits-a-new-record-8-4-billion-authentications-in-2024/

in the hands of the institution setting up the network. In a commonly used setup, the authentication consists of three phases: the RADIUS protocol, **outer authentication** and **inner authentication**. Since 2002, protocols that have been proven to be weak have been available as a choice in the inner authentication phase, and are still actively used; see table 3.1. One of these protocols, which is used by Radboud, is the Microsoft Challenge Handshake Authentication Protocol version 2, or **MSCHAPv2** for short [22]. This protocol was created in the year 2000 and has some questionable design choices that were acceptable at that time, but have become a major problem over the years when looked at in a vacuum. To alleviate the problem of the inner authentication protocols being weaker, the outer authentication phase sets up a TLS tunnel between client and RADIUS server. chapter 2 shortly touches upon previous work that is closely related to this thesis, and partially inspired it. Then Chapter 3 of this thesis talks about Eduroam's underlying protocols, examples of specific protocols used, and how they work. It also looks at MSCHAPv2 and the security issues that accompany it. Section 3.1 gives an overview of how Eduroam works without going into details of the protocols that are used. Section 3.2 is the first section where the details of the RADIUS protocol are described and how it is implemented in Eduroam. Section 3.3 and Section 3.4 then continue with an explanation of the outer and inner authentication protocols used in Eduroam, which serve the purpose of setting up a secure communication channel and communicating client credentials securely, respectively. Then section 4.1 discusses the security issues of the Microsoft Challenge Handshake Authentication Protocol. Following this, section 4.2 shows an actual attack on a fake eduroam network and how credentials can be extracted from this. Then section 4.3 talks about configuration tools which are solutions made for the problem of users not willing to check server certificates, in addition to providing an easy way for users to set up their devices to run Eduroam on. Lastly, section 4.4 discusses an alternative protocol which removes the need for passwords to be sent between a client and a server and does authentication through mutual authentication.
At the end of this thesis, you will be more aware of the risks to which you are exposed when connecting to an Eduroam network, but also of what has been done to mitigate these risks and what steps might still have to be taken.

Chapter 2

Related Work

Eduroam has been in use for over 20 years, and since security is important it has been analysed before. This thesis can be considered a follow-up of Matthias Ghering's bachelor thesis, who was also a student at the Radboud University who published his work in 2016, named "Evil Twin vulnerabilities in Wi-Fi networks" [9]. This thesis focuses on evil twins in three types of networks: public networks, private networks and enterprise networks (such as Eduroam) and also attacks these networks. The attack on the enterprise network described in Matthias' thesis is similar to the attack used in this thesis as it uses the same vulnerabilities in the MSCHAPv2 protocol. At the time that Matthias' thesis was written, tools such as geteduroam (which we will see in section 4.3) did not yet exist and have not yet been reviewed for security purposes. Because of this time difference, we hope that we can come up with a different solution for how security in Eduroam can be handled. One conclusion of [9] was that institutes should switch their inner authentication protocol from PAP to MSCHAPv2. Although MSCHAPv2 is more secure than PAP, it is currently still not a fully secure option, and there have been developments over the last 9 years like the implementation of EAP-TLS into Eduroam (which we will see in section 4.4), which is a different protocol from what is commonly used now in Eduroam, but is more secure.

Another paper that is closely related to this thesis is called "Careful with that Roam, Edu"[19], which was published in 2022. This paper demonstrates an actual attack on students and professors and shows that an attack using an evil twin works near a real Eduroam Wi-Fi network. This paper talks about "recovering credentials" without actually going into detail on how this actually works, presumably to keep the research more general. This thesis will focus on actually showing how credentials are recovered after capturing authentication packets, as we believe that this is not a step that should be brushed over if you want to show the possible dangers of connecting to an

Eduroam Wi-Fi network.

Chapter 3

Preliminaries on Eduroam

Before we perform an attack on an Eduroam network, it is important to understand how Eduroam works. Eduroam can be seen as an authentication protocol that is in charge of authenticating users wanting to access an institute's Wi-Fi network. There is a difference in how Eduroam works based on the preference of an institute, since there is some freedom in the choice of which protocols are used. Some institutes have wifi-portals to access the network, whilst others just require the user to fill in their credentials in their Wi-Fi list on their device. We will see that the freedom of choice, in this case, may lead to some security issues.

3.1 Architecture of Eduroam

Eduroam can be seen as a huge collection of Wi-Fi networks sharing the same name (or technically, the SSID). A network that is part of this collection is set up as a WPA2-Enterprise network. The main differences between WPA(2)-PSK and Eduroam's WPA2-Enterprise are that with Enterprise networks, the users log in using their username and password (usually via IEEE 802.1X and RADIUS, see section 3.2), while with WPA(2)-PSK users log in using a pre shared key (the "Wi-Fi password").

When a user wants to connect to an Eduroam network, the following happens:

1. The user connects to the network that has the "eduroam" SSID on their device and the client initializes the connection to the AP using IEEE 802.1X. At this point the client can send frames to the access point, but is not yet connected to the internet.

2. The client starts EAP over (wireless) LAN which encapsulates requests between the client and the AP as described in RFC 3579 [2]. These requests contain either the identity of the user or the **anonymous**

identity, which allows users to send initial access requests without revealing their username and password to the visited network.

3. The AP receives the request and decapsulates it to recover the user's identity. This request is then encapsulated using RADIUS and sent into the RADIUS hierarchy (see section 3.2) to reach the user's home authentication server.

4. When the request reaches the user's home authentication server through the RADIUS hierarchy, the outer authentication protocol (see section 3.3) is initialized which has as purpose to establish a TLS tunnel between client and home authentication server. At this point, either the client automatically verifies the certificate sent by the authentication server (if the certificate is already known because, for example, the user has already connected to the home authentication server before), or the user may be asked to verify it manually.

5. After the TLS tunnel is established, the inner authentication protocol (see section 3.4), such as MSCHAPv2 (see section 3.4.2), is started which authenticates the user towards the home authentication server.

6. The access point is informed that the user has successfully authenticated via RADIUS/EAP, which means that the user is now allowed to access the network at the institution they requested access at. Due to the roaming nature of Eduroam, this can be any institute making use of Eduroam.

3.2 RADIUS

Remote Authentication Dial-In User Service (RADIUS) is a networking protocol used in Eduroam which authenticates and authorizes users to access a network that an institute provides. A basic RADIUS authentication and authorization setup involves a client, a network access server (NAS) or an access point (AP) (there is a slight difference in these two, but usually an AP acts as a NAS) and a RADIUS server which stores user credentials. This simple setup, with only one access point and RADIUS server that may be used in a small office, might look like this:

1. The user's client sends a request to a NAS or an AP. This request might contain user credentials based on the chosen authentication methods.

2. The NAS or AP forwards this request to the RADIUS server which checks the validity of the request (if the request contains legitimate credentials and comes from a valid source).

3. The RADIUS server responds to the request by sending an accept response, a reject response or a challenge response to the client. This challenge is usually sent to verify that the user is actually who they are claiming to be, but is not always sent in non challenge-response protocols such as PAP (see section 3.4.1).

3.2.1 RADIUS in Eduroam

Eduroam allows a user to access an institute's Wi-Fi network without that institute being the user's home institute. This is possible thanks to the RADIUS protocol being implemented in Eduroam.

