Praktikum Verteilte Systeme

Praktikumsaufgabe 5 –

Authentifizierung beim RPC-Publish-Subscribe-System

Ziel dieses Praktikums ist es, für das in Praktikumsaufgabe 4 entwickelte System eine Erweiterung für eine Authentifizierung zu entwickeln. Die Aufgabe muss bis

Montag: 29.04.2019 Dienstag: 30.04.2019 Donnerstag: 02.05.2019

zu 80% fertiggestellt werden.

Einleitung

Der Schnittstelle des Publish-Subscribe-Systems aus der vorherigen Aufgabe fehlte die Möglichkeit, Benutzer an- und abzumelden. Dies soll nun ergänzt werden.
Die Implementierung soll auf Basis eines **Challenge-Response-Protokolls** erfolgen:

- Bei dieser Art der Authentifizierung wird die Übertragung eines Credentials in Form eines Passworts im Klartext vom Client zum Server vermieden.
- Außerdem wird das Passwort nicht direkt auf dem Server gespeichert. Stattdessen verwaltet der Server **Hash-Werte** *H(user;pwd)* aus den Kombinationen von Nutzername *user* und Passwort *pwd* zu dem Nutzernamen in einem sog. *Digest*.

Auf eine Authentifizierungsanfrage antwortet der Server zunächst mit der Übermittlung eines Zahlenwerts *nonce* (*Number used once*). Der anfragende Client antwortet mit einer Nachricht *nonce;user;H(nonce;user;H(user;pwd))*. Der Server kann auf Basis des gespeicherten Hashwertes *H(user;pwd)* den übermittelten Hashwert nun verifizieren ohne das Passwort des Nutzers zu kennen. Die Annahme dabei ist, dass die Hash-Funktion eine **Einwegfunktion** ist, d.h. der Hash-Wert eindeutig ist und keinerlei Rückschluss auf die eingegebene Zeichenkette zulässt. Bei jeder der folgenden Aktionen kann nun auch die Integrität der übermittelten Daten (z.B. RPC-Parameter) server-seitig verifiziert werden. Dazu werden im Client Nachrichten der Form *nonce;data;H(nonce;data;H(user;pwd))* generiert und an den Server übertragen. Setzen Sie dieses Protokoll nun für das bestehende Publish-Subscribe-System um. Gehen Sie dabei in zwei Schritten vor:

5.1 Erweiterte Schnittstellendatei (3 Punkte)

Erweitern Sie zunächst die Schnittstellen-Datei. Es werden 3 neue Funktionen

```
sessionid get_session (user) = 5;
short validate (param) = 6;
short invalidate (sessionid) = 7;
```

benötigt. Der zurückgegebene nonce-Wert der Funktion `get_session` wird als Sitzungsschlüssel interpretiert. Da bei einem RPC nur ein Parameter übertragen werden kann, wird eine Datentyp `param` eingeführt, der für die bislang vorhandenen Funktionen genutzt werden kann. In Abwandlung des o.g. generellen Ablaufs wird der Nutzername nicht in `param` übermittelt, sondern zusammen mit der Session im Server gespeichert.

```
const           USERLEN = 12;
const           HASHLEN = 96; /* bei SHA 256 */
typedef string  hashstring <HASHLEN>; /* Parameter */
typedef string  user <USERLEN>; /* User */
typedef int     sessionid;

union argument switch (int topic_or_message) {
    case 0: topic t; /* topic wird uebertragen */
    case 1: message m; /* message wird uebertragen */
    default: void; /* kein Parameter */
};
```

