KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Problem Solving and Engineering Design part 3

ESAT2A2

Jarle Braeken (r0998124)
Arthur Cukier (r0976603)
Pieter Deferme (r0995734)
Robin Derikx (r0978208)
Willem Hendig (r0995944)
Giel Swenters (r1006315)

Crack the Wi-Fi

PRELIMINARY REPORT

Co-titular
Vincent Rijmen

Coach(es)
John Gaspoz
Dilara Toprakhisar

ACADEMIC YEAR 2024-2025

Declaration of originality

We hereby declare that this submitted draft is entirely our own, subject to feedback and support given us by the didactic team, and subject to lawful cooperation which was agreed with the same didactic team.
Regarding this draft, we also declare that:

1. *Note has been taken of the text on academic integrity (https://eng.kuleuven.be/studeren/masterproef-en-papers/documenten/20161221-academischeintegriteit-okt2016.pdf).*
2. *No plagiarism has been committed as described on https://eng.kuleuven.be/studeren/masterproef-en-papers/plagiaat.*
3. *All experiments, tests, measurements, …, have been performed as described in this draft, and no data or measurement results have been manipulated.*
4. *All sources employed in this draft – including internet sources – have been correctly referenced.*

Contents

0 Introduction

This section provides a simplified overview of the project, "Crack the Wi-Fi".

This project demonstrates how a single vulnerability in a network can compromise the security of the whole Local Area Network (LAN). The network configuration, illustrated in Figure 3, consists of two interconnected routers: one secured with the weak WEP protocol and another with the stronger WPA2 security. These routers are connected via Ethernet, with a Raspberry Pi 3 hosting a local website on the WPA2-secured network. To demonstrate the vulnerability of the WEP-protected network, the connection between a victim and the first router is abused to intercept and modify data transmitted to the secure website through Address Resolution Protocol (ARP) manipulation, *see Chapter 4*. Before the attack, exploiting the LAN is crucial to gather necessary information, like the physical- or MAC address, *see Chapter 4*. While network packets are broadcast through the air and encrypted for security, they can be intercepted without establishing a direct network connection. The encryption key, normally restricted to authorized network devices, can be obtained using the Aircrack-ng suite. The remainder of this paper is organized as follows:

- Chapter 1 details how the Aircrack-ng network security suite [Devine, 2022] is used to obtain the WEP-encryption key.
- 2 describes how sent packages are intercepted and decrypted, with Section 2.2 explaining how the WEP protocol encrypts its data using an RC4 algorithm.
- Chapter 3 describes the implementation of the local website
- Chapter 4 explains the concept, process and implementation of a man-in-the-middle (MitM) attack.
- Chapter 5 contains a brief summary of the project as well as outlining further improvements to the project
- Chapter 6 details how knowledge obtained through past courses is integrated in this project

1 Aircrack

The first step in the project is acquiring the WEP key, as it's a key component for all the following steps. Aircrack-ng is used to find this key.

1.1 Overview of Aircrack

Aircrack-ng, further referred to as Aircrack, is a suite of network security tools used for monitoring, attacking, and cracking WEP and WPA encrypted networks [Devine, 2022]. It's commonly used to assess network security. In this project, Aircrack will crack a WEP-encrypted network to retrieve the WEP key.

1.2 Cracking the WEP access point

WEP encryption uses a combination of a shared key and an Initialization Vector (IV) to encrypt network traffic. However, due to the weak IV implementation, it is possible to recover the shared key after capturing enough packets. Aircrack-ng exploits this vulnerability by analyzing captured data and IVs to reconstruct the WEP key.

1.2.1 Structure of a WEP-encrypted packet

The inherent vulnerability of WEP encryption lies in its packet structure. To encrypt a packet, a key is used, which consists of:

- A 24-bit Initialization Vector (IV)
- A WEP key consisting of either 40 bits (for 64-bit encryption) or 104 bits (for 128-bit encryption)

This key is used in an RC4 algorithm to encrypt the packet. However, as the IV is randomly generated for every packet, this causes a problem for the receiver. To enable decryption, the IV is appended to the front of the encrypted packet. This is the core weakness of WEP encryption, as it creates two critical vulnerabilities:

1. The IV's limited size (24 bits) leads to the inevitable reuse
2. The transmission of the IV enables attackers to collect and analyze IV patterns

When two packets share the same IV, also known as IV Collision, cryptanalysis can be used to determine the static WEP key, allowing for all subsequent packets to be decrypted.

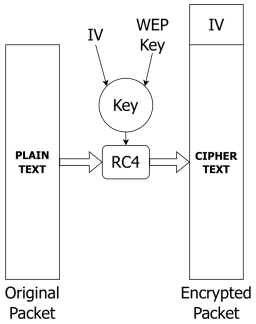

Figure 1: WEP-encrypted packet

1.3 Aircrack commands

Aircrack-ng consists of several tools, each performing a specific function in the process of cracking WEP encryption. The key tools used in this project are explained in the following paragraphs. The specific commands used are detailed in Appendix 7.1.

1.3.1 airmon-ng

Typically, a Raspberry Pi can only receive packets addressed to itself. This necessitates the use of the "airmon-ng" command. This command enables monitor mode on the Raspberry Pi, allowing it to listen to all packets in the air, thus enabling it to intercept WEP network traffic.

1.3.2 airodump-ng

"airodump-ng" captures the data that the Raspberry Pi detects. By specifying a channel and BSSID, it can be configured to exclusively capture data from a specific network. This is how data from the WEP network is gathered.

1.3.3 aircrack-ng

This command makes use of the weakness of WEP encryption. Once a sufficient amount of packets have been captured by "airodump-ng", "aircrack-ng" is employed to crack the WEP key. Using packets with a shared IV, Aircrack-ng utilizes cryptanalysis to determine the WEP key. A more detailed explanation of the RC4 algorithm can be found in Section 2.2.

Figure 2: A WEP key that has been found by Aircrack

2 Sniffer

To capture and read network traffic on the router, a network capture system, known as a sniffer, is implemented. This system is developed using Scapy which is " a Python program that enables the user to send, sniff, dissect and forge network packets. This capability allows construction of tools that can probe, scan or attack networks." [Biondi, 2024] In this implementation, the system intercepts and analyzes traffic from the WEP-encrypted network. By combining Scapy with a decryption algorithm that uses the WEP key obtained with Aircrack, the sniffer captures, decodes and reads network traffic. Notably, the packet capture operates externally to the target network, as illustrated in Figure 3.