The so called **trust fabric** of Eduroam consists of a hierarchy of RADIUS servers which are located on organizational, national or global level [20], see fig. 3.1 for an example. Requests are routed through a chain of RADIUS servers with the destination being the user's home RADIUS server, also called the Identity Provider (IDP).

For a local request, the flow is as follows (example: s1234567@ru.nl requesting network access on Radboud's campus):

1. A user's client sends an EAP access request, containing their credentials, to an AP located somewhere near them.
2. The AP forwards the request to the institute's AS (which is a RADIUS server).
3. The AS looks at the realm part of the user's email (ru.nl) to check if the user is a local user.
4. Since the user is local, the RADIUS server decapsulates the EAP request and verifies the credentials.
5. The RADIUS server instructs the AP to either accept or reject the access request.

For a visiting request, the flow is as follows (example: s1234567@ru.nl requesting network access on University of Amsterdam's campus):

1. A user's client sends an EAP access request, containing their credentials to, an AP located somewhere near them.
2. The AP forwards the request to the institute's AS (which is the RADIUS server).
3. The AS looks at the realm part of the user's email (ru.nl) to check if the user is a local user.

4. Since the user is not local (visiting), the request is forwarded to the .nl RADIUS server.

5. The .nl RADIUS server looks at the realm to see if it is in it's list of known realms. Since RU and UVA both use the .nl subdomain, the request is forwarded to the RU RADIUS server.

6. The RU RADIUS server decapsulates the EAP request, checks the credentials and backtracks the response to UVA's RADIUS server to either accept or reject the access request (which is done through the AP).

See also fig. 3.1.
If the top level domain (example: .nl) of the campus and user's credentials are not the same, the request is forwarded from the subdomain server to the root (.) server (the global RADIUS server).

When a request reaches the authentication server of a user through the trust fabric of RADIUS, the outer authentication protocol is initialized.

3.3 Outer authentication

The outer authentication protocol in Eduroam is used to establish a TLS tunnel between the client and authentication server. The outer authentication protocol used in Eduroam is **EAP-PEAP** or **EAP-TTLS** [9], both with the goal of setting up a TLS tunnel. Recent developments have also starting pushing the implementation of **EAP-TLS**, see section 4.4.

When a request reaches an authentication server, that is the home authentication server of the client, through the RADIUS protocol, the authentication server verifies the credentials of the user and sends over a certificate during the TLS handshake. This certificate has to be accepted by the client, and sometimes manually by the user. When the certificate gets accepted, a TLS tunnel is set up between the client and the authentication server, meaning that any requests sent from this point onward will be encrypted and can only be decrypted by the client and authentication server. Also, if the client/user properly checked the certificate, they can be sure that they are communicating with their home authentication server.

When the TLS tunnel is established, the client can "securely" send over the user's credentials for verification by the authentication server. This second step is called the inner authentication.

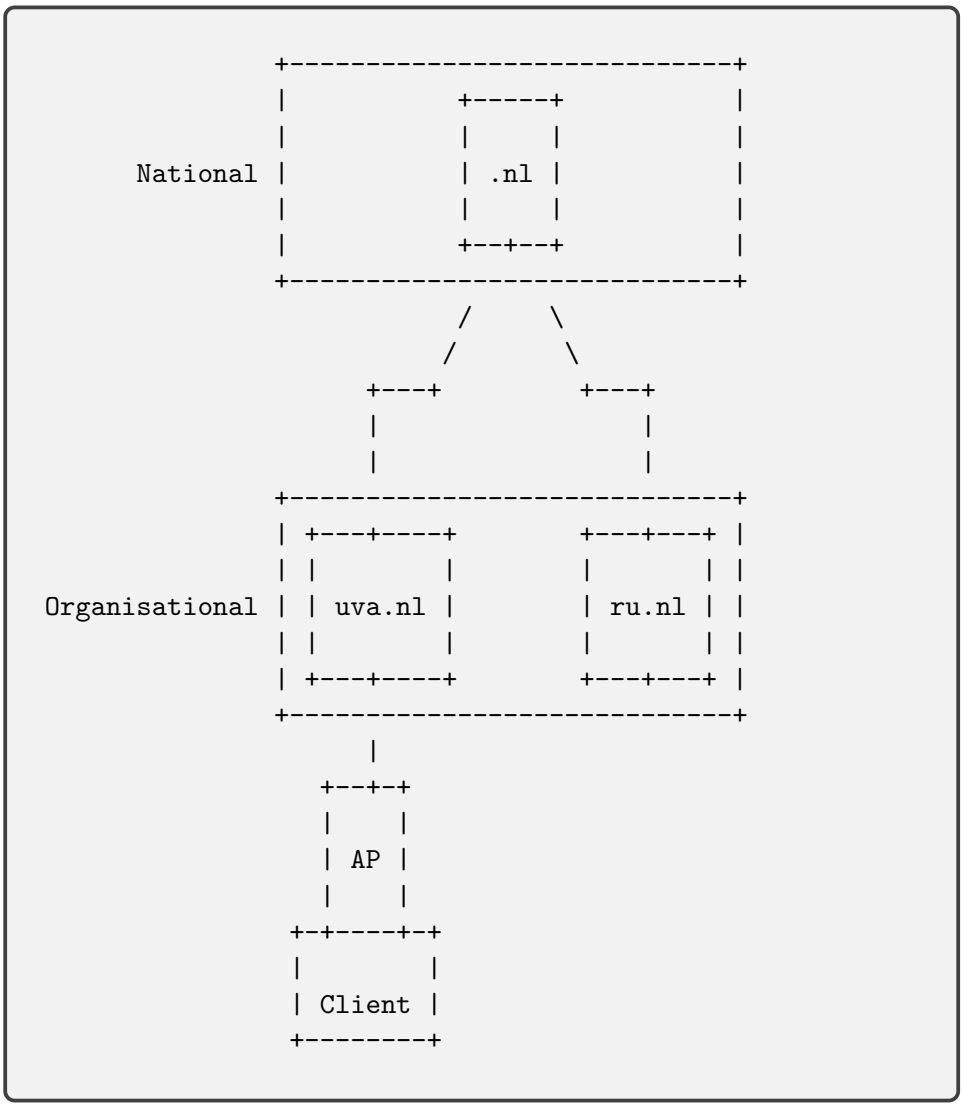

Figure 3.1: RADIUS hierarchy

3.4 Inner authentication

Eduroam supports multiple protocols to be used as the inner authentication protocol. Table 3.1 (an updated table of Table 4.4 from [9]. Protocols in Italics have changed since the original table was created) shows a list of institutes in the Netherlands and the protocols used for Outer and Inner authentication. It shows that MSCHAPv2 is the predominant choice for most institutes.

