Hüseyin Bozkurt

Die Dokumentation des Ersten Sprints:

Evaluation

Sprint01

Evaluation-Sprint

Contents

[1) Changelog 1](#_Toc422324491)

[2) Sprintdaten 1](#_Toc422324492)

[4) Kommentare 2](#_Toc422324493)

[C 3](#_Toc422324494)

[Visual C++ 3](#_Toc422324495)

# 1) Changelog

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Datum* | *Name* | *Unterschrift* |
| Erstellt | 16.06.2015 | Hüseyin Bozkurt |  |
| Geprüft | 17.06.15 | Erik Brändli |  |
| Freigegeben |  |  |  |

# 2) Sprintdaten

In diesem Sprint wurden US Nr: 1, 3, 9, 10, 12 bearbeitet.

Dauer: 2 Monate.

Zurückzuverfolgen auf taiga.io

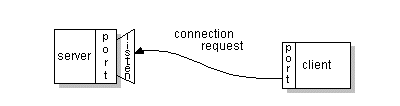
# 4) Kommentare

**US#1 Sockets:** Ließen sich nur schwer per Hand Ver/Entschlüsseln und man muss sich um die übertragenen Daten komplett selbst kümmern => Kein Mechanismus

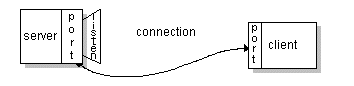
Sockets sind recht einfach zu verwenden, wenn sie nur zur Datenübertragen verwendet werden. Es muss immer klar sein was nun kommt, denn es gibt nichts das die Daten interpretiert (muss man selbst schreiben).

Normalerweise läuft ein Server auf einem Spezifischen Computer und hat einen Socket, der an einen bestimmten Portnummer gebunden ist. Der Server wartet einfach, wartet auf eine Verbindungsanfrage.

Auf der Seite des Clients: Der Client kennt den Hostnamen der Machine auf dem der Server läuft und die Portnummer auf dem der Server aktiv ist. Um eine Verbindungsanfrage zu starten, versucht sich der Client sich mit dem Server verabzureden(am Port des Servers). Der Client muss sich auch bei dem Server identifizieren. Somit wird der Client auf eine lokale Portnummer zugewiesen, welche der Client auf die Dauer der Verbindung nutzt. Dies wird normalerweise vom System angewendet.



Wenn alles gut geklappt hat, erlaubt der Server den Zugriff. Nach der Erlaubnis kriegt der Server ein neues Socket gebunden an den selben Port und sein Remote Endpunkt wird zu der Clientadresse und Port gesetzt. Der Server braucht einen neuen Socket, sodass er wieder für den Originalen Port zuständig sein kann, während der angemeldete Client ungestört Arbeiten kann.



Auf der Seite des Client, wenn die Verbindung erlaubt wird, wird ein Socket erfolgreich erstellt und der Client kann den Socket zum kommunizieren mit dem Server verwenden.

Nun können Server und Client durch Sockets miteinander kommunizieren.

**Code-Beispiele:**

### C

int listenfd = socket(AF\_INET, SOCK\_STREAM, 0);

printf("Socket retrieve success\n");

struct sockaddr\_in serv\_addr;

memset(&serv\_addr, '0', sizeof(serv\_addr));

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

serv\_addr.sin\_port = htons(port);

bind(listenfd, (struct sockaddr\*)&serv\_addr,sizeof(serv\_addr));

if(listen(listenfd, 10) == -1)

{

printf("Listen Failed \n");

return -1;

}

int connfd = accept(listenfd, (struct sockaddr\*)NULL ,NULL);

### Visual C++

int sock = 0;

if ((sock= socket(AF\_INET, SOCK\_STREAM, 0))< 0)

{

printf("\n Error : Could not create socket \n");

return 1;

}

struct sockaddr\_in serv\_addr;

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(port); // port

serv\_addr.sin\_addr.s\_addr = inet\_addr(host);