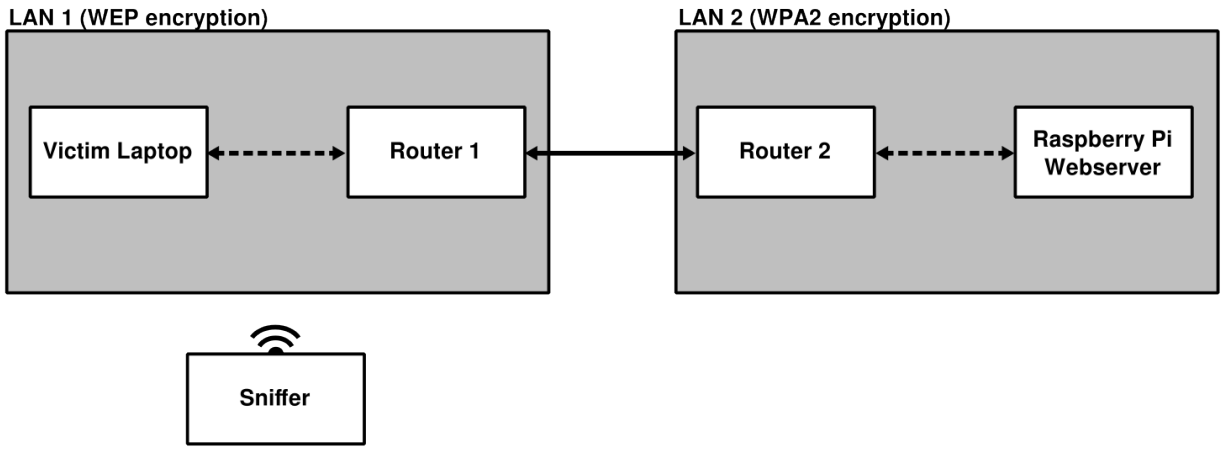

Figure 3: Network Layout

2.1 Sniffing

The system utilizes Scapy's "sniff" functionality to intercept network traffic directed at the Raspberry Pi. When using monitor mode, it can detect and sniff all wireless traffic within range. This can be filtered to only include packets sent from a specific MAC address.

2.2 WEP protocol

WEP stands for Wired Equivalent Privacy, which also describes it's function. This protocol tries to protect your private data in a wireless connection with the same security as a wired connection. In 1999, it was the standard for Wi-Fi security. It makes use of the RC4 encryption algorithm to encrypt the data. On this day WEP isn't used anymore because flaws were detected and abused. In the configured network used for this report 3, router 1 uses the WEP protocol. This router will be used as a weak point in the network.

The following paragraphs describe how the RC4 algorithm works and how to decrypt its packages using the key gathered by the Aircrack attack [chapter 1].

2.2.1 RC4 encryption

The following explanation on the RC4 encryption is based upon the article written by S Sriadhi, Robbi Rahim and Ansari Saleh Ahmar [S Sriadhi, 2018]. This explanation is a simplified description, but provides enough insight to understand the use of it in our decryption. 7.3 gives a pseudocode written based upon the process in this paragraph.

The package before encryption consists of a plain, readable text. To encrypt this package, a secret key is necessary. This key could be any string of text with a length between 1 and 256 characters. In most cases this key is less than 256 characters long. To obtain a string with the desired length of 256, the shorter key is simply repeated.

This 256 character long string is now used to generate a keystream. Using the ASCII-table [ASCII-table, 1963], the extended key is converted to a 3 digits long code. This is done for the entire extended key. At the start, the keystream is a simple table with 256 entries. The first entry holds 0, the one after holds 1. Repeating this results in the last entry holding 255. Using a simple repetitive algorithm, this keystream table will be permutated using the extended key. Note the following two variables:

- i holds the amount of iterations the algorithm has been done. It starts at 0 and ends at 255.
- j holds an integer between 0 and 255 which is calculated by formula 1. Using the modulus operator, j will never exceed these boundaries. For the first iteration, j holds a value 0.

These variables are also used as indices for the key or the keystream. In this case they are noted key(stream)$[i]$.

$$j = (j + \text{keystream}[j] + \text{key}[i]) \bmod 256 \tag{1}$$

Note that the j on the right side of the equation still holds the value of the previous iteration.

To finish the algorithm, 2 entries in the keystream are swapped. These entries are defined by the calculated indices i and j.

$$\text{keystream}[i], \text{keystream}[j] = \text{keystream}[j], \text{keystream}[i]$$

Note that the swapped values can be any value between 0 and 256 and are only indicated by the indices i and j.

After completing the 256 iterations. The final keystream is generated. Using this keystream, the original plain text can now be encrypted. This process starts with converting the ASCII codes of each character to an 8 digit long binary code. This is done for the keystream and for the plain text. Next, the logical operator XOR is used on both binary strings to generate the encrypted output string, which also is an 8 digits long string for each character.

The visualization in table 1 gives an example of the process described in the paragraph above. The keystream used in this example is completely random.

INPUT			**KEYSTREAM**			**OUTPUT**
index	*character*	*binary*	*index*	*character*	*binary*	*binary*
0	P	01010000	0	¥	10100101	11110101
1		10100000	1	D	01000100	11100100
2	e	01100101	2	c	00101110	01001011
3	n	01101110	3	L	01001100	00100010
4		10100000	4	š	10011010	00111010
5	O	01001111	5	ö	11110110	10111001

Table 1: Example of XOR-encryption using a random keystream

One of the benefits of using the XOR operator is how easy it is to reverse. To reverse the encryption, use the XOR operator on the encrypted input (or the output in 1) and the keystream.

The final step of the encryption is to turn the final binary output back into symbols using the ASCII-table [ASCII-table, 1963].

2.2.2 RC4 decryption

As mentioned in the introduction 2.2, the way the intercepted packages are decrypted will be the exact same way as the victim normally would decrypt them. This is possible because the secret key is known as a result of the successful Aircrack attack 1.

First, the key is used to generate the same keystream as described in the encryption process. Then, the encrypted message and the keystream are both converted to binary. When converted, the XOR operator is used on both binary streams to generate the decrypted binary stream. This illustrates how easily reversible the XOR operator is, as explained above. The last step is to convert the binary stream back to the decrypted message.

3 Website

For the website, it was decided to make a clone of Netflix. The idea is to trap the victim by making them believe they are consulting the real Netflix website. The user would then enter their personal information either by logging in to their existing Netflix account or by creating a new one. In the first scenario, it would be possible to steal their password and e-mail/phone number. In the second, the financial information of the victim such as cardholder name, card number, expiry date and CVC code would be at risk.

3.1 Website structure

The website consists of six distinct pages:

- The "index" page, where the user begins their journey. This page of the website is shown in Figure 6, Figure 7 and Figure 8.
- The "login" page, where users can sign in to an existing account. This page of the website is shown in Figure 9.
- The "getstarted" page, allows users to create a new account. This page of the website is shown in Figure 10 and Figure 11.
- The "keuze" page, where users can select from different payment options. This page of the website is shown in Figure 12 and Figure 13.
- The "bank" page, where users finalize the purchase of their chosen Netflix plan. This page of the website is shown in Figure 14 and Figure 15.
- The "fout" page, where the user is directed after entering their information. This page notifies the user of an error and redirects them to the real Netflix website. This way, the user remains unaware that their data has been stolen. This page of the website is shown in Figure 16.

The complete code for these six pages is available in the GitHub repository. A link of the repository is provided in the appendices section.

3.2 Programming languages

The decision was made to code in HTML, CSS and JavaScript for the website. Each programming language has a clear and distinct purpose. The HTML code adds the structure and the content, the CSS code customizes the style and the JavaScript code changes the language and checks if all fields are correctly filled in.

3.3 Description of user experience

When a user visits the website, he first lands on the homepage where he can adjust the language as desired, log in and create an account. When scrolling down, sections are shown that highlight the positives of Netflix. There is a FAQ at the very bottom of the page. If the user wishes to log in, he will be taken to another page where his phone number or email and his password will be requested. Once the user is logged in, he will be taken to an error page with a link to the real Netflix website. This ensures that the user will never know that his information has been stolen.