Institute	Outer authentication	Inner authentication
University of Twente	TTLS	*MSCHAPv2*
University of Amsterdam	*PEAP*	*MSCHAPv2*
Erasmus University*	PEAP	MSCHAPv2
Universiteit Leiden*	TTLS	PAP
Universiteit Utrecht	PEAP	MSCHAPv2
Universiteit van Tilburg*	PEAP, TTLS	MSCHAPv2, EAP-MD5
TU Delft	PEAP, TTLS	PEAP-MSCHAPv2, TTLS-PAP
Radboud Universiteit	PEAP	MSCHAPv2
HAN University	PEAP	MSCHAPv2
Wageningen Universiteit*	PEAP	MSCHAPv2

Table 3.1: Institute settings

3.4.1 PAP

Password authentication protocol, also known as PAP, is a simple protocol that can be used as the inner authentication protocol in Eduroam. The client sends over the username and password in plaintext (over the TLS tunnel created during outer authentication), and the RADIUS server sends back an accept or reject response. PAP is considered to be the least secure option, and is usually only utilized if the other protocols are unavailable.

3.4.2 MSCHAPv2

Radboud's RADIUS server uses the MSCHAPv2 protocol as its inner authentication protocol. It stands for Microsoft Challenge-Handshake Authentication Protocol version 2 and is an improvement on the CHAP and MSCHAPv1 protocols. A typical flow of the protocol is as follows (note that in the explanation below, RFC refers to RFC 2759 [22]), see fig. 3.2 for a visual explanation:

1. The authentication server (AS) generates a random 16 byte **server challenge** (**AuthenticatorChallenge** in RFC) and sends this over to the client.

2. The client generates a random 16 byte **challenge** (**PeerChallenge** in RFC).

*These results were taken from secondary sources which were not official institute websites.

3. The client calculates a **client challenge hash** (**Challenge** in RFC) by hashing the concatenation of the PeerChallenge, Authenticator-Challenge, and ASCII **client username** (**UserName** in RFC) using SHA-1. This hash is then truncated from 20 to the first 8 bytes.

4. The client calculates an **NTHash** (**PasswordHash** in RFC) using the MD4 algorithm on the UTF-16 **user password** (**clientPass** in RFC) in little endian mode without any byte order mark (BOM), see appendix A.1 for explanation why this is important in implementations, then pads this with zeros on the right to 21 bytes.

5. The client calculates three **DES** keys (see section 3.5) by splitting the PasswordHash into three 7 byte strings, then adding a parity bit to each 7 bit string to create three 8 byte **DES keys** (see section 3.5.2).

6. These DES keys are then used to encrypt the Challenge once per key, using single DES in ECB mode, see section 3.5, resulting in a 24 byte long digest called the **challenge response** (**NT-Response** in RFC).

7. The client sends the 24 byte NT-Response, the 8 byte Challenge and the UserName to the AS.

8. (The server proves to the client that it knows the password of the user.)

For this thesis, we are not interested in how the protocol continues after step 7.

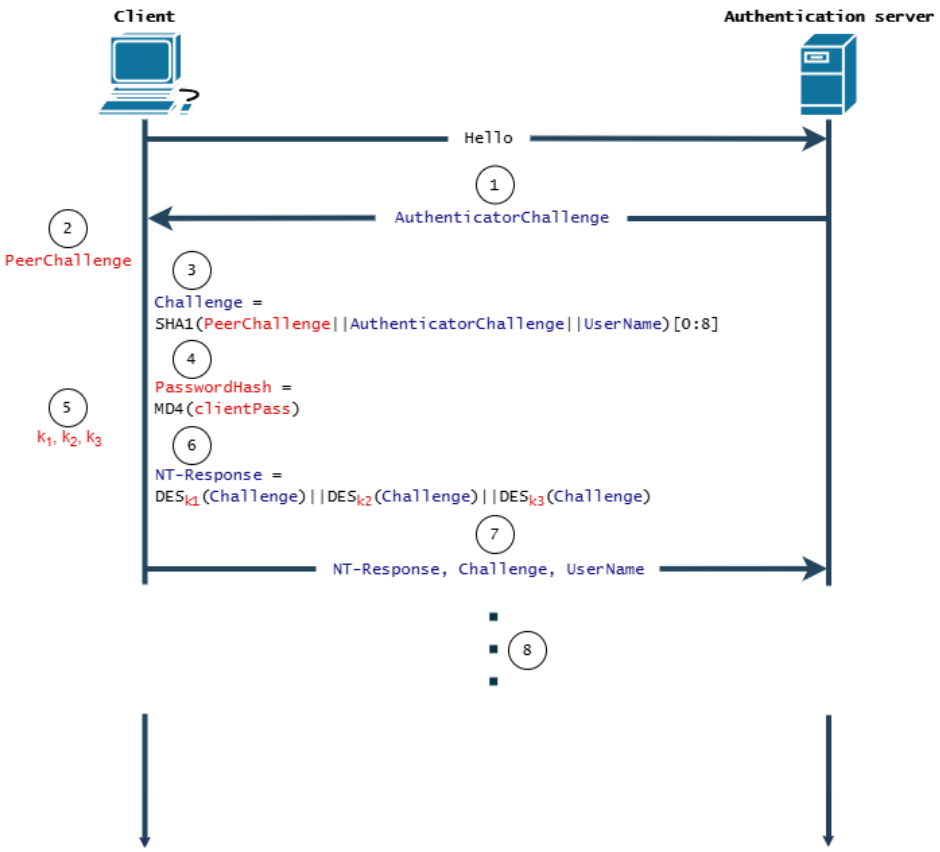

Figure 3.2: MSCHAPv2 protocol flow

3.4.3 Example of MSCHAPv2 flow

Below is an example of the MSCHAPv2 protocol and the corresponding (example) values of the aforementioned variables. In this example, UserName and clientPass are `s1234567` and `password1234`, respectively.

Step 1: Authenticator generates a challenge
Function: `generateChallenge()`
Output (AuthenticatorChallenge):
`5a:d7:04:57:70:ff:1c:9f:d2:13:49:93:b4:f8:8b:d7`

Step 2: Client generates a challenge
Function: `generateChallenge()`
Output (PeerChallenge):
`dd:2a:3f:6e:1d:d2:b4:b1:c9:af:e4:b7:a5:42:ed:7b`

Step 3: Client calculates challenge hash and truncates
Function: `SHA1()[0:8]`

Input: `PeerChallenge||AuthenticatorChallenge||UserName`
Output (Challenge):
`78:90:6d:34:95:64:23:03`

Step 4: Client hashes password and applies padding to generate NTHash
Function: `MD4()`, `pad()`
Input: `clientPass` (UTF-16le)
Output (PasswordHash):
`d4:a1:be:17:76:ad:10:df:10:38:12:b1:a9:23:cd:e4:00:00:00:00:00`

Step 5: Client calculates DES keys
Input: `PasswordHash`
Output (k_1, k_2, k_3):
`d5:51:6e:c2:76:b5:b5:20, df:89:0e:02:2a:8c:a4:46, cd:f2:01:01:01:01:01:01`

Step 6: Client calculates challenge response
Function: $\mathbf{DES}_{k_1}()$, $\mathbf{DES}_{k_2}()$, $\mathbf{DES}_{k_3}()$
Input: k_1 , k_2 , k_3
Output (NT-Reponse):
`82:07:f8:96:0d:1e:8f:8a:14:17:f5:aa:14:c7:49:48:f1:25:3e:cc:51:f7:1e:0b`

Step 7: Client sends over NT-Response, Challenge and UserName

3.5 Data Encryption Standard (DES)

As previously discussed, MSCHAPv2 relies on some legacy cryptographic mechanisms, most notably the Data Encryption Standard (DES). DES is a symmetric key encryption algorithm developed in 1975 and standardized in 1977. While at the time is was a staple of cryptographic systems, its 56 bit key size is now insecure due to the feasibility of brute force attacks [14]. Nevertheless, DES remains to be used in protocols like MSCHAPv2 which in turn is used in Eduroam, largely due to its ease of use and compatibility with older devices.

