# AUFGABE 4.1
# DIGEST BASIERTE AUTHENTIFIZIERUNG

15. März 2023

Haller Tobias, 5AHIF
Katalognummer: 4
HTBLuVA Wiener Neustadt
Abteilung Informatik

März 2023

# Inhaltsverzeichnis

# Kapitel 1

# Aufgabenstellung

Simulation einer Digest-baiserte Authentifizierung auf einer Website. Dabei werden ein Client und ein Server simuliert, auf der Client-Seite werden die Anmeldedaten per Konsole oder über eine Config-Datei eingelesen. Nach dem Authentifizierungsprozess beendet sich der Client und der Server ist für eine neue Anfrage bereit.

# Kapitel 2

# Digest-basierte Authentifikation - Theorie

### 2.0.1 Allgemeines

Digest-basierte Authentifizierung ist ein Weg für Service-Provider eine Person aufgrund ihrer Anmeldeinformation mit der Benutzung eines Webbrowser zu verifzieren. Digest-basierte Authentifizierung benutzt das HTTP-Protkoll und verwendet MD5-Hashing, sowie eine Nonce (einmalig generierte Zeichenfolge) um wiederholende Attacken abzuwehren. Die generierten Hashes werden an den Benutzernamen und das Kennwort der Person angehängt, bevor sie über das Netzwerk gesendet werden, so dass der Server des Providers die Person authentifizieren kann. Diese Art der Authentifizierung wird der klassichen Basic-HTTP-Authentifizierung vorgezogen, da es unsicheres Base64-Encoding verwendet.

### 2.0.2 Vorteile/Nachteile der Digest-basierten Authentifikation

Die Vorteile dieser Arten der Authentifikation sind:

- Das Passwort wird nicht klar an den Server gesendet.
- Das Passwort ist nicht direkt gehasht sondern in Kombination mit Username, Realm.
- Die Client-Nonce erlaubt, das Verhindern von *Chosen-Plaintext-Attacken*, sowie *Rainbow-Table-Attacken*.
- Die Server-Nonce erlaubt Zeitstempel um wiederholende Attacken zu verhindern. Der Server ist auch erlaubt eine Liste von bereits genutzen Noncen zu speichern
- *Phishing-Attacken* werden durch das Hashing des Passworts verhindert.

Die Nachteile sind:

- Die Webseite hat keine Kontrolle über das UI, welches den End-Benutzer gezeigt wird.
- Viele der Sicherheitsoptionen sind optional.
- Digest-basierte Authentifikation ist für eine *man-in-the-middle-Attacke* anfällig.
- Der Server speichert den HA1-Hash, welcher das Passwort beinhaltet. Damit kann er sich dann bei einem Datenleak trotz Verschlüsselung authentifizieren.
- Die Verwendung von starken Passwörtern wird verhindert, da entweder Passwort, Username oder Realm und Passwort wiederherstellbar sein müssen.

# Kapitel 3

# Implementierung

Bei den Code Beispielen werden diese Namespaces verwendet.

```
using namespace std;
using namespace asio::ip;
```

### 3.0.1 Ablauf der Authentifizierung

1. Zuerst sendet der Client einen String mit ”LOGON” und einer Digest-URI an den Server.
2. Der Server liest diese Anfrage und prüft durch ”LOGON” ob es sich um eine Authentifizierungsanfrage handelt. Der Server prüft dannach ob die angegebene URI auch für eine Verifizierung geeignet ist.
3. Der Server sendet den Client dann den zugehören Realm zur URI und eine zufällig berechnete Nonce.

```
unsigned int generateNonce() {
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<unsigned int> distrib(0, UINT_MAX);
    return distrib(gen);
}
```

4. Der Client liest den Realm und die Nonce und generiert sich mithilfe dieser Informationen und den Credentials des Users die Hashes, den ”Reponse”-Hash, welcher dann gesendet wird. (MD5 wird von einer externen Libary bereitgestellt)

```
string generateHashes(string username, string password, string uri,
                      string realm, unsigned int nonce) {
    string ha1 = username + realm + password;
    string ha2 = "GET" + uri;
    MD5 md5;
    string responseCalc = md5(md5(ha1)+to_string(nonce)+md5(ha2));
    return responseCalc;
}
```

5. Der Server liest den Response und berechnet auch den gleichen Hash, diese beiden Werte werden dann verglichen und wenn sie gleich sind sendet der Server den Client ein ”ACK” mit der Nonce. Wenn nicht gleich wird ein ”NAK” gesendet.
6. Der Client liest die Antwort vom Server, gibt sie auf der Konsole aus und beendet sich.

### 3.0.2 Ablauf der Verbindung des Clients mit dem Server

Zur Verbindung des Clients mit dem Server wird die C++ Libary asio benutzt. Es ist eine Bibliothek zur Netzwerkprogrammierung.

1. Zuerst legt der Server einen IO-Stream an mit dem sich der Server verbinden kann.

```
asio::io_context ctx;
tcp::endpoint ep{tcp::v4(), port};
tcp::acceptor acceptor{ctx, ep};
acceptor.listen();
tcp::socket sock{ctx};
```

2. Danach verbindet sich der Client auf den Server.

```
tcp::iostream stream("localhost", to_string(port));
```

3. Der Server nimmt die Anfrage an. Durch die while-Schleife, beenden sich der Server nach Abarbeitung der Anfrage nicht und ist für weitere Anfragen bereit.

```
while(true) {
    acceptor.accept(sock);
    tcp::iostream strm{move(sock)
};
```

# Kapitel 4

# Quellen

- Digest-basierte Authentifizierung Theorie `https://en.wikipedia.org/wiki/Digest\_access\_authentication`
- Externer Hashing-Libary `https://github.com/stbrumme/hash-library/blob/master/md5.cpp`