If the user wishes to register, he will have to provide his full name, email, phone number, password and gender. He will then be taken to another page where three different subscriptions will be offered. When the subscription choice is made, the user will have to add his financial information. Finally, after clicking on "Proceed", he will arrive at an error page with once again a link to the real Netflix website.

The full code of all the files that are responsible for this website is in the appendix. Necessary explanations of the different parts of the code were added.

3.4 References

In developing the website, inspiration and information were gathered from various sources, including YouTube videos. One video [Coding With Dawid, 2022] provided valuable insights for creating a clone of the Netflix main page. Another video [CodingNepal, 2023] helped replicate the Netflix login page. In order to have a better understanding of registration and validation forms in JavaScript, a third video [JavaScript Academy, 2021] was a useful resource.

4 Man-in-the-Middle

The main target of our project is to intercept Locale Area Network (LAN) traffic packages and alternate them. This way a man-in-the-middle (MitM) attack can be performed between the traffic forwarding (forwarding to the Wide Area Network (WAN)) router and the victim's device.

To perform a MitM attack, the victim's device has to think that the attacker's device is the traffic forwarding router and vice versa the router has to think the attacker's device is the victim's device.

This manipulation is done by fooling the Address Resolution Protocol (ARP). ARP is a commonly used protocol pertaining to computer communications. In a LAN ARP messages are shared. With these messages, the router in the network knows which IP addresses belong to which computer in the network, through correlating IP addresses and corresponding physical addresses (MAC addresses). The boundary that the messages are sent within the LAN, obligates the attacker to connect to the LAN for carrying out an ARP Poisoning or Spoofing attack. [Aayush Majumdar, 2021]

For this project, the MitM will be demonstrated by changing the redirection links. This way, a user who clicks on "sign in" can be redirected to a web page of the attacker's choice.

4.1 ARP poisoning

An ARP Poisoning attack is an attack in which spoofed ARP messages are sent to the default gateway on the locale network (LAN) with the intent of changing the ARP cache table. [Aayush Majumdar, 2021] The default gateway on a network is a node/gate with the purpose of connecting and so gaining access to another network. In the ARP cache table is information of the relation IP address $\longleftrightarrow$ MAC address stored. [Wikipedia, 2024]

Once the default gateway receives these ARP packets, the changes are broadcast to all devices connected to the network. This way other devices can locate other devices, from whom it has to request something. Now all devices recognize the attacker's device as the victim's device. So all requests are first sent to the attacker. Depending on the kind of attack to use, the attacker has to forward the network packets to the victim and in reverse if the packets came from the victim. In this process, data containing the packets can be changed. This process of intercepting and alternating the data packets is the man-in-the-middle (MitM) attack. [Aayush Majumdar, 2021]

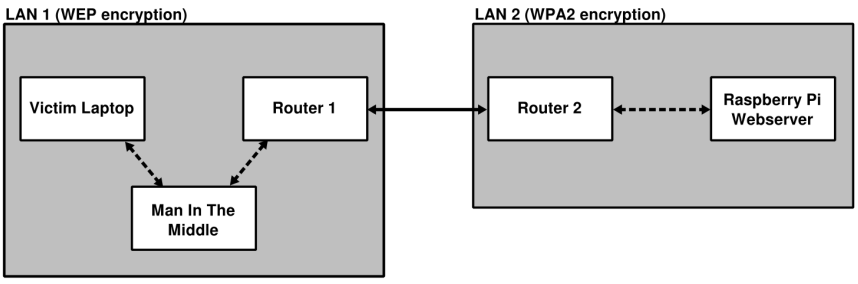

Figure 4: Network scheme

4.2 Scapy

For ARP-spoofing and the MitM attack, Scapy is used in Python for setting up an ARP poisoning packet, sending it to the gateway and afterward restoring the network by reversing the ARP spoofing. The physical address (MAC address), the gateway and the victim's IP needed for this program are found by Aircrack.

To control if the ARP cache table has changed, the ARP table can be printed using the command: >>> **ARP -a** or using the package 'sniff' from Scapy in Python.

4.3 Live intervention

Once the ARP poisoning attack succeeds, the attacker can implement its goal. This project's goal of the MitM attack is to implement redirect links. After a certain time and amount of user interactions, the next interaction causes the redirect link to be activated. Another possible goal locks the user in an unwanted screen view.

5 Conclusion and further improvements

To conclude this preliminary report, this paragraph will go over the parts of the project which have already been successfully executed, and that will be made in the near future.

The Aircrack attack can gather the WEP key, as wanted. The use of this key to decrypt the RC4 encryption is also understood and complete. The current version of the website is valid for testing, though adjustments will probably be made in the future.

Getting the sniffer to work without being connected to the network seems to be the most challenging and important part of the project, as this still does not work. The second part of the project which will take some time in the future will be the modification of the data. The Man in the Middle attack has to be studied and understood in order to modify the victim's data, without the victim having this knowledge.

5.1 Gantt-chart

Included below is a Gantt chart detailing the distribution of subtasks in the project.

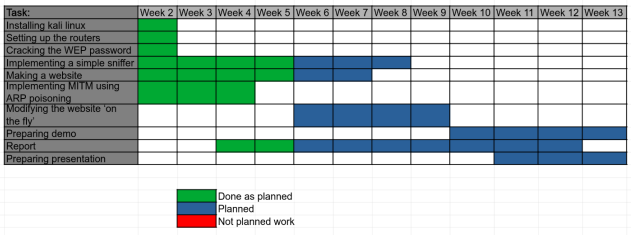

Figure 5: Gantt chart

6 Subject integration

The project is divided in multiple different interconnected domains (Software setup, Aircrack, sniffer, man-in-the-middle attack (MitM) using ARP-poisoning, website hosting). These domains use a combination of certain subjects seen in the first and second year of the Bachelor of Engineering.

The integrated subjects:

- Methodiek van de Informatica
- Toegepaste algebra
- Informatieoverdracht en -verwerking (IOV)
- Kansrekenen en statistiek

Aircrack For this domain *Kansrekenen en statistiek* are used. The more data gathered by the sniffer, the greater the chances of cracking the WEP key.

Sniffer The sniffer integrates knowledge from multiple subjects, including *Methodiek van de informatica*, *Toegepaste algebra*, and *IOV*. The sniffer is programmed in Python, applying techniques from *Methodiek van de informatica*. Capturing data packets from the network involves working with transmission protocols, which is part of the *IOV* subject area. Additionally, the decryption of captured, encrypted data relies on the RC4 algorithm, which uses algebraic techniques covered in *Toegepaste algebra*.

MitM De MitM is programmed in Python and uses techniques from the subject *Methodiek van de informatica*.

Web Hosting The website is programmed with HTML, CSS and JavaScript.

References

[Aayush Majumdar, 2021] Aayush Majumdar, Shruti Raj, T. S. (2021). Arp poisoning detection and prevention using scapy. *Journal of Physics*.

[ASCII-table, 1963] ASCII-table (1963). `https://www.ascii-code.com/`.

[Biondi, 2024] Biondi, P. (2024). Introduction — scapy 2.6.0 documentation. `https://scapy.readthedocs.io/en/latest/introduction.html`.

[Coding With Dawid, 2022] Coding With Dawid (2022). Netflix clone with html and css (tutorial for beginners). Accessed: 2024-10-01.

[CodingNepal, 2023] CodingNepal (2023). Create a netflix login page in html and css — netflix login page clone in html and css. Accessed: 2024-10-01.