3.5.1 Encryption and Decryption

DES is a block cipher that has a Feistel structure containing Feistel functions F (Figure 3.4) which over the course of 16 repeating operations, called rounds, turns a plaintext into a ciphertext during encryption (or the other way around during decryption), see Figure 3.3.

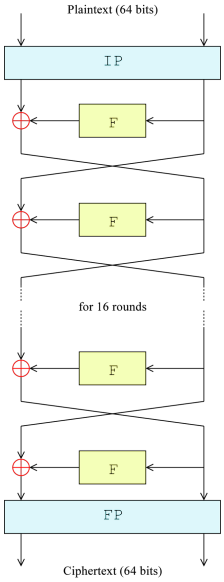

Figure 3.3: DES Feistel structure [21]

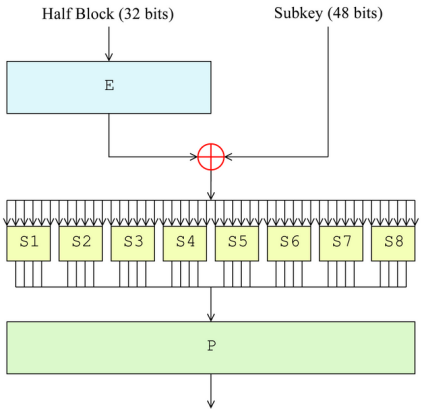

Figure 3.4: DES Feistel function[21]

3.5.2 DES keys

DES requires a 64 bit key to operate, but only 56 bits of this key are used. This is because for each of the eight bytes, the least significant bit is a parity bit which is stripped off during the actual encryption or decryption phase. The parity bits are calculated in the following way:

1. Split the input key of 56 bits into eight, 7 bit strings.
2. Take the first 7 bit string and count the number of 1's in this string.
3. If the number of 1's in the string is even, add a 1 as the parity bit. If the number of 1's in the string is odd, add a 0 as the parity bit.
4. Repeat steps 2 and 3 for the other 7 bit strings.

3.5.3 Security issues

DES has a small key size of 56 bits which is the primary reason for it becoming out of date. A modern computer can run **hashcat**'s [12] DES key recovery mode, which can find a DES key in 15 days on average (measured in 2016 [11]) on a single Nvidia GTX 1080Ti (the whole rig uses 8 GPUs, but this result considers only 1 GPU, which costs around US$1000). A rainbow table has also been constructed for the plaintext *1122334455667788*, which allows one to recover the DES key in under 25 seconds [4]. In practice, this rainbow table is not very useful, as the chance of the plaintext being

1122334455667788 (assuming that the plaintext is randomly generated and perfectly random) is 1 in 2^{56}.

Chapter 4

Attacking Eduroam

With the knowledge of how Eduroam works and where it may fail, it is now possible to carry out actual attacks on a user of Eduroam. We have seen that MSCHAPv2 may be used as the inner authentication protocol. We will see in section 4.1 that this is a problem, as MSCHAPv2 has some security issues which are normally minimal since this protocol is wrapped in a TLS tunnel. However, when a user accepts a certificate without properly validating it, this TLS tunnel may be bypassed resulting in MSCHAPv2 being isolated. We will see this in section 4.2. In section 4.3 and section 4.4, we will look at some possible solutions which are already available but not widely adopted.

4.1 Problems with MSCHAPv2

MSCHAPv2 is a relatively old protocol and its age is showing. We will see that it is vulnerable to dictionary attacks if an attacker wants to recover passwords, or the goal of an attacker might be to authenticate as another user. A way to achieve this is to recover the PasswordHash.

4.1.1 DES in MSCHAPv2

When calculating the NT-Response in MSCHAPv2, DES is used three times, each time with a different key which are derived from the same hash, namely from the PasswordHash. Looking at the PasswordHash, we notice that the last 5 bytes are always zero bytes, due to the padding that was applied to make sure that PasswordHash was 21 bytes long. This results in k_3 always having the following form:

$k_3 =$ `XX:XX:XX:01:01:01:01:01 (with parity bits, see section 3.5.2)`

$$k_3 = \texttt{XX:XX:00:00:00:00:00 (without parity bits)}$$

In other words, k_3 is effectively only 16 bits or 2 bytes long, meaning that breaking DES, and subsequently figuring out the last 7 bytes of the PasswordHash, will only take a maximum of 2^{16} key searches.

Key k_1 and k_2 are both full length DES keys (56 bits) and will require (in the worst case) 2^{56} searches to recover per key, resulting in $2^{56} + 2^{56} = 2^{57}$ searches. The amount of searches are added because the results of both DES computations are independent of each other (as a dependent example, one could use the output of the first DES computation as input to the second DES computation which means the amount of searches have to multiplied). This brings the total amount of searches needed to recover the PasswordHash to $2^{57} + 2^{16} \approx 2^{57}$. One might notice however, that all three DES computations take the same input, namely the Challenge. This results in the possibility to recover all three keys in one search of the whole keyspace, namely by computing all three DES results at the same time per key. Since the keyspace consists of 2^{56} keys, the maximum amount of key searches to recover the PasswordHash will be 2^{56}. As mentioned before in section 3.5.3, recovering DES keys in a matter of weeks is feasible using consumer hardware. We will also see this in section 4.2.1.

At this point, the attacker has the PasswordHash, and looking at the protocol we can see that this is all that is needed to authenticate as someone else. This is problematic for institutions as now they can not be sure if only students and staff are connected to their network. The next step might be to actually recover the password by breaking the MD4 hash.

4.1.2 Security of MD4 in MSCHAPv2

To recover the password of a user, an attacker must break the MD4 encryption used in step 4 of MSCHAPv2, see fig. 3.2. MD4 is a hash function that has a digest of 16 bytes or 128 bits. MD4 has been broken since 1995[5] by generating collisions in the hash function. For our research, this is not useful as we are interested in finding the actual password and not another value that hashes to the same password.

A big problem in the implementation of MSCHAPv2 is that passwords are not salted before they are hashed with MD4. This opens up the door for brute force attacks using rainbow tables since a password will always hash to the same value. A brute force attack in this context would refer to guessing the password as opposed to guessing the key as mentioned in section 4.1.1. Rainbow tables require a huge amount of storage and time to compute, which is the reason as to why there are only rainbow tables for passwords of 8 or 9 characters [1]. Table 4.1 shows a list of institutes and their required minimum password length. We can see that most researched institutes require a password above 8 or 9 characters, except for the HAN University. This poses a security risk for their students and staff, not only for their institute accounts, but maybe also for their personal accounts. A 2019 survey

by Google [10] shows that about two-thirds of the surveyed users reuse a password, whether that is for all their accounts or just a few. If users of Eduroam reuse their personal passwords for their institute accounts, their personal accounts might be vulnerable as well.

