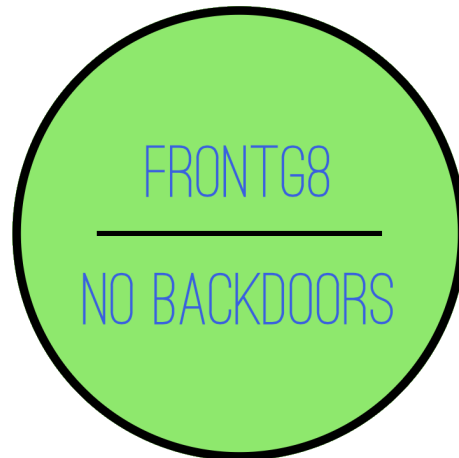




Projekt: frontg8

Handbuch frontg8d

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



Änderungsgeschichte

Datum	Version	Änderung	Autor
28.05.15	1.0		Felix Morgner



Inhalt

Änderungsgeschichte	2
Inhalt.....	3
1. Installation	4
1.1 Requirements	4
1.2 Eigene Build-Umgebung einrichten.....	4
1.3 ASIO	5
1.4 CUTE	5
1.5 Protobuf 2.5¶	5
1.6 Kompilieren des Servers.....	5
2. Konfiguration	5
2.1 Konfigurationsdatei	5
3. Benutzung	6

1. Installation

1.1 Requirements

- CMake (mit cmake)
- C++-Compiler
- git

1.2 Eigene Build-Umgebung einrichten

Umgebung unter Linux einrichten:

```
# mkdir toolchain
# cd toolchain
# git clone https://github.com/llvm-mirror/llvm.git
# cd llvm
# git checkout release_36
# cd tools
# git clone https://github.com/llvm-mirror/clang.git
# cd clang
# git checkout release_36
# cd ../..
# cd projects
# git clone https://github.com/llvm-mirror/compiler-rt.git
# cd compiler-rt
# git checkout release_36
# cd ..
# git clone https://github.com/llvm-mirror/libcxx.git
# cd libcxx
# git checkout release_36
# cd ..
# git clone https://github.com/llvm-mirror/libcxxabi.git
# cd libcxxabi
# git checkout release_36
# cd ..
# cd ../..
# mkdir build
# cd build
# cmake ../llvm
```

c drücken. Nach dem cmake die initiale Konfiguration gemacht hat, folgende Änderungen vornehmen:

- CMAKE_BUILD_TYPE auf Release stellen
- LIBCXX_ENABLE_ASSERTIONS auf Off stellen
- LLVM_ENABLE_LIBCXX auf On stellen
- LLVM_ENABLE_RTTI auf On stellen
- LIBCXX_INSTALL_HEADERS auf On stellen
- CMAKE_INSTALL_PREFIX auf /opt/clang-toolchain stellen
- LLVM_TARGETS_TO_BUILD auf X86 stellen

Nochmal c drücken. Nach dem cmake die erneute Konfiguration durchgeführt hat, g drücken.
wenn cmake fertig ist, make starten

danach mit

```
# sudo make install
```

installieren

1.3 ASIO

Asio 1.10.6 (non-boost) herunterladen (<http://think-async.com/Asio/Download>) und entpacken. aus dem verzeichniss include den Ordner asio und die Datei asio.hpp nach /usr/local/include kopieren.

1.4 CUTE

CUTE herunterladen (http://cute-test.com/projects/cute/wiki/CUTE_standalone) Version 2.0.0 und entpacken. Den Ordner cute_lib als cute nach /usr/local/include kopieren.

1.5 Protobuf 2.5¶

Protobuf 2.5 herunterladen (<https://protobuf.googlecode.com/files/protobuf-2.5.0.tar.bz2>) und entpacken.

```
cd protobuf-2.5.0
./configure CC=clang CXX=clang++ CXXFLAGS='-std=c++11 -stdlib=libc++
-O3 -g' LDFLAGS='-stdlib=libc++' LIBS="-lc++ -lc++abi"
sudo make install
```

1.6 Kompilieren des Servers

Der Sourcecode ist frei erhältlich, ein git-Clone kann mit dem folgenden Kommando erstellt werden:

```
git clone https://source.arknet.ch/frontg8/server.git
```

Anschliessend wird das Binary mit folgendem Script gebaut:

```
cd server/build
./build
```

2. Konfiguration

2.1 Konfigurationsdatei

In der Konfigurationsdatei „**frontg8d.conf**“ können folgende Einstellung verändert werden:

- | | |
|------------------------|---|
| • persistence_interval | Wie oft die Daten in ein File geschrieben werden |
| • cleanup_interval | Wie oft wird überprüft, ob alte Nachrichten vorliegen |
| • message_lifetime | Nach welcher Zeit dürfen alte Nachrichten gelöscht werden |
| • root | Legt den Pfad zu den Daten fest |
| • tls | Legt den Pfad zu den verwendeten TLS-Zertifikaten fest |
| • port | Legt den verwendeten Server-Port fest (TCP) |
| • address | Legt die Listen IP-Adresse fest |



3. Benutzung

Starten des Servers:

```
user@localhost:~ $ ./frontg8d
Please enter the journal username: username
Please enter the journal password:
[INFO] frontg8d v0.0.1 "Twoflower" starting to listen on
0.0.0.0:10000
```

Unmittelbar nach dem Start wird nach Benutzer und Passwort gefragt, der Account wird benötigt, damit das Message-Journal entschlüsselt werden kann.