[darkAudax, 2010] darkAudax (January 11, 2010). Tutorial: Simple wep crack. `https://www.aircrack-ng.org/doku.php?id=simple_wep_crack`.

[Devine, 2022] Devine, C. (May 10, 2022). Aircrank-ng. `https://www.aircrack-ng.org/`.

[Ibrahim, 2021] Ibrahim, A. (March 16, 2021). Wired equivalent privacy (wep) explained. `https://www.youtube.com/watch?v=6cKgoA3Qj60`. [Online; accessed 14-October-2024].

[JavaScript Academy, 2021] JavaScript Academy (2021). Form validation using javascript on the client side for beginners. Accessed: 2024-10-01.

[S Sriadhi, 2018] S Sriadhi, Robbi Rahim, A. S. A. (2018). Conf. ser. 1028 012057. *Journal of Physics*.

[Wikipedia, 2024] Wikipedia (2024). Default gateway — Wikipedia, the free encyclopedia. [Online; accessed 14-October-2024].

Use of Artificial Intelligence (AI)

Gebruik van een AI-schrijfassistent bij het schrijven van dit P&O3-verslag

Teamnaam: ESAT2A2

Teamleden: Jarle Braeken, Arthur Cukier, Pieter Deferme, Robin Derikx, Willem Hendig & Giel Swenters

Duid aan met "X":

O **Dit verslag maakte geen gebruik** van een AI-schrijfassistent.

X **Dit verslag maakte wel gebruik** van een AI-schrijfassistent (vb.: ChatGPT, Microsoft Copilot, Google Gemini, Grammarly, …), nl.:
ChatGPT

Duid aan met "X" (meerdere opties mogelijk) op welke manier dit gebruikt werd.

X ~~**Uitsluitend**~~ **als hulp bij de taal**

- *Gedragscode*: Dit is gelijkwaardig met het gebruik van een spellingscontrole

X **Als zoekmachine om te leren over een bepaald onderwerp**

- *Gedragscode*: Dit gebruik is vergelijkbaar met bijvoorbeeld een Google-zoekopdracht of het raadplegen van Wikipedia. Wees je ervan bewust dat de output van de AI-schrijfassistent evolueert en in de loop van de tijd kan veranderen.

O **Voor een literatuurstudie**

- *Gedragscode*: Dit gebruik is vergelijkbaar met bijvoorbeeld een Google Scholar-zoekopdracht. Houd er echter rekening mee dat sommige AI-schrijfassistenten geen of onjuiste verwijzingen kunnen geven. Als student ben je verantwoordelijk voor het verder controleren en verifiëren van de afwezigheid of juistheid van verwijzingen.

O **Voor ondersteuning bij korte stukken tekst (bv. parafraseren, samenvatten, herschrijven, …)**

- *Gedragscode*: Dit gebruik is vergelijkbaar met bijvoorbeeld Google Docs, dat wordt aangedreven door generatieve taalmodellen.

X Voor het genereren van programmeercode

- *Gedragscode*: Vermeld correct het gebruik van een AI-schrijfassistent en citeer het. Je kunt ook vragen aan het AI-schrijfassistent hoe je het moet citeren.

O **Voor het genereren van nieuwe onderzoeksideeën**

- *Gedragscode*: Controleer in dit geval verder of het idee nieuw is of niet. Het is waarschijnlijk dat het gerelateerd is aan bestaand werk, waar dan naar moet worden verwezen.

O **Voor het genereren van (grote) tekstblokken**

- *Gedragscode*: Het invoegen van tekstblokken zonder aanhalingstekens van een AI-schrijfassistent in je verslag is niet toegestaan. Volgens Artikel 84 van het onderwijs- en examenreglement moet bij het evalueren van je werk correct worden beoordeeld op je eigen kennis. Indien het echt nodig is om een tekstblok van een AI-schrijfassistent in te voegen, maak hier dan een citaat van door gebruik te maken van aanhalingstekens. Dit moet hoe dan ook tot een absoluut minimum beperkt blijven.

O **Ander gebruik**

- *Gedragscode*: Bespreek het gebruik van een AI-schrijfassistent in dit geval met je begeleiding. Motiveer hoe je voldoet aan Artikel 84 van het onderwijs- en examenreglement. Leg het gebruik en de toegevoegde waarde van een AI-schrijfassistent in deze situatie uit:

.....

.....

.....

.....

7 Appendices

7.1 Aircrack commands

These are the Aircrack commands used for cracking the WEP key.

```
airmon-ng check kill
airmon-ng start wlan0
airodump-ng wlan0mon
```

Replace CHANNEL en BSSID with the right channel and bssid.

```
airodump-ng -c CHANNEL --bssid BSSID -w dump wlan0mon
```

After enough packets are recieved the encryption key can be cracked using:

```
aircrack-ng -b BSSID dump-01.cap
```

After cracking the password the wlan0mon interface can be stopped using:

```
airmon-ng stop wlan0mon
```

And the regular interdace can be restarted using:

```
systemctl start NetworkManager.service
```

7.2 Github repository

All the code for the website, ARP spoofing, Aircrack, and MitM can be found in a GitHub repository, accessible via the following link: `https://github.com/Jarle-student/ESAT2A2`.

7.3 RC4 encryption

```
import pyperclip

def extend_key(key):
    key_lenght = len(key)
    extended_key = []
    for i in range(256):
        extended_key.append(key[i % key_lenght])
    return extended_key

def convert_to_ASCII(list):
    for i in range(len(list)):
        list[i] = ord(list[i])
    return list

def generate_keystream(key):
    keystream = [i for i in range(256)]
    j = 0
    for i in range(256):
        j = (j + keystream[j] + key[i]) % 256
        keystream[i], keystream[j] = keystream[j], keystream[i]
    return keystream

def convert_to_list(text):
    output = []
    for i in range(len(text)):
        output.append(text[i])
    return output

def convert_to_binary(list):
    for i in range(len(list)):
        list[i] = format(list[i],'08b')
    return list

def XOR(bin1, bin2):
    output_bin = ""
    for i in range(len(bin1)):
        if bin1[i] == bin2[i]:
            output_bin = output_bin + "0"
        else:
            output_bin = output_bin + "1"
    return output_bin

def adjust_keystream_length(keystream, text_list):
    while len(text_list) > len(keystream):
        keystream = keystream + keystream
    return keystream[:len(text_list)]

def crypt_bin(keystream, text):
    encrypted_bin_list = []
    for i in range(len(text)):
        encrypted_bin_list.append(XOR(keystream[i], text[i]))
    return encrypted_bin_list

def convert_to_standard(list):
    encrypted_message = ""
    for i in range(len(list)):
        char = chr(int(list[i], 2))
        encrypted_message = encrypted_message + char
    return encrypted_message