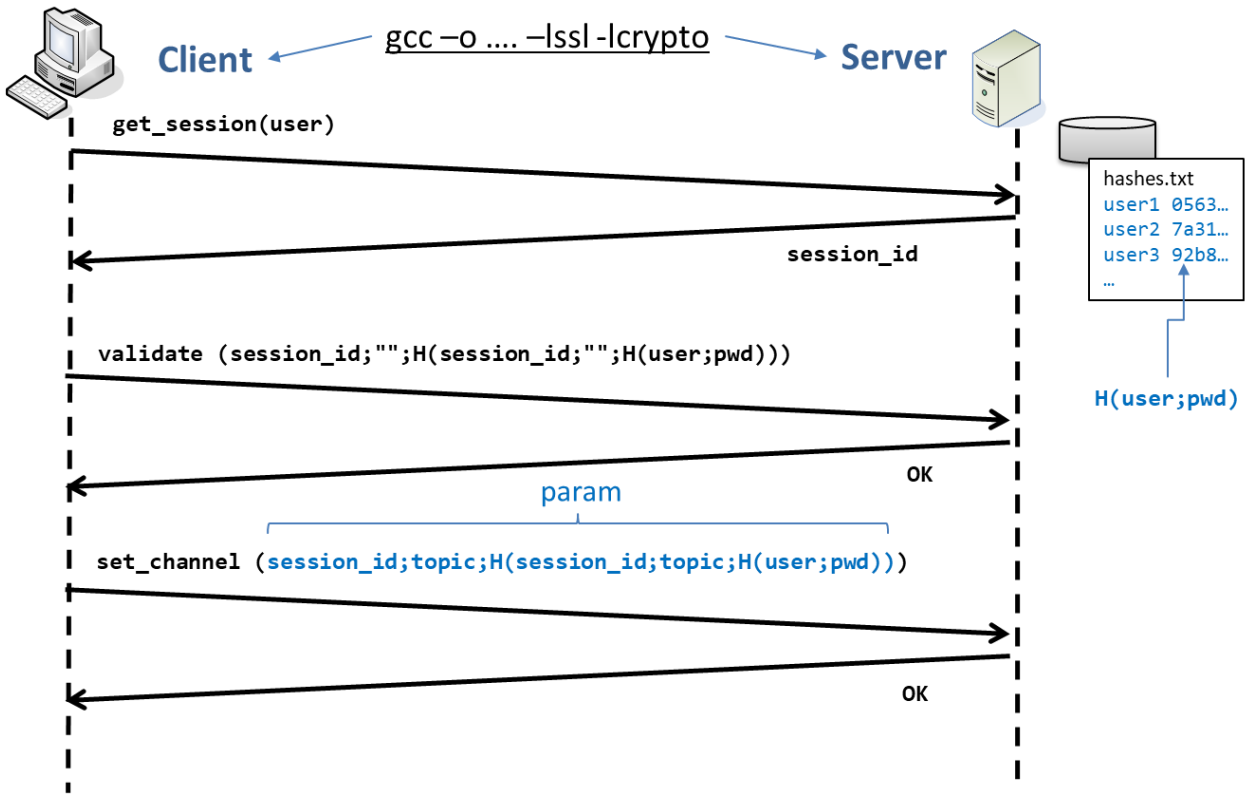
```
struct param {
    sessionid id;
    argument arg;
    hashstring hash;
};
typedef struct param;
```

Ergänzen Sie aussagefähige Rückgabewerte in **`return_code.h`** und besprechen Sie Ihren Ansatz mit einem Betreuer.

5.2 Implementierung und Test (7 Punkte)

Der Ablauf der Authentifizierung mit Hilfe dieser Funktionen wird in dem Sequenzdiagramm unten dargestellt. Bei einem Login werden vom Client aus nacheinander die Funktionen **`get_session()`** und **`validate()`** aufgerufen. Bei einem Logout die Funktion **`invalidate()`**.

z



- Für die Generierung des Nonce-Wertes kann die System-Funktion **`clock()`** verwendet werden, welche die aktuelle CPU-Zeit als eindeutigen Wert liefert.
- Als Hashfunktion H soll SHA-256 verwendet werden. In OSCA finden Sie **`sha_hashing.c`** mit Angaben zur Nutzung der Bibliothek und der SHA-Funktion. Dort wird auch der Umgang mit dem Hash-Digest in einer Datei hashes.txt gezeigt. Einträge haben das Format:
  `student db398679a8775d55b61228cbeb8dff61a86dbe5d3b68557e5a4c36a927c17e5f …`
  `hashes.txt` und **`sha_hashing.c`** werden in OSCA als **`Input_prakt05.zip`** bereitgestellt.
- Es wird empfohlen, die als Hashwert übermittelten Credentials zu Anfang im aufgerufenen RPC durch eine zentrale Funktion check_session() im Server zu überprüfen. Des Weiteren wird im Server eine Datenstruktur (Liste oder Dictionary) zur Zuordnung der Sessions zu Nutzern benötigt, so dass nach geglückter Authentifizierung nur die Session-ID bei einem RPC übermittelt werden muss.

Hinweis zur Fehlersuche:

Kompilieren Sie mit **`gcc`** und **`-g`** zur Erzeugung von Debug-Informationen. Speicherprobleme können mit **`valgrind`** gefunden werden: **`valgrind -v <ausführbare Datei>`**
Auf manchen Linux-Systemen muss der Aufruf mit **`/usr/bin/valgrind`** erfolgen.