/\* Attempt a connection \*/

if (connect(sock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr))<0)

{

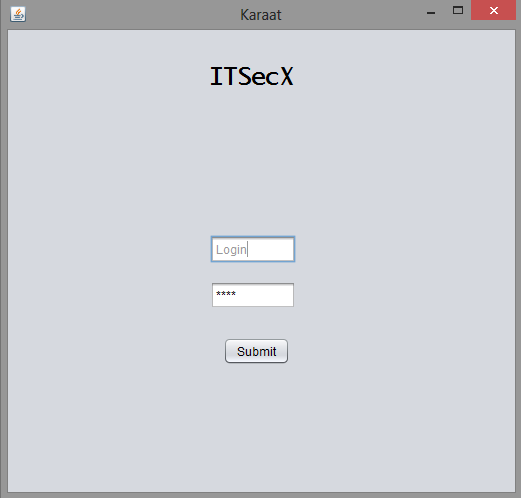
printf("\n Error : Connect Failed \n");

return 1;

}

send(sock, command, strlen(command), 0);

WSACleanup();

**US#3 GUI-Recherche:** Da sich die mehrheit per demokratische Abstimmung für die Entwicklung der Graphischen Oberfläche für Java entschieden hat, wurde von H.Bozkurt eine Testversion mithilfe von der NetBeans DIE (8.0.2-JavaFX) erstellt (siehe Technologien/NetBeans). 

**Aufbau des Startframes:**

Label zur Beschriftung ("ITSecX")

C:\Users\Hüseyin\Desktop\as.PNG => Jlabel1.Textfield

Login Textfeld

C:\Users\Hüseyin\Desktop\as.PNG => tf1.Textfield

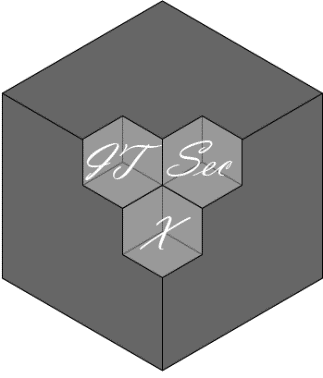
Passwort Textfeld

C:\Users\Hüseyin\Desktop\as.PNG => tf2.Textfield

Submit Button

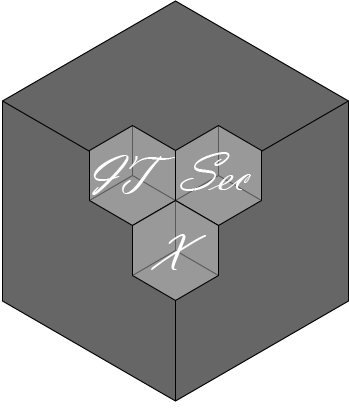
C:\Users\Hüseyin\Desktop\as.PNG => button1.Button

Auf der aktuellen Version ist das Logo der ITSecX ebenfalls implementiert

 => Logo

Dieses Design wurde von Hüseyin Bozkurt überdacht und implementiert. Der Grund warum wir keine großen Farbreräume ausgewählt haben war schlicht, dass das Thema Netzwerksicherheit eher ein düsteres Scheinbild verbreitet. Die Farben, die verwendet werden sind hauptsächlich Mischungen aus allen 3 Komplementärfarben (Weiss-Grau-Schwarz). Das Hauptfenster dieser Implementierung war dem Endprodukt sehr ähnlich.

Dieses Logo wurde von Hüseyin Bozkurt in Kooperation mit Markus Schulmeister entwickelt. Die Hauptidee, ein Würfel in 3 Räume einzuteilen, soll ein Herz wiederspiegeln. Dieses Herz soll die Liebe zur IT und Netzwerksicherheit wiederspiegeln. Dabei wird jede Kammer mit den einzelnen Teilbereichen unseres Themas gefüllt ("IT", "Sec", "X"). Die Farben wurden natürlich der Graphischen Oberfläche angepasst. Ebenfalls düsteres Thema => düstere Farben.

****