```

```
61 def encrypt():
62     plain_text = input("Insert message here: ")
63     text_list = convert_to_list(plain_text)
64     binary_text = convert_to_binary(convert_to_ASCII(text_list))
65     key = input("Insert secret key here: ")
66     extended_key = extend_key(key)
67     ASCII_key = convert_to_ASCII(extended_key)
68     keystream = generate_keystream(ASCII_key)
69     bin_keystream = convert_to_binary(keystream)
70     adj_bin_ks = adjust_keystream_length(bin_keystream, binary_text)
71     encrypted_bin = crypt_bin(adj_bin_ks, binary_text)
72     encrypted_message = convert_to_standard(encrypted_bin)
73     pyperclip.copy(encrypted_message)
74     return encrypted_message, key
75
76 def decrypt(encr_message, key):
77     messge_lst = convert_to_list(encr_message)
78     extended_key = extend_key(key)
79     ascii_key = convert_to_ASCII(extended_key)
80     keystream = generate_keystream(ascii_key)
81     bin_keystream = convert_to_binary(keystream)
82     adj_ks = adjust_keystream_length(bin_keystream, messge_lst)
83     decrypted_bin = crypt_bin(adj_ks, convert_to_binary(convert_to_ASCII(
           messge_lst)))
84     decrypted_message = convert_to_standard(decrypted_bin)
85     return decrypted_message
86
87
88 def main():
89     encrypted_message, key = encrypt()
90     print("This is the encrypted message: " + encrypted_message)
91     decrypted_message = decrypt(encrypted_message, key)
92     print("This is the decrypted message: " + decrypted_message)
93     return
94
95 main()
```

