Documentation for the HSP project “LogAnalyzer”

By Philipp Rieger, Andreas Kalo Fogados, Michael Dorr

Date: WS 15/16

Examiner: Prof. Dr. Skornia

# 1 Environment Setup

## 1.1 Log Production via RADIUS Server and Syslog

## 1.2 Database Setup

The setup of the Mongo database is divided into two major steps:

* Setting up the database itself
* Setting up the Mongo C++ driver for Linux so that the “.cpp”-files created later in the programming part of the project can be compiled properly.

### 1.2.1 Set up the Mongo database

Following the steps below in the given order will set up the required Mongo database. It will start running after the setup is completed and start up with the operating system going forward. The provided steps are meant to work on Ubuntu 14.04, it’s not guaranteed to work on other versions or other operating systems.

1. Open the Terminal
2. Execute the following statements from the Terminal one after another:
   * sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927
   * echo "deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list
   * sudo apt-get update
   * sudo apt-get install -y mongodb-org
3. MongoDB is now installed and running in the background. By default, the database is running on port 27017. The connection string to a MongoDB running on the local machine is mongodb://localhost:27017.

### 1.2.2 Set up the Mongo C++ Driver

To be able to compile code using the Mongo features, the Boost library and the Mongo C++ driver version 1.1.0 are needed. The following steps will go through the setup of both of these. After finishing this part, it is possible to compile and execute code taking use of the Mongo features.

1. Open the Terminal
2. Install the prerequired libraries:
   * sudo apt-get install build-essential
   * sudo apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev scons git
3. Set up G++4.9:
   * sudo add-apt-repository ppa:ubuntu-toolchain-r/test
   * sudo apt-get update
   * sudo apt-get install g++-4.9
4. Set up Python:
   * cd Downloads/
   * wget <http://python.org/ftp/python/2.7.2/Python-2.7.2.tgz>
   * tar -xvf Python-2.7.2.tgz && cd Python-2.7.2/
   * ./configure
   * make
   * sudo make altinstall
5. Set up the Boost library:
   * sudo apt-get install libboost-all-dev
6. Set up the Mongo C++ driver:

* cd $HOME/Downloads
* Download the source code from <https://github.com/mongodb/mongo-cxx-driver/archive/legacy-1.1.0.tar.gz>
* tar –xvf mongo-cxx-driver-legacy-1.1.0.tar.gz && cd mongo-cxx-driver-legacy-1.1.0/
* sudo scons install --prefix=/usr/local/ --c++11=on

1. After finishing the driver setup the following command compiles an „example.cpp“ making use of the Mongo features:

g++-4.9 example.cpp -std=c++11 -o example -pthread -lmongoclient -lboost\_thread -lboost\_filesystem -lboost\_program\_options -lboost\_system -lboost\_regex

Following these two guides will set up everthing required to compile and execute the code provided in this project. Additionally, they provide the possibility to develop other application making use of Mongo features in C++ on Ubuntu 14.04.

## 1.3 Project Structure

The hierarchy of the project is as following:

* src: this folder contains all source and header files
* files: the directory lists different files such as log files or test files that were made use of during the development
* bin: folder with the final binary file
* Makefile: file that builds the source files and stores the binary in the bin folder when the “make” command is issued

The development process of the project was performed on a Linux virtual machine, more specifically Ubuntu 14.04, that was running the mongo db service. Furthermore, Git was used as a version control system to simplify the distributed work.

# 2 Implementation Details

The application reads content from log files and stores them into the database. The analyzer starts by reading those log contents from the mongo database and applying a regular expression in order to build login record objects.

As a further step, the application filters information from these records to build user objects that are, in turn, stored in the repository. Thereafter, the analyzer goes ahead with determining the handy mac address for each user based on the frequency of its appearance in the user’s login records.

Eventually, the standard deviation of all considered entries is calculated and further used to check if any anomalies occurred.