PROGRAMMING LANGUAGE RESEARCH GROUP
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF CALIFORNIA, IRVINE

FIDELIUS DOCUMENTATION

Table of Contents

Getting Started

Please first read the **Fidelius** poster and technical report to understand the big picture of the system.

Code Repository: Master

- The **Master** branch contains code that can be run on a Linux client, e.g., desktop PC, Raspberry Pi, etc.

- All the code for **Fidelius** is located in the *iotcloud* git repo.
    - *git clone git://plrg.eecs.uci.edu/iotcloud.git*

- Current version of **Fidelius** located in:
    - *iotcloud -> version2 -> src*

Folder	**Content**
iotcloud/version2/src/server	Server code
iotcloud/version2/src/java/iotcloud	**Fidelius** client code in Java
iotcloud/version2/src/java/light_fan_benchmark	Light fan demo running on computer
iotcloud/version2/src/java/light_fan_embed_benchmark	Light fan demo running on Raspberry Pi's
iotcloud/version2/src/java/Control	Light fan demo Android app
iotcloud/version2/src/C	**Fidelius** client code in C++
iotcloud/version2/src/server_malicious_ignore_seq	Simulated Server attack (server code)
iotcloud/version2/src/server_malicious_override_do_rej	Simulated Server attack (server code)
iotcloud/version2/src/server_malicious_switch_slot	Simulated Server attack (server code)

Code Repository: Core

- The **Core** branch contains code that can be run on an Arduino device, e.g., Particle Photon. The Java version is the same as the **Master** branch.

- All the code for **Fidelius** is located in the *iotcloud* git repo.
    - *git clone git://plrg.eecs.uci.edu/iotcloud.git*
    - *git checkout -t remotes/origin/core*

- Current version of **Fidelius** located in:
    - *iotcloud -> version2 -> src*

Folder	**Content**
iotcloud/version2/src/server	Server code
iotcloud/version2/src/server_malicious_ignore_seq	Simulated Server attack (server code)
iotcloud/version2/src/server_malicious_override_do_rej	Simulated Server attack (server code)
iotcloud/version2/src/server_malicious_switch_slot	Simulated Server attack (server code)
iotcloud/version2/src/C	**Fidelius** client code in C++
iotcloud/version2/src/others/Arduino_DHT	Library for DHT devices from Arduino
iotcloud/version2/src/others/PietteTech_DHT	Library for DHT devices from PietteTech
iotcloud/version2/src/others/functions	Information leakage experiment
iotcloud/version2/src/others/ino	INO files for testing
iotcloud/version2/src/others/ino/IR-Sensor	INO files and Makefile for IR sensor
iotcloud/version2/src/others/ino/Magnetic-Sensor	INO files and Makefile for magnetic sensor
iotcloud/version2/src/others/ino/Temp-Sensor	INO files and Makefile for temperature sensor
iotcloud/version2/src/others/RPi	LiFX controller for Particle Cloud
iotcloud/version2/src/others/setup	Picture of sensor wiring
iotcloud/version2/src/Control-2bulbs.zip	Android app to control 2 light bulbs

Server Setup

To compile this code do the following:

1. Do **ssh** to your cloud server.

2. Clone the git repo.

3. Run the following on command prompt.
    - *$cd iotcloud/version2/src/server*
    - *$make*

Once the code is compiled it must be placed in the correct location:

1. **IMPORTANT: If this is the first time you set up the Fidelius cloud server, please follow the instructions in the *iotcloud/version2/src/server/README.txt*.**

2. If this is just to refresh the setup, do **ssh** to your cloud server.

3. Do **cd** to the location where the code was compiled.

4. Run the following on command prompt.
    - *$sudo rm /usr/lib/cgi-bin/iotcloud.fcgi*
    - *$sudo cp iotcloud.fcgi /usr/lib/cgi-bin/iotcloud.fcgi*
    - *$sudo service apache2 restart*

The server is now ready. If there are any issues, look into *Server.txt* in *iotcloud/version2/src/server* for help.

Linux Clients

Java Clients

Installation

This code runs on a device, e.g., a Raspberry Pi or even a desktop computer that has Java (this was tested on Java 8).

To compile this code do the following.

1. Clone git repo.
    - *git clone git://plrg.eecs.uci.edu/iotcloud.git*

2. Run the following commands.
    - *$cd iotcloud/version2/src/java/iotcloud*
    - *$mkdir bin*
    - *$make*

The code is now compiled.