7.4 Website

7.4.1 Main page

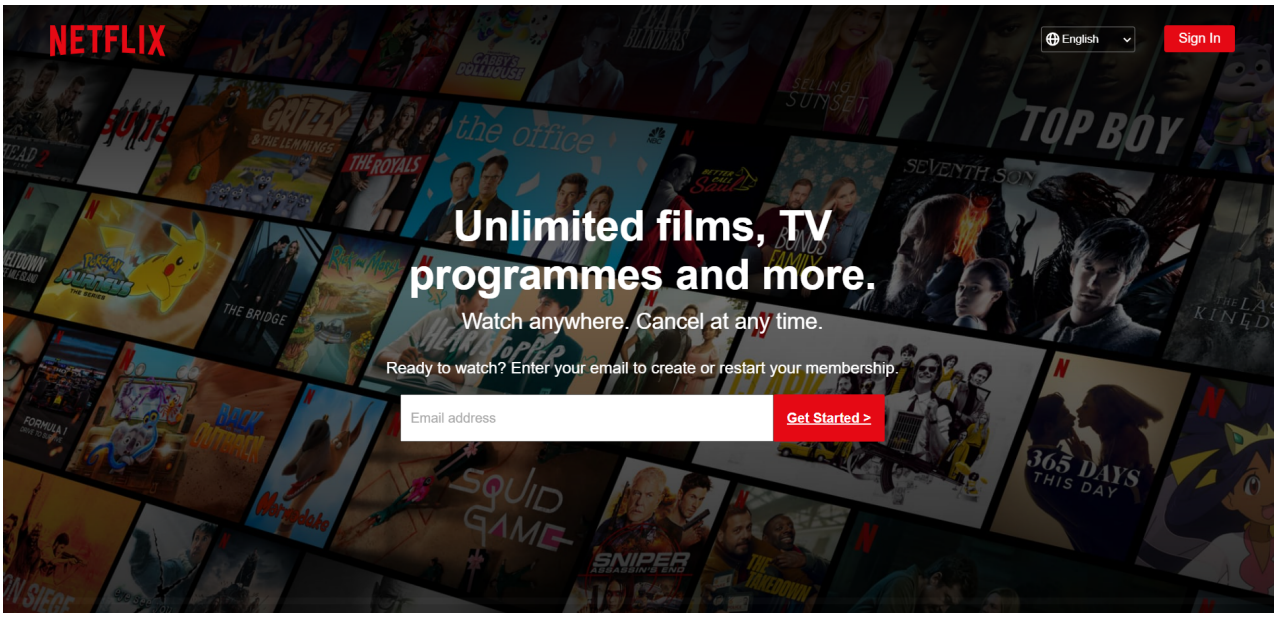

Figure 6: Website main page

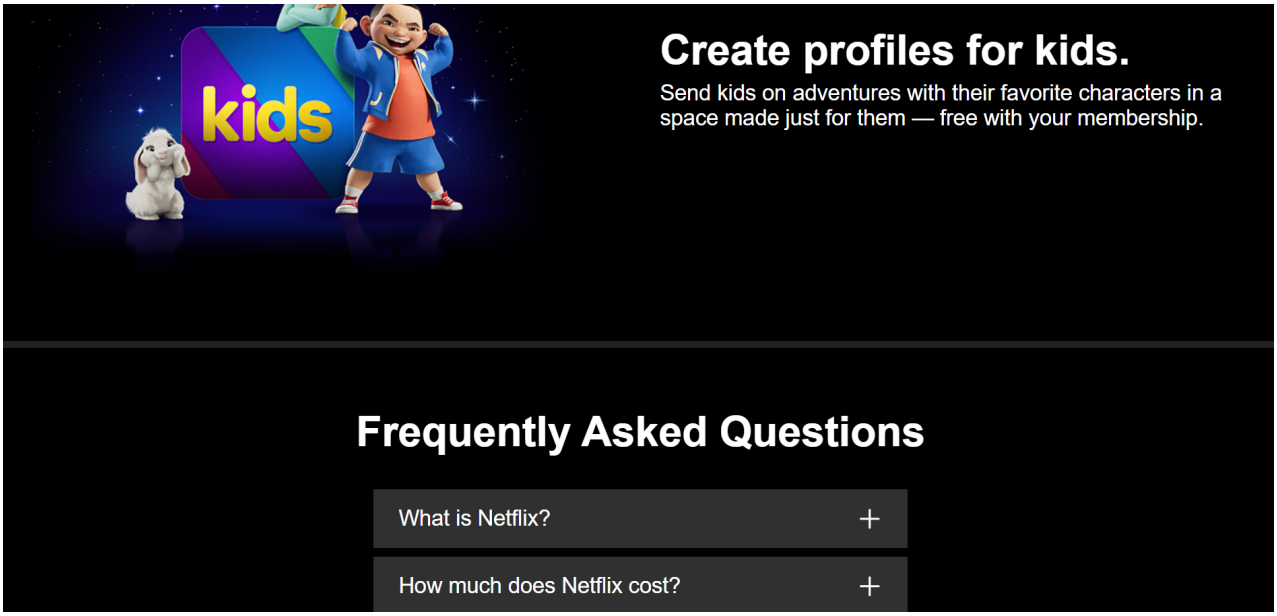

Figure 7: Website main page

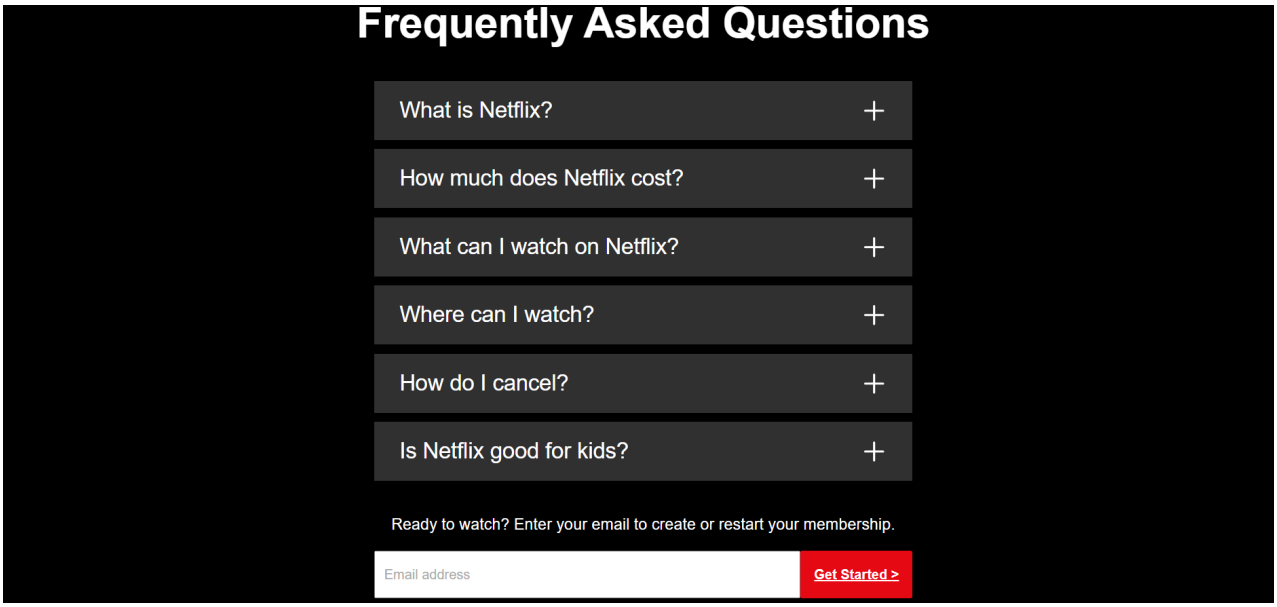

Figure 8: Website main page

7.4.2 Login page

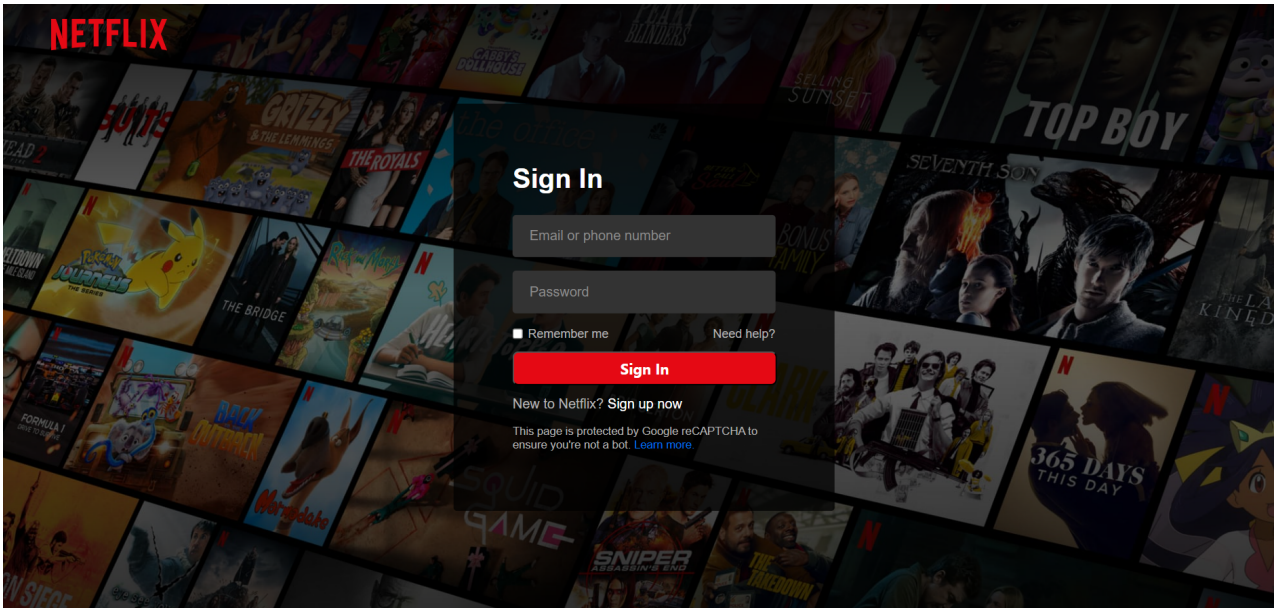