If an institute requires a user to have a password that is longer than 9 characters, then there still might be a way for an attacker to recover passwords reliably, just not targeted to a specific user. By attacking users over a longer period of time, a list containing password hashes can be created. Using this list, the attacker can now brute force random passwords until they find a match with a hash that is in the list, meaning that a password has been found. The time it takes to execute this brute force attack depends on the amount of hashes recovered, where more hashes mean that the attack speeds up. If the goal of the attacker is to simply recover a nonspecific password, then this attack is viable if we assume that not all users have very secure passwords.

Institute	**Minimum password length**
University of Twente	14
University of Amsterdam	12
Universiteit Leiden	15
Universiteit Utrecht	10
Universiteit van Tilburg	8 or 12*
Radboud Universiteit Nijmegen	10
HAN University of Applied Sciences	8**
Wageningen Universiteit	12

Table 4.1: Minimum password lengths. See also appendix B.

4.2 Staging an attack on Eduroam

In this section we will describe a practical attack using an **evil twin**[9] Eduroam network, which uses MSCHAPv2 as the inner authentication protocol. The objective is to recover an NTHash/PasswordHash or password of a user, where the latter is more severe. To mimic an access point, we use **hostapd-wpe** [15] to set up a rogue network, also known as an evil

*There are multiple web pages with conflicting information. The page where you change your password specifies an 8 character minimum. See table B.1

**This result was the most recent result we could find, from a FAQ page: `https://www.han.nl/over-de-han/datalek/`

twin. Hostapd-wpe is a tool which can be used to set up fake networks, whether that is for research or actual attacks. It supports a wide variety of customization to, for example, the protocols which are used, but also has the option to send fake server certificates when TLS tunnels are involved. When hostapd-wpe is ran, it sets up an access point which when connected to will capture any traffic that goes through the network. In the case of this thesis, we used a laptop to run hostapd-wpe which will act as our access point and our authentication server. We adjust the configuration files so that we generate a certificate that looks similar to an actual certificate used by Radboud's eduroam network, see fig. 4.1.

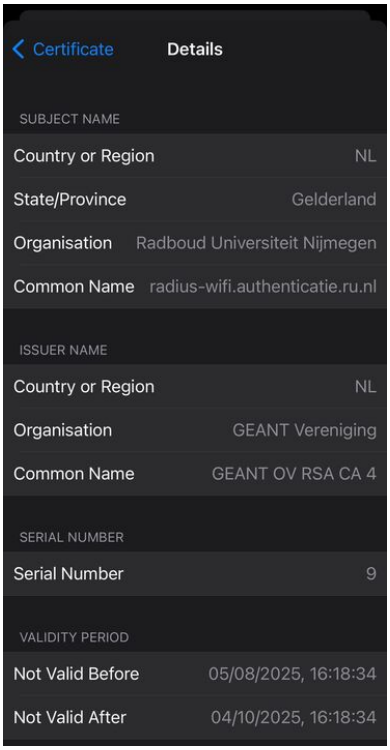

Figure 4.1: Fake Eduroam certificate

We connect to the rogue network using an iPhone 13 mini with username "s1106321@ru.nl" and an easy password "secret123" and accept the certificate. Since we accept the certificate, a TLS tunnel is set up (outer authentication) and MSCHAPv2 (inner authentication) is initiated. Hostapd-wpe now captures the NT-Response, Challenge and UserName of the user (see fig. 3.2) and the client only disconnects after step 8 of MSCHAPv2 is not completed by our laptop, as we do not know the password of the user.

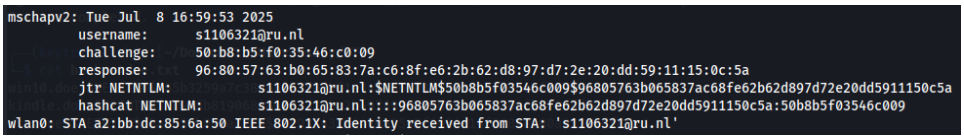

Figure 4.2: Hostapd-wpe response capture

When the NT-Response is recovered, we can start brute-forcing to find the password. Tools like *John the Ripper* or *Hashcat* can be used to brute force a password using dictionaries (using the *jtr* **NETNTLM**, also known as the NTHash, and *hashcat* NETNTLM given by hostapd-wpe, respectively). For this attack, we will be using *John the Ripper* and a common dictionary called "rockyou.txt" which contains the most common passwords. This dictionary comes preinstalled with certain Linux distributions like Kali Linux.
We add the recovered jtr NETNTLM hash to a text file, run *John the Ripper* and see the following:

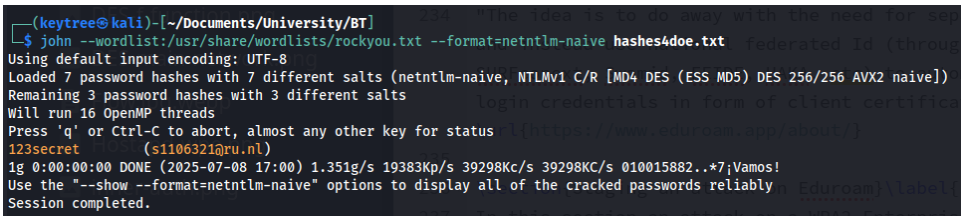

Figure 4.3: JTR command and output

Insecure passwords like the one used in this example can be quickly recovered if dictionaries are used. This is a huge problem as in Radboud's case, these credentials are also used by professors to access Brightspace and Osiris, where they manage courses and course grades. However, if a brute-force attack using dictionaries fails, there is the option to find the NTHash using DES crackers. An example of this is crack.sh which promises to crack a (random) key in a couple of days for about 17 euros. At the time of writing, crack.sh is temporarily unavailable due to proclaimed "maintenance". As explained in section 4.1.1, due to some design choices, the maximum number of key searches needed is 2^{56}. Crack.sh uses multiple FPGA's (appendix A.2) which are designed to crack DES keys and can search this 2^{56} keyspace in about 26 hours, however most people do not have FPGA's available to them.

4.2.1 Reliably cracking DES keys using consumer hardware

Cracking DES keys is not limited to organizations that have enough money to buy dedicated hardware. You are able to recover keys and passwords with consumer-grade GPUs, although much slower than using specialized

hardware. *Hashcat* has a specific mode to crack DES keys, however this would be less efficient than writing your own implementation to crack all three keys in a single search as you would still have to crack each key individually. Hashcat also comes with a benchmark feature which allows you to see how long it would take to crack a certain hash/ encrypted string. A test on an NVIDIA RTX 3080 GPU which was released in 2020 costing on average €1000 reveals that hashcat can crack 51000 million DES outputs per second (5.1×10^{10} hashes/s), see fig. 4.4. At this rate, checking all 2^{56} keys will take less than 16 days, and 8 days on average (checking 2^{55} keys).

```
OpenCL API (OpenCL 3.0 CUDA 12.8.97) - Platform #1 [NVIDIA Corporation]
============================================================================
* Device #1: NVIDIA GeForce RTX 3080, 10112/10239 MB (2559 MB allocatable), 68MCU

Benchmark relevant options:
===========================
* --optimized-kernel-enable

-------------------------------------------------
* Hash-Mode 14000 (DES (PT = $salt, key = $pass))
-------------------------------------------------

Speed.#1.........: 50917.8 MH/s (22.19ms) @ Accel:256 Loops:1024 Thr:64 Vec:1
```

Figure 4.4: Hashcat DES cracking benchmark

4.2.2 Feasibility of cracking MSCHAPv2 using consumer hardware

Hashcat also has the option to crack MSCHAPv2 as a whole using the NetNTLM hashes that it produces. Another benchmark on the same GPU shows that it can check 55000 million NetNTLMv1 hashes per second. The cracking is done by brute forcing the passwords. Let's assume that a password is 8 characters long, and can contain uppercase and lowercase letters (52), numbers (10) and special characters (32 counted from Radboud's webpage). This results in 94 characters total, meaning that there are 94^8 possible passwords. This results in recovering the password in about a day. However, most institutes do not allow passwords of length 8, see table 4.1. Following Radboud's minimum character count, let us assume that the password is of length 10, then there are 94^{10} possible passwords, which results in a password being found in 11334 days or about 31 years. An estimated cracking time of 31 years on consumer hardware is sufficiently long to consider the protocol secure against such brute-force attacks.