Running a Basic Test

Before running any tests, please make sure that the Table instantiation URL in the classes are updated. Currently we use *dc-6.calit2.uci.edu* as our cloud server and *test.iotcloud* as our repository folder in the cloud. Thus, you need to update the following URL with the specific URL based on your server configuration (see Server Setup).[1]

"http://dc-6.calit2.uci.edu/test.iotcloud/"

Within the implementation of **Fidelius** there is a test file called:

iotcloud/version2/src/java/iotcloud/Test.java

To run this test:

1. Run this command on server before running a new test.
    - *$sudo rm -rf /iotcloud/test.iotcloud/**

[1] Please note that command lines listed here have to be adjusted to your cloud server configuration as well.

2. Run these commands on client whenever you want to run a new test.
    - *$cd iotcloud/version2/src/java/iotcloud/bin*
    - *$java -cp .:./iotcloud/* iotcloud.Test <test number>*

The test numbers range from 2 to 14. Look at

iotcloud/version2/src/java/iotcloud/Test.java

for what the test does.

Running the Light Fan Test

Perform the following to run this test. You have to prepare the necessary hardware, i.e., Raspberry Pi devices, LiFX bulbs, and WeMo switches.

1. Run this command on server before running this test.
    - *$sudo rm -rf /iotcloud/test.iotcloud/**

2. Run these commands on client.
    - *$./iotcloud/version2/src/java/light_fan_embed_benchmark/build.bash*
    - *$./iotcloud/version2/src/java/light_fan_embed_benchmark/runSetup.bash*

    Open the Android app Control in Android Studios
    - In *iotcloud/version2/src/Control/app/src/main/java/com/example/ali/control* change the IP address on line 320 to the IP address of the raspberry pi that will be running the light bulb controller.
    - Compile the application. You might see errors depending on which Android Studios. Please make the necessary adjustments to proceed.
    - Push and run the application on an Android device.
    - Shut down the application for now.

3. Perform these on each Raspberry Pi:.
    - Clone the git repo: *git clone git://plrg.eecs.uci.edu/iotcloud.git*
    - Compile the IoTCloud Client code (instructions above)
    - Run the following on command prompt.
        - *$cd iotcloud/version2/src/java/light_fan_embed_benchmark*
        - *$./build.bash*
    - Perform this on one Raspberry Pi (the one that is to control the LiFX bulbs):
        - $./run1.bash
    - Perform this on a different Raspberry Pi (the one that is to control the WeMo switches):
        - $./run2.bash

4. Now relaunch the Android app.

C++ Clients

Installation

This code runs on a device, e.g., a Raspberry Pi or even a desktop computer that has C++ (this was tested on Java 8).

To compile this code do the following.

1. Clone git repo.
    - *git clone git://plrg.eecs.uci.edu/iotcloud.git*

2. Run the following commands.
    - *$cd iotcloud/version2/src/C*
    - *$mkdir bin*
    - *$make*

3. Install the necessary C++ libraries.
   If it complains: *SecureRandom.cpp:3:24: fatal error: bsd/stdlib.h: No such file or directory*, then we need to install the standard *libbsd-dev* library. Run the following command.
    - *sudo apt-get install libbsd-dev*

The code is now compiled.

Running a Basic Test

Before running any tests, please make sure that the Table instantiation URL in the classes are updated. Currently we use *dc-6.calit2.uci.edu* as our cloud server and *test.iotcloud* as our repository folder in the cloud. Thus, you need to update the following URL with the specific URL based on your server configuration (see Server Setup).[2]

"http://dc-6.calit2.uci.edu/test.iotcloud/"

To run this test:

1. Run the following commands on server before running a new test.
    - *$sudo rm -rf /iotcloud/test.iotcloud/**

2. Run these commands on client to initialize the cloud server.

[2] Please note that command lines listed here have to be adjusted to your cloud server configuration as well.

    - *$./bin/init*
3. Run these commands on client to perform updates to the cloud.
    - *$./bin/update*
4. Run these commands on client to read from the cloud server.
    - *$./bin/read*

Look at the following files

iotcloud/version2/src/C/Init.C
iotcloud/version2/src/C/Update.C
iotcloud/version2/src/C/Read.C

for what the test does.

Arduino Clients

C++ Clients

Installation

This code runs on an Arduino class of device. In our experiment, we use the Particle Photon device.

To compile this code do the following.

1. Clone git repo and checkout the *remotes/origin/core* branch.
    - *git clone git://plrg.eecs.uci.edu/iotcloud.git*
    - *git checkout -t remotes/origin/core*

2. Install the Particle CLI toolchain on your system.

3. Run the following commands.
    - *$cd iotcloud/version2/src/C*
    - *$make*

The code is now compiled and we have firmware binary file, i.e., *photon_firmware_XXXXXXXXXXXX.bin* in the same folder.