Figure 9: Website login page

7.4.3 Getstarted page

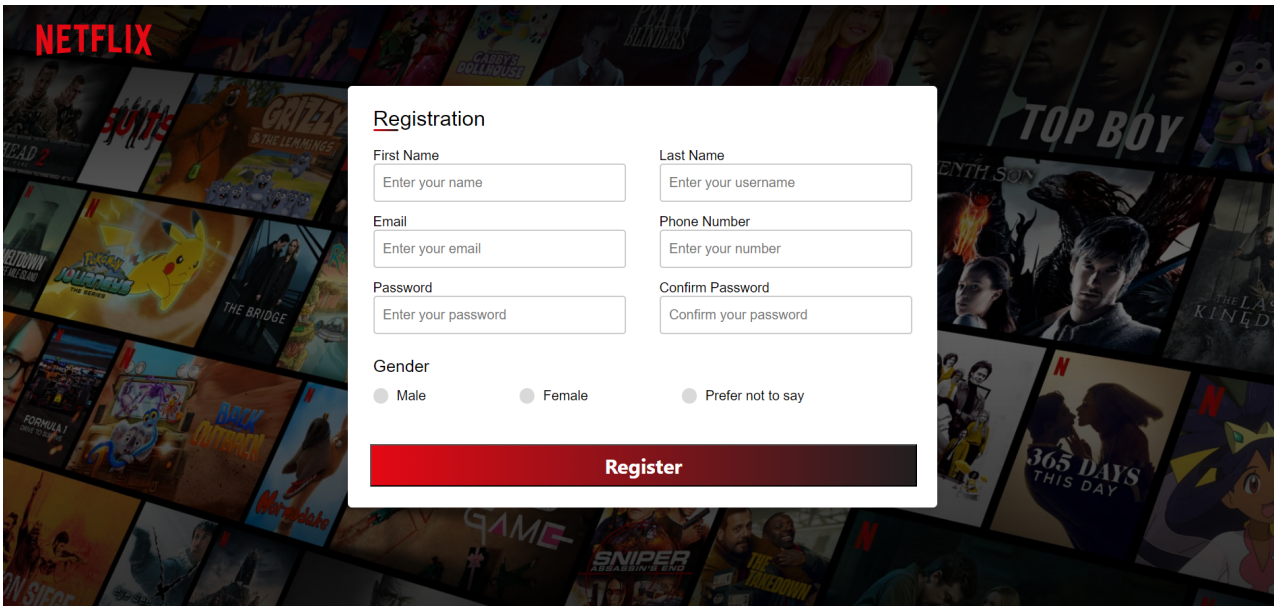

Figure 10: Website getstarted page

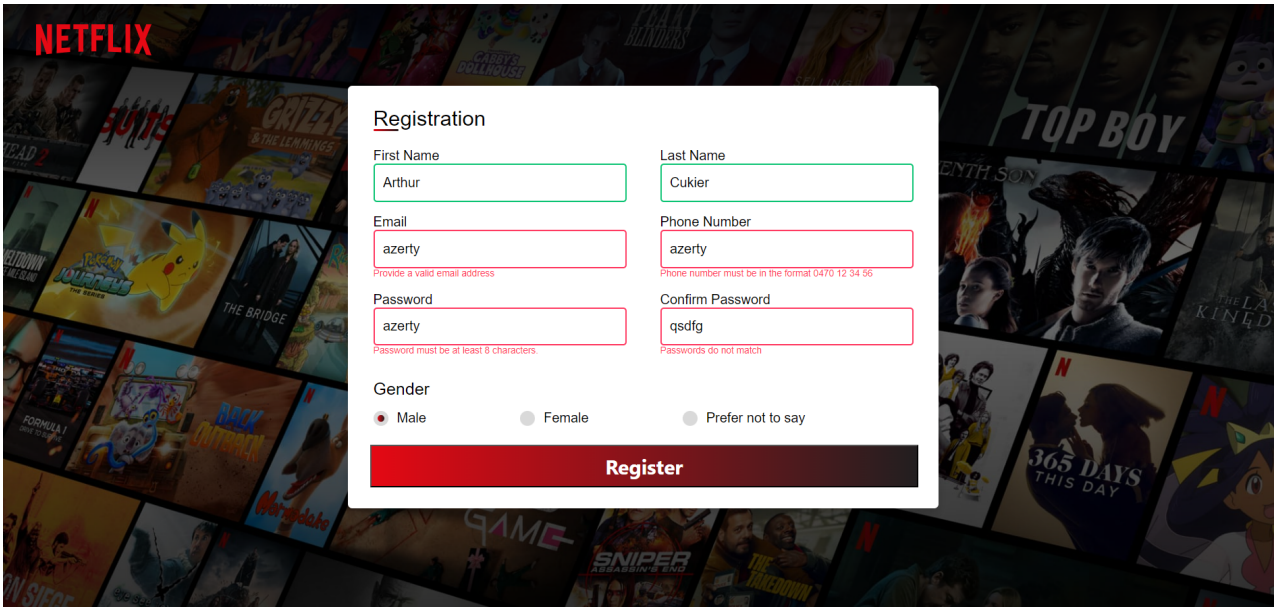

Figure 11: Website getstarted page

7.4.4 Keuze page

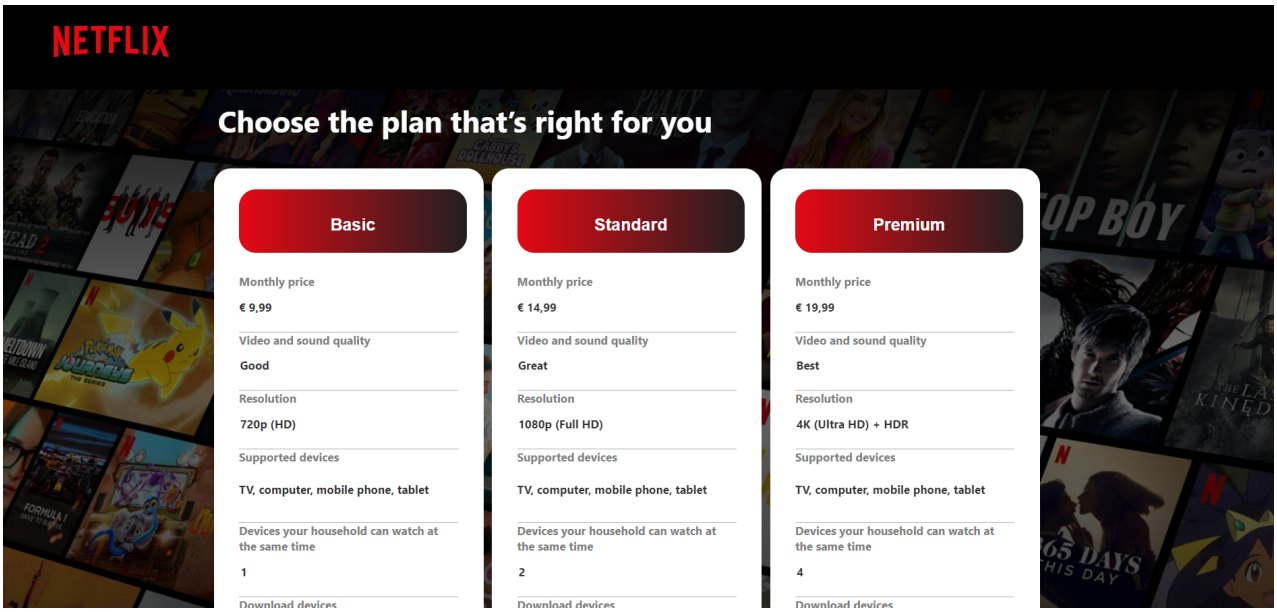

Figure 12: Website keuze page