We have now seen multiple ways that an attacker might go about attacking an Eduroam Wi-Fi network, more specifically the MSCHAPv2 protocol. All these possibilities are dependent on the fact that a user may accidentally accept a fake certificate, which could result in them connecting to a rogue network. Recent developments have eliminated the need for users to

manually verify certificates, thereby significantly reducing the risk of them connecting to rogue access points, such as evil twins.

4.3 CAT, geteduroam and SecureW2

To help with connecting to an eduroam network, tools have been created to make the process easier and more secure. One of these tools is the **Configuration Assistant Tool (CAT)** [3], which allows institutes to set up configurations for their specific networks so that its users would not have to bother with setting up this configuration themselves. When CAT is downloaded, a user is able to download their institute's Eduroam configuration which also includes server certificates or intermediate certificates. CAT is not very popular anymore and has been largely replaced by **geteduroam**.

To elaborate, geteduroam is a project that has the main goal of making authentication to Eduroam networks more uniform and secure. They make a distinction in this goal between client and server software, as described on their website [8]:

1. Create apps to allow user to correctly set up Eduroam on their device
2. Create server software to issue client certificates to be used by these apps.

The rationale behind these goals is that the geteduroam project wants to push users and institutions to use EAP-TLS instead of EAP-TTLS/PAP or EAP-PEAP. "The idea is to do away with the need for separate eduroam user accounts, and instead use national federated Id (through eduGAIN such as SURFconext, Swamid, FEIDE, HAKA, etc) to automatically generate eduroam login credentials in form of client certificates as needed." [8] EAP-TTLS/PAP and EAP-PEAP are both protocols that are based on inner and outer authentication using different protocols, whereas EAP-TLS does authentication through both a server certificate and a client certificate. This removes the need for a user to send their credentials over a network which could open them up to a MITM attack, as described in section 4.2.

The geteduroam project has also developed an app with the same name. The geteduroam app allows users to easily connect to a nearby Eduroam network. A user must choose their home institute in the app, and is then prompted to either fill in their credentials which are then stored on their device, or they are sent to a website where they have to fill in their credentials if their institute has not signed up for geteduroam/CAT (since geteduroam uses CAT as a backend), where the authentication is then handled by eduGAIN [6]. This signing up for geteduroam/CAT can be done by institutes if they want to give up a specific setting configuration that they want

their users to use. After filling in their credentials once, a user does not have to fill in their credentials again, and your operating system will always try to connect to a nearby network that has the Eduroam SSID, whether it is a real or fake network (but will fail the authentication process if the network is fake).

SecureW2 is similar to the geteduroam app but differs in the fact that it is a general-purpose (so not limited to eduroam) cloud based network authentication platform, which is not open source and is commercial.

4.3.1 Connecting to an evil twin with geteduroam

To prove that connecting to Eduroam is more secure using geteduroam, we will show what happens when your client tries to connect to a fake Eduroam network with geteduroam set up. Since geteduroam installs a configuration which contains a certificate, the authentication process will fail at the stage where the client checks the server certificate (see step 4 in section 3.1). This looks as follows:

```
wlan0: STA 92:3d:5e:7f:b0:d2 IEEE 802.11: authenticated
wlan0: STA 92:3d:5e:7f:b0:d2 IEEE 802.11: associated (aid 1)
wlan0: CTRL-EVENT-EAP-STARTED 92:3d:5e:7f:b0:d2
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25
wlan0: STA 92:3d:5e:7f:b0:d2 IEEE 802.1X: Identity received from STA: 'anonymous@ru.nl'
wlan0: STA 92:3d:5e:7f:b0:d2 IEEE 802.1X: Identity received from STA: 'anonymous@ru.nl'
wlan0: STA 92:3d:5e:7f:b0:d2 IEEE 802.1X: Identity received from STA: 'anonymous@ru.nl'
SSL: SSL3 alert: read (remote end reported an error):warning:close notify
OpenSSL: openssl_handshake - SSL_connect error:00000000:lib(0):func(0):reason(0)
wlan0: CTRL-EVENT-EAP-FAILURE 92:3d:5e:7f:b0:d2
wlan0: STA 92:3d:5e:7f:b0:d2 IEEE 802.1X: authentication failed - EAP type: 0 (unknown)
wlan0: STA 92:3d:5e:7f:b0:d2 IEEE 802.1X: Supplicant used different EAP type: 25 (PEAP)
wlan0: STA 92:3d:5e:7f:b0:d2 IEEE 802.11: disassociated
wlan0: STA 92:3d:5e:7f:b0:d2 IEEE 802.11: deauthenticated due to local deauth request
```

Figure 4.5: Hostapd-wpe output after being connected to with geteduroam

Since no actual credentials (username and password) were sent, the only thing hostapd-wpe captured was the initial request that only contains the anonymous identity (6th line in fig. 4.5).

4.4 EAP-TLS

Another way to address the leaking of NTLM hashes of Eduroam users under MSCHAPv2, is to not use MSCHAPv2 at all. The EAP-TLS authentication protocol is a substitute for the EAP-PEAP and EAP-TTLS authentication protocols used in Eduroam. It only consists of one phase in contrast to the two phases used in EAP-PEAP and EAP-TTLS (the outer and inner

authentication protocols). As seen in table 3.1, Dutch institutes have not moved to this authentication method yet. This is mostly due to the fact that EAP-TLS is based on the public key infrastructure which is difficult to manage at large, however SecureW2 which was mentioned in section 4.3 manages this. EduVPN, which is a VPN that can be used to access an institute's network remotely, does have this mutual authentication through certificates in place, albeit via openVPN.

4.4.1 How does EAP-TLS work?

EAP-TLS is based on mutual authentication between a client and a server. This is done through server and client certificates, which are checked by the opposite party. The client is provided a certificate through an automated provisioning portal such as CAT or a web portal. The authentication flow goes as follows: [18]

1. The client requests the server to start EAP-TLS, leading the server to respond with its server certificate containing (not exclusively) its public key.

2. The client validates the server certificate and sends its own client certificate over to the server, in the case that the server certificate is valid.

3. If the client certificate is valid, the server will authenticate the client by sending a RADIUS accept packet to the access point.