Running a Basic Test

Before running any tests, please make sure that the Table instantiation URL in the classes are updated. Currently we use *dc-6.calit2.uci.edu* as our cloud server and *test.iotcloud* as our repository folder in the cloud. Thus, you need to update the following URL with the specific URL based on your server configuration (see Server Setup).[3]

"http://dc-6.calit2.uci.edu/test.iotcloud/"

To run this test:

1. Run this command on server before running a new test.
    - *$sudo rm -rf /iotcloud/test.iotcloud/**

[3] Please note that command lines listed here have to be adjusted to your cloud server configuration as well.

2. Within the implementation of **Fidelius** there is a test file called: *iotcloud/version2/src/others/ino/Test.ino*. Copy this file into *iotcloud/version2/src/C*.

3. Compile the code by running the following command.
    - *$cd iotcloud/version2/src/C*
    - *$make*

4. Run the following command to flash the binary (.bin) file into a Particle Photon.
    - *$particle flash <photon-device-name> photon_firmware*.bin*

    Please consult Particle Photon documentation for further detail on how to do this. The current *Makefile* has a list of *particle flash* commands that we used to flash the firmware we used in our experiment. You can adjust these to your need.

5. Activate/reset the Particle Photon device to let the code run.

6. We can use the reader code (*Read.C*) to verify the data stored in the cloud server (see Running a Basic Test).

We can replace the INO (*Test.ino*) file that we compile with **Fidelius** with one of the INO files in *iotcloud/version2/src/others/* (see Code Repository: Core).

Test bed on Particle Cloud

We first ran our test bed on Particle Cloud. This test bed consists of 16 devices:

1. 8 Particle Photons with temperature and humidity sensors,

2. 4 Particle Photons with magnetic door sensors,

3. 3 Particle Photons with IR-based motion sensors, and

4. a Raspberry Pi 1 that controls 2 LiFX smart light bulbs.

The related files stored in the following folders.

Folder	**Content**
iotcloud/version2/src/others/Arduino_DHT	Library for DHT devices from Arduino
iotcloud/version2/src/others/PietteTech_DHT	Library for DHT devices from PietteTech
iotcloud/version2/src/others/functions	Information leakage experiment
iotcloud/version2/src/others/ino	INO files for testing

iotcloud/version2/src/others/ino/IR-Sensor	INO files and Makefile for IR sensor
iotcloud/version2/src/others/ino/Magnetic-Sensor	INO files and Makefile for magnetic sensor
iotcloud/version2/src/others/ino/Temp-Sensor	INO files and Makefile for temperature sensor
iotcloud/version2/src/others/RPi	LiFX controller for Particle Cloud
iotcloud/version2/src/others/setup	Picture of sensor wiring
iotcloud/version2/src/Control-2bulbs.zip	Android app to control 2 light bulbs

To prepare the benchmark please perform the following steps:

1. Run this command on server before running a new test.
    a. *$sudo rm -rf /iotcloud/test.iotcloud/**

2. Wire the sensor according to the pictures in *iotcloud/version2/src/others/setup* (please see the INO file to get exact pin information for proper wiring as well).

3. Use the library for DHT devices either from Arduino or PietteTech; copy the *.cpp* and *.h* files into *iotcloud/version2/src/C* (see Running a Basic Test).

4. Copy the INO file (and other related files such as the Makefile) into the same folder. Please adjust the Makefile to your specific configuration of Particle Photon device.
   Please take note that there are two versions of INO files for each sensor; use the one with the word "Particle" for this experiment.

5. Compile the INO file together with **Fidelius** to get the binary file and flash it onto a Particle Photon device.

6. Please repeat steps 2 and 3 for all of the 3 sensors.

7. Compile the controller code for LiFX light bulbs in *iotcloud/version2/src/others/RPi*. Flash the binary file onto a Raspberry Pi (please see the most current Particle documentation to do this).

8. Execute/reset the Particle Photon devices that control the sensors and let the light bulb controller run on the Raspberry Pi.

9. The Particle Photon devices are going to update the cloud server periodically to report sensor readings.

10. The light bulb controller can be given inputs through the Particle Cloud login page or Particle phone app (please see the most most current Particle documentation to figure out more about function inputs).

11. The summary of sensor readings can also be monitored through the Particle Cloud login page.

Test bed on Fidelius Cloud

We then also ran our test bed on **Fidelