

Projektantrag frontg8

1. Datum: 20. Februar 2015, Ausgabe 1.2

2. Team

- Felix Morgner, fmorgner@hsr.ch
- Tobias Stauber, tstauber@hsr.ch
- Ueli Bosshard, ubosshar@hsr.ch

Beratungs- und Review-Termine ankreuzen, an denen alle Teammitglieder anwesend sein können:

☐ Do 08-09	☐ Do 09-10	☒ Do 10-11	☒ Do 11-12
☐ Do 13-14	☐ Do 14-15	☐ Do 15-16	☐ Do 16-17
☒ Fr 10-11	☒ Fr 11-12	☒ Fr 13-14	☒ Fr 14-15

3. Antrag Virtueller Server

- x Ja, wir hätten gerne ein virtuellen Server und zwar vom Typ: **Ubuntu Linux 14.04 [SE2-BuildServer]**

4. Motivation

Auf der Suche nach einer Text-Chat-Lösung zwischen mobilen Geräten (Bsp. WhatsApp oder Threema) haben sich folgende Kriterien herauskristallisiert, die uns wichtig erscheinen:

1. Sichere Kommunikation, Ende-zu-Ende verschlüsselt und Partner identifizierbar
2. Serverkomponente selber betreibbar
3. Quellcode offen, d.h. kann eingesehen und überprüft werden.
4. Unterstützung für mehrere Geräte pro Benutzer

Uns ist keine Lösung bekannt, die diese Kriterien umsetzt. Darum möchten wir in die Lücke springen. Technologisch konzentrieren wir uns auf sichere Ende-zu-Ende Kommunikation zwischen den Teilnehmern.

5. Programmidee

Unsere Lösung für ein betreiberunabhängiges Text-Chatsystem besteht aus einer Serverkomponente für POSIX konforme Systeme und einer Desktop-Clientkomponente. Fernziel, ausserhalb des SE2 Projekts, sind auch Clients für Android und iOS.

Der Server kümmert ausschliesslich um den Austausch von Text-Messages und wird bewusst sehr schlank gehalten. Dabei ist uns wichtig, dass diese zentrale Stelle möglichst keine Angriffsfläche bietet, d.h. es werden keine Informationen gespeichert, die nicht absolut nötig sind. Die Nachrichten werden in einem flüchtigen Puffer für maximal 3 Tage aufbewahrt.

Der Client ist eine CLI-Applikation und kümmert sich komplett um die Ende-zu-Ende Verschlüsselung und die Identifizierung des Kommunikationspartners mittels Zertifikaten. Er verfügt über ein Adressbuch, damit die „anonymen“ Kommunikationspartner identifiziert und überprüfte Zertifikate zugeordnet werden können.

Unsere Zielpublikum sind geschlossene Benutzergruppen mit sehr hoher Anforderung an Privatsphäre.

6. Realisierung

- Plattform: POSIX Compliant auf Server und Desktop-Client
- Programmiersprachen, Bibliotheken und Technologien: C++, Python, X.509, TLS
- Entwicklerwerkzeuge: git, Redmine (HSR-Build-Server)
- Benutzerschnittstelle: CLI auf Client, Config-File für Server
- Art der Datenhaltung: Flat-Files