Installing OWS Signal 1.88
Ubuntu & Android

Dependencies

Shortcut -> Run *./install-dependencies.sh*

If your script executes correctly skip to the twilio section.

If your script fails for some reason you will need to do the following sections to find out what happened.

Java

Java 1.8 is the version specified in the config files; I used openJDK version 8 and can confirm it works, other versions of java did not work for me.

- Download: *sudo apt install openjdk-8-jdk*
- List installed JDK's: *update-java-alternatives --list*
  - *Output*

```
java-1.11.0-openjdk-amd64     1101     /usr/lib/jvm/java-1.11.0-openjdk-amd64
java-1.8.0-openjdk-amd64      1081     /usr/lib/jvm/java-1.8.0-openjdk-amd64
```

- Select JDK: *sudo update-java-alternatives --set /path/to/your/jdk*
  - Note: if your output was the same as mine, you would use the command: *sudo update-java-alternatives --set /usr/lib/jvm/java-1.8.0-openjdk-amd64*
- You will want to be sure you have java 1.8 as other versions of java did not work for me. (Java 1.11)

PostgreSQL instance

- PostgreSQL setup help
- Note: if you see this error,

```
psql: FATAL:  peer authentication failed for user "postgres"
```

  - Start a postgres server: sudo -u postgres psql postgres

        - Add a password by \password postgres
          You will need to type this password plaintext in the servers config files, so use a throw away password.
        - You will need to change it from peer authentication to password authentication. Stack Overflow
        - /etc/init.d/postgresql reload
        - TODO create user
    - Note: to drop the databases
        -
        - Login: sudo -u postgres psql postgres
        - drop database accountsdb;
        - drop database messagedb;
        - \list to make sure they are gone

Redis (For caching purposes)

- Download
- Run Redis
    - redis-server &

Turn Server

- sudo apt-get install coturn
- sudo turnserver -a -o -v -n --no-dtls --no-tls -u test:test -r "someRealm"
- Help

Memcached

- sudo apt-get install memcached

Twilio (SMS Messaging)

Note: You can skip this step, we modified the server code to print the verification code to the console. If you still want text message verification codes you would need to set this up.

- We have an lab account. Ask someone with access (Austin, Aaron, Tarun, or Dr. Seamons for an access email)

- The Signal Server config file needs a twilio accountId and accountToken.
    - You can find these values by clicking on the home icon and then clicking on dashboard.
    - You should see something that says project info.
    - Account SID is your accountId and AUTH TOKEN is your accountToken.
    - Copy these values into the server config/*.yml file.
- Since it is a trial account, each time we want to use a new phone number we need to add it to a verified caller ID on the twilio website.
    - Click the **#** icon on the left hand side.
    - Click Verified Caller IDs
    - Add the new phone number you are using.

Amazon AWS S3 (message attachments)

Note: You can skip this step, if you want to send message attachments you would need to set this up.

- You can create a free educational account with our lab emails.
    - We do not currently use this feature.

GCM (Google Cloud Messaging: for Android notifications)

Note: You should be able to use our GCM ID already in the config files, this is incase something goes wrong.

- You can create a GCM account at this link.
- Your Project number is your senderId for the config file.
- Great website for finding your Project number and API key for the config file.

Note: You should be able to just use the credentials already in the config file for GCM. This is just in case you need to make new credentials.

Note1: Google may be making this complicated in the future as they have deprecated GCM and are replacing it with Firebase Cloud Messaging (FCM).

APN (Apple Push Notifications for iOS)

- We have modified the code to avoid using this dependency.

Getting the Sources

About

- I forked the signal code for version 1.88 and made a few changes to have the code run.
  - Added credentials to server config file. Signal-Server/config/sample.yml
  - Commented out APN code.
  - Changed the Android App to use our server instead of a OWS server.

Source

- Project is located on the BYU ISRL repository.
- Called signal-server-components (may need to ask Dr. Seamons, Austin, Aaron, or Tarun for access)
- Clone the repository.
- Server code is located at ows1.88/Signal-Server
- Modified MITM Server code is located at ows1.88/Signal-Server-MITM
- Android code is located at ows1.88/Signal-Android

Building TextSecure-Server

```
cd ows1.88/Signal-Server
mvn package -DskipTests
```

Note: Use -DskipTests as Signals test code does not work.

Or…

```
./build.sh
```

Initializing PostgreSQL Databases

```
sudo createdb -U postgres accountdb
sudo createdb -U postgres messagedb
```

Configuration Files

- Server configuration file should be located config/sample.yml

- Hopefully most of the values will be the same ones you will use, but you may need to check that everything matches up.

Import Table Schemas

- Once your config files have the correct values you will need to populate your postgres databases. To do so run the following commands.

```
java -jar target/TextSecureServer-1.88.jar accountdb migrate config/sample.yml
java -jar target/TextSecureServer-1.88.jar messagedb migrate config/sample.yml
```

- Each time you make a change to your YML file, it is good practice to delete the accountdb and messagedb databases, recreate them and then re-run these commands. Shortcut is run `./refresh-postgres.sh`

Run the Server

- In the Signal-Server folder run the following command.

  ```
  java -jar target/TextSecureServer-1.88.jar server config/sample.yml
  ```

  Or…

  ```
  ./run.sh
  ```

- Verify that your server runs without any errors.