See also fig. 4.6 for an overview of what is described above, including the continuation of the RADIUS protocol.

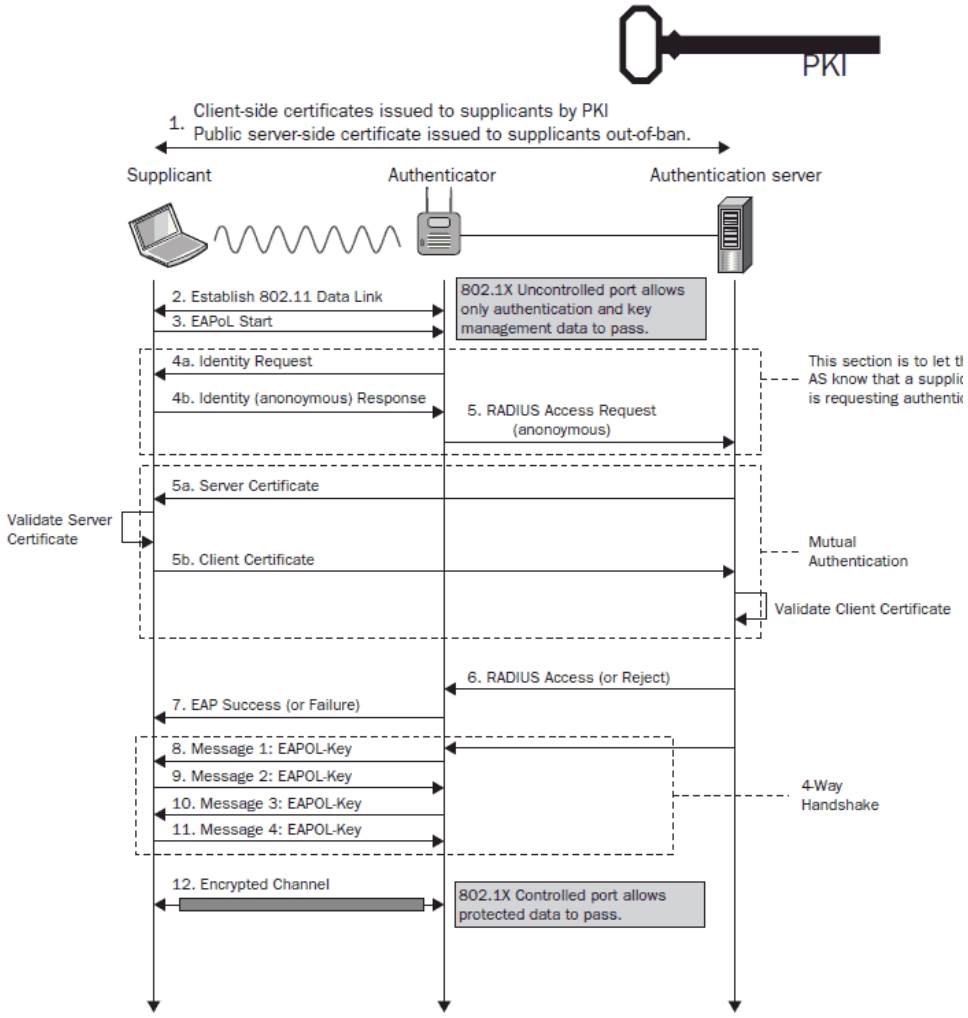

Figure 4.6: EAP-TLS overview [17]

4.4.2 Advantages and disadvantages of EAP-TLS

EAP-TLS comes with security advantages in the fact that it is passwordless authentication. As described earlier in section 4.4.1, instead of credentials (the user password) being sent through a tunnel, a certificate is sent instead. This does not remove the problem of possibly connecting to an evil twin network, but an attacker managing this evil twin will not gain knowledge of any passwords.

The thing holding institutions back from using EAP-TLS in Eduroam is the fact that some (older) devices do not support client certificates [17] which makes it difficult to ensure that everyone can access the network. It

is generally also just hard to manage a public key infrastructure and certificates, which a lot of developers do not care to spend time on if there are alternatives.

Chapter 5

Conclusions

This thesis focused on investigating the security of Eduroam's authentication method relating to accepting certificates, more specifically, which protocols play a role in authentication a user if an institute uses Eduroam and how these may be exploited if the client or user does not properly check server certificates. We have also seen that there are tools and alternative protocols which are more secure than currently used methods, but have not been adopted. At the beginning of this thesis we mentioned that the starting point of our research was the fact that a user is asked to check a certificate when trying to connect to an Eduroam network. Users will blindly accept a certificate because most people are understandably unaware of what a certificate is, and Radbouds' website also states that you should just accept the certificate. The aim of this thesis was therefore to research if these facts were exploitable in such a way that a malicious actor could recover user credentials by setting up a fake Eduroam network with fake certificates.

We have seen that Eduroam is a large collection of WPA2-Enterprise Wi-Fi networks which are all connected using the RADIUS protocol, more specifically that it is a large web of RADIUS servers which communicate with each other to authenticate a user. The RADIUS protocol is used to establish a TLS connection between a client and the user's home authentication server (outer authentication) which is then followed by the authentication server verifying a user's credentials (inner authentication). An institute has the freedom to choose which protocols are used for outer and inner authentication. For this thesis, we mostly focused on how the Radboud University has its Eduroam network set up, as they use a configuration that is used by most other institutes as well. Radboud uses MSCHAPv2 as its inner authentication protocol. This protocol is known to be weak in isolation, but continues to be used in Eduroam even though it is trivial to isolate MSCHAPv2 by acting as a man-in-the-middle to circumvent outer authentication, if certificates are not properly checked.

The weaknesses in MSCHAPv2 lie in the fact that it still uses single DES which has a keylength of 56 bits, and also has some questionable design choices such as having one DES key that consists of 5 zero bytes. Whilst recovering this subkey does not break the security that much, we still believe that it is showing cracks which should not be ignored. When pairing DES with a (TLS) tunnel, such as in Eduroam, this might not be an issue since an attacker would have to break the encryption of this tunnel first. However, if a user does not properly check the certificate and accidentally connects to a rogue network, becoming a man-in-the-middle is not that difficult and removes the worry about the tunnel entirely, leaving MSCHAPv2 completely isolated.

The attack reviewed in this thesis target MSCHAPv2 in an Eduroam network through the use of a so-called Evil Twin. Using this Evil Twin, the challenge and response hashes that are sent in MSCHAPv2 are captured, and with the use of brute-force methods like dictionary attacks, it is quite cheap to extract passwords from this captured information. This vulnerability stems from the user having to check the server certificate when it is presented to them. Tools like geteduroam have been specifically designed to help users connect to their institute's Eduroam network by configuring settings, including certificates, for them. This results in a user never having to worry about accidentally accepting a rogue certificate.