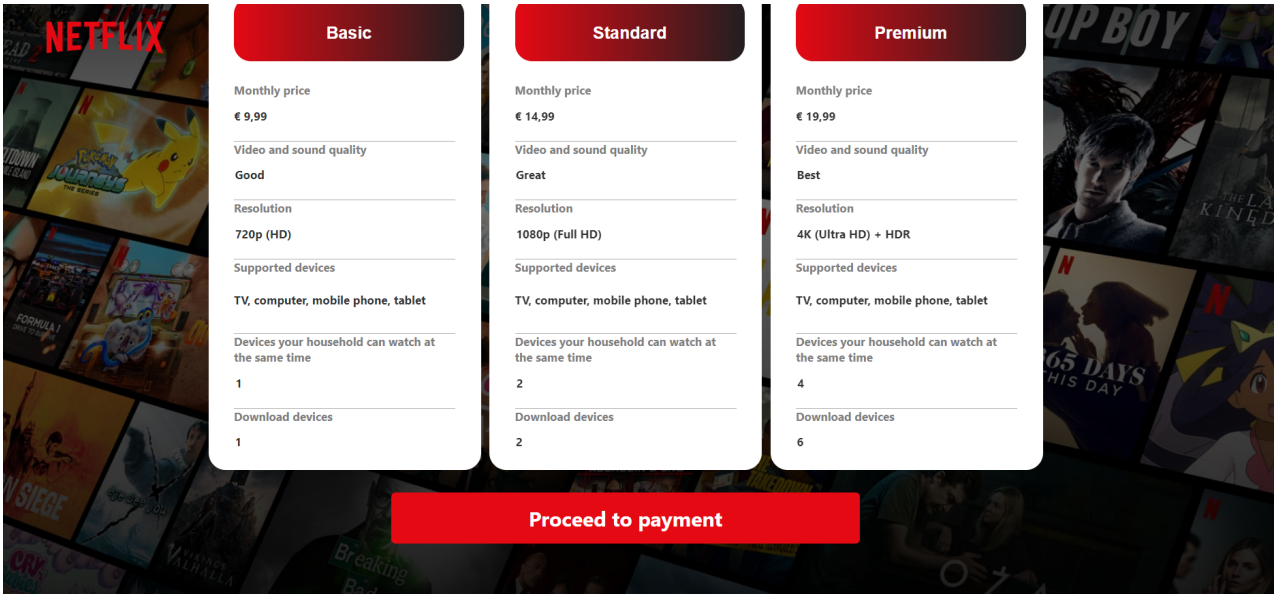

Figure 13: Website keuze page

7.4.5 Bank page

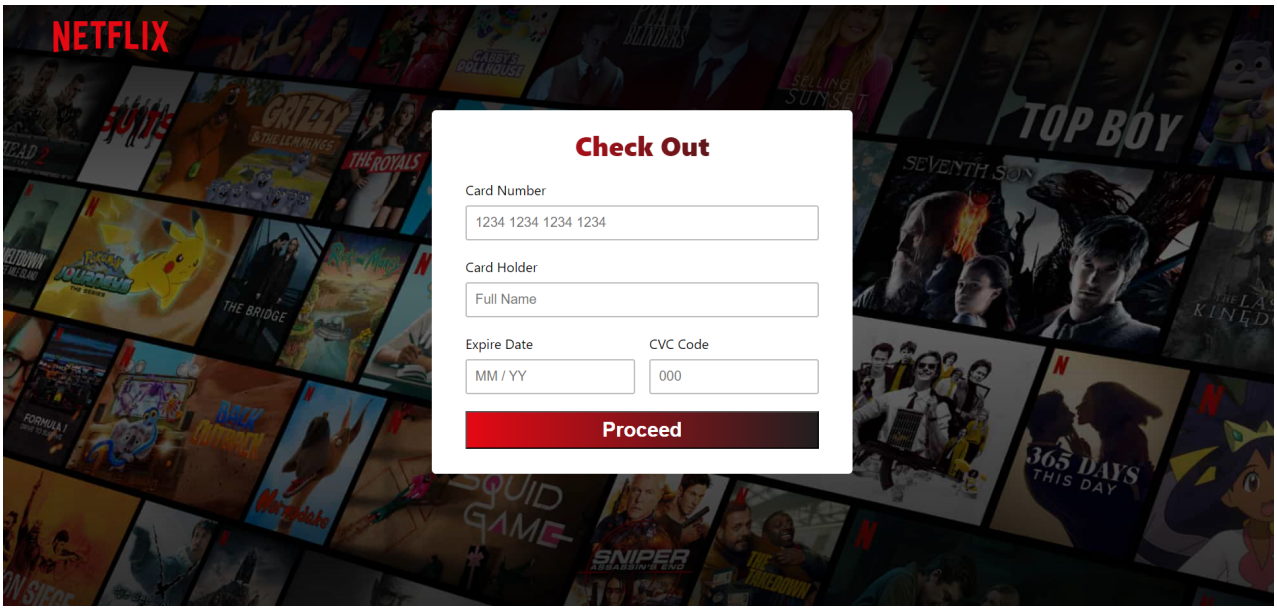

Figure 14: Website bank page

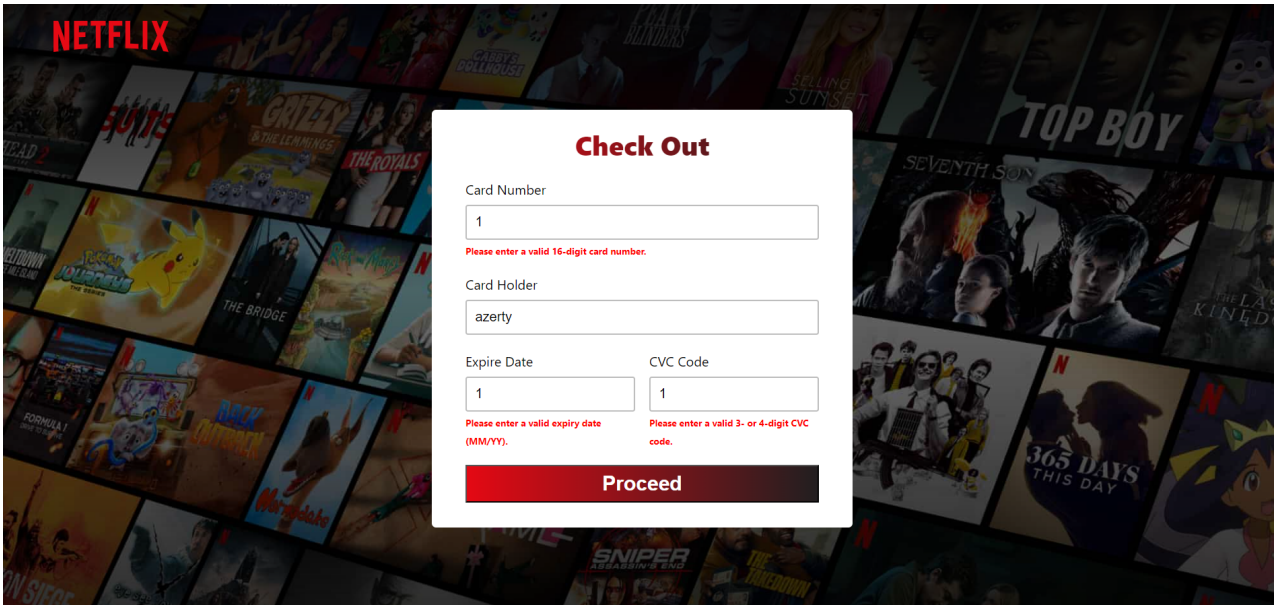

Figure 15: Website bank page

7.4.6 Fout page

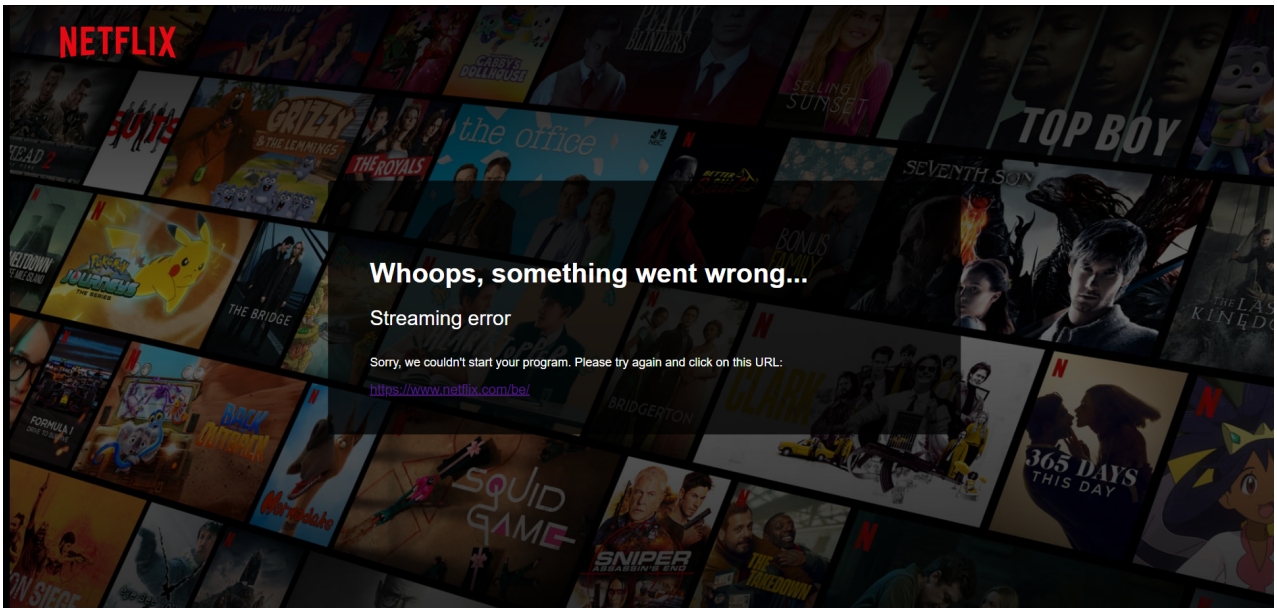

Figure 16: Website fout page