Android

- Open the project up in android studio
- In your build.gradle file look for SIGNAL_URL and SIGNAL_SERVICE_STATUS_URL
  - Change the OWS url to the domain name of the computer you are running the server on.
    In Terminal: `nslookup <your ip address>`

Example:

```
buildConfigField "String", "SIGNAL_URL", "\"https://textsecure-service.whispersystems.org\""
```
to
```
buildConfigField "String", "SIGNAL_URL", "\"https://johnvonneumann.cs.byu.edu:8080\""
```

```
buildConfigField "String", "SIGNAL_SERVICE_STATUS_URL", "\"uptime.signal.org\""
```
to
```
buildConfigField "String", "SIGNAL_SERVICE_STATUS_URL",
"\"https://johnvonneumann.cs.byu.edu:8080\""
```

- Verify that you can install the android app on a device or emulator.

Using HTTPS

- You server and android application should be running, but if you attempt to register it will break. The reason is the Android Application forces the use of HTTPS. To be able to register and send messages we need to set up HTTPS. The following steps will go over setting up HTTPS for Signal on a local area network (LAN).

Note: Skip steps 1-5 by running `./create-certificates.sh`
Note: To create a certificate with IP address in the script added IP.1 = ${domain}

1. Creating your own Certificate Authority (CA)

- In terminal run the following two commands

```
openssl genrsa -des3 -out myCA.key 2048
```

```
openssl req -x509 -new -nodes -key myCA.key -sha256 -days 1825 -out myCA.pem
```

The first command creates the private key for your certificate authority. While the second command creates the CA's certificate.

(With Current Server configs this is false, but doesn't hurt to do it anyway)
Your CA certificate needs to be an all devices that use certificates signed by your CA. If you upload your certificate to google drive and open it on a android device it will help your install the certificate on your device. You may need to restart your device for the certificate to work.

2. Creating certificates for your Signal Server

- In terminal run the following two commands

```
openssl genrsa -out johnvonneumann.cs.byu.edu.key 2048
```

```
openssl req -new -key johnvonneumann.cs.byu.edu.key -out johnvonneumann.cs.byu.edu.csr
```

Again the first command creates a private key for the signal server certificate and the second command creates a certificate for signal to use. I named the certificates johnvonneumann .cs.byu.edu because that is the domain name of my computer. I believe you can name this whatever you like.

3. Signing the Signal Server certificates with our CA

- Our signal certificates need to be signed by our CA to be considered valid. To do this run the following command in the terminal.

```
openssl x509 -req -in johnvonneumann.cs.byu.edu.csr -CA myCA.pem -CAkey myCA.key
-CAcreateserial -out johnvonneumann.cs.byu.edu.crt -days 1825 -sha256 -extfile config.ext
```

  Where config.ext is just a text file I created containing the following.

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = johnvonneumann.cs.byu.edu
```

  NOTE: DNS.1 is the domain name of my computer. Change this to the domain name of your computer. You can find this by doing a nslookup followed by your ip address.

4. Exporting to PKCS12 (Needed for next step)

- Run the following command in terminal changing the names of the files as appropriate.

```
openssl pkcs12 -export -password pass:password -in johnvonneumann.cs.byu.edu.crt -inkey
johnvonneumann.cs.byu.edu.key -out keystore.p12 -name example -CAfile myCA.pem
```

5. Exporting to Java Keytool

- The signal server expects its certificates in the java keytool format.

```
keytool -importkeystore -srckeystore keystore.12 -srcstoretype PKCS12
-destkeystore keystore.jks
```

  This should create a file keystore.jks. This file will be used in your server config file.

  Note: If you are having problems check out this post.
  [Signal Community - Server on HTTPS]

6. Change config files from HTTP to HTTPS

- In the root directory of signal server open up the config folder.
- Open sample.yml
- If you see

```
server:
applicationConnectors:
- type: http
  port: 8080
```

  Your server is configured to run HTTP.

- Replace with

```
server:
applicationConnectors:
- type: https
  port: 8080
 keyStorePath: keystore.jks
 keyStorePassword: password
 validateCerts: false
```

  Note: If your keyStorePassword does not match your password for your private key it will throw an exception. To get around this change your keystore password. The command below worked for me.

```
keytool -storepasswd -new password -keystore example.keystore
```

  Note1: validateCerts: false is a little hand wave I did to get around putting in other parameters I did not know how to do. Could be worth looking more into.

7. Putting your server certificate on Signal Android

- Locate the whisper.store file in the project.
  - It should be at *res/raw/whisper.store*
- We need to put our certificate in this file, to do so download Keystore Explorer
  - Keystore Explorer download page
  - In the menu click: File -> Open and open your whisper.store file.
    - Password is whisper

  - Click import trusted certificate. (Red ribbon with blue down arrow)
  - Find the certificate you created for your signal server and import.
  - You should see two entries one for OWS’s server and one for your server.
  - Save the file (do not believe it autosaves)
  - Open up whisper.store in android studio and verify you see two entries.

Rebuild the server and client and verify you can register and send messages.

References

https://github.com/lucaconte/BeatTheMeddler
http://debabhishek.com/writes/Installing-and-Running-TextSecure-Signal-Server-on-Windows/
https://community.signalusers.org/