To conclude, can you trust Eduroam? Generally, yes. We have seen that in order for an attacker to retrieve passwords, there must be a set of pretty specific circumstances: the user must connect to our fake Eduroam Wi-Fi network, their password must be short and in a dictionary or insecure, and finally your institute must use an insecure protocol like MSCHAPv2. We believe that this scenario is pretty unlikely, but not impossible. We would argue that Eduroam is not flawless, specifically in terms of how institutes handle the security of their users. If, for example, the Radboud University really cared about their students and staff, they should consider the following recommendations: Institutes should strongly advise their users to connect to their Eduroam network using geteduroam, or at least list geteduroam as a security option instead of only helping with setting up a user's device. This increases the security of the authentication process and makes it nearly impossible for users to make a mistake. The security increases because it removes the potential for a man in the middle to retrieve the Challenge and Response that are sent over the network with the MSCHAPv2. We also considered if we would recommend institutes to start thinking about setting up authentication through EAP-TLS, however since it has problems with legacy devices, we do not think that this is currently feasible yet.

Chapter 6

Future Work

When a user is abroad and wants to connect to an Eduroam network that is offered at an institute, the access point there uses the RADIUS protocol to send the request all the way over to the user's home authentication server. Because these two servers are not located in the same country, or at least do not share the same top level domain (for example .nl and .de), the request is sent through the root RADIUS server. Future research could examine why the choice was made to actually send the requests through each RADIUS server instead of recursively doing a lookup similar to how DNS works. After this it might be an interesting idea to see if a denial of service attack would be feasible on this root server.

Another interesting thing to research would be to investigate password habits of students (and possibly staff) of the Radboud University. This could be done through a survey asking participants to create a password and then ask them to create a variation of that password based on some specific rules that Radboud has listed on their website. This could possibly lead to an insight on what an "average" password looks like on Radboud's campus, which could drastically speed up password recovery rates. Based on this, Artificial Intelligence or decision models could be trained to find an optimal way of checking passwords which have to adhere to these rules.

Bibliography

[1] A case for modern rainbow table usage. 2019. [Online; accessed April 2025] https://www.rainbowcrackalack.com.

[2] B. Aboba. Radius (remote authentication dial in user service) support for extensible authentication protocol (eap). RFC 7593, RFC Editor, September 2003. https://datatracker.ietf.org/doc/html/rfc3579.

[3] https://cat.eduroam.org/.

[4] https://crack.sh/.

[5] Hans Dobbertin. Cryptanalysis of MD4. Technical report, Ruhr-University Bochum, 1995. https://doi.org/10.1007/s001459900047.

[6] https://edugain.org/.

[7] https://www.eduroam.nl/.

[8] https://www.geteduroam.app/.

[9] Matthias Ghering. Evil twin vulnerabilities in wi-fi networks, 2016. https://www.cs.ru.nl/bachelors-theses/2016/Matthias_Ghering___4395727___Evil_Twin_Vulnerabilities_in_Wi-Fi_Networks.pdf.

[10] Google and Harris Poll. Online security survey. Technical report, 2019. https://services.google.com/fh/files/blogs/google_security_infographic.pdf.

[11] Jeremi M. Gosney. 8x1080ti.md. https://gist.github.com/epixoip/ace60d09981be09544fdd35005051505, 2016.

[12] https://hashcat.net/hashcat/.

[13] P. Hoffman and F. Yergeau. Utf-16, an encoding of iso 10646. RFC 2781, RFC Editor, February 2000. https://www.ietf.org/rfc/rfc2781.txt.

[14] S. Kelly. Security implications of using the data encryption standard (des). RFC 4772, RFC Editor, December 2006. https://datatracker.ietf.org/doc/html/rfc4772.

[15] OpenSecurityResearch. hostapd-wpe. https://github.com/OpenSecurityResearch/hostapd-wpe, 2012.

[16] http://web.archive.org/web/20250814155715/https://www.ru.nl/services/campusfaciliteiten-gebouwen/ict/wifi/wifi-eduroam-instellen/wifi-eduroam-instellen.

[17] Vivek Raj. https://www.securew2.com/blog/eap-tls-vs-eap-ttls-pap.

[18] D. Simon, B. Aboba, and R. Hurst. The eap-tls authentication protocol. RFC 5216, RFC Editor, March 2008. https://datatracker.ietf.org/doc/html/rfc5216.

[19] Sjoerd van der Kamp, Rolf van Wegberg, and Michel van Eeten. "Careful with that Roam, Edu": Experimental Analysis of Eduroam Credential Stealing Attacks. In *2022 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 400–414. IEEE, 2022. https://ieeexplore.ieee.org/document/9764586.

[20] K. Wierenga. The eduroam architecture for network roaming. RFC 7593, RFC Editor, September 2015. https://datatracker.ietf.org/doc/html/rfc7593.

[21] Wikipedia, the free encyclopedia. Data encryption standard, 2025. [Online; accessed April 2025]. https://en.wikipedia.org/wiki/Data_Encryption_Standard.

[22] G. Zorn. Microsoft PPP CHAP extensions, version 2. RFC 2759, RFC Editor, January 2000. https://datatracker.ietf.org/doc/html/rfc2759.

Appendix A

Extra information

This chapter explains some extra information about topics mentioned in the thesis. This information is not needed to understand the conclusions of the thesis but might help with clarifying some specific information that is mentioned.

A.1 Byte order Mark (BOM)

The **Zero width non-breaking space**, also known as the Byte order mark or BOM, is a special Unicode character code (0xFEFF) that has several different purposes [13]. It is a character code that may be prepended to a unicode string. The two main important usages of the BOM is to

1. Signal to the receiver of the message that unicode characters are being used.
2. Signal to the receiver of the message which byte order is being used. 0xFE followed by 0xFF signifies big-endian and 0xFF followed by 0xFE signifies little-endian.

Depending on the programming language and implementation, the BOM might be automatically prepended to a UTF-16 message. By specifying if the message is in big endian (UTF-16BE) or little endian (UTF-16LE), the BOM will not be prepended.

A.2 Field Programmable Gate Arrays

Field Programmable Gate Arrays (FPGAs) are specialized pieces of hardware that are commonly used in research for specific tasks. An FPGA consists of multiple logic blocks which are interconnected and can be programmed on the fly (in the field) to do a sub-task of the final task. Because

of this specificity, FPGAs are much faster at completing tasks than non-specific CPUs and GPUs and therefore preferred if you are working with large amounts of data (such as the amount of possible DES keys). FPGAs are relatively expensive, as a high end FPGA can start at around US$4000.

Appendix B

Links used

University	Password Change Link
University of Twente	https://www.utwente.nl/en/service-portal/workplace-support/accounts-passwords/ut-account-incl.-activation-password-change#password-change-and-reset
University of Amsterdam (UvA)	https://student.uva.nl/en/information/use-a-different-strong-password-for-each-of-your-ac
Leiden University	https://www.staff.universiteitleiden.nl/ict/ulcn-account/forgotten-your-password
Utrecht University	https://manuals.uu.nl/manual/wachtwoord-veranderen-vergeten/
Tilburg University (Info Page)	https://www.tilburguniversity.edu/nl/over/gedrag-integriteit/privacy-en-security/informatiebeveiligingsbeleid/wachtwoord
Tilburg University (Set Password)	https://setpassword.uvt.nl/gatekeeper/changepassword?language=nl
Radboud Universiteit	https://www.ru.nl/en/services/campus-facilities-buildings/ict/password/password-safety-guidelines
HAN University	https://www.han.nl/over-de-han/datalek/
Wageningen University	https://password.wur.nl/Start.aspx

Table B.1: Table containing links used for finding minimum password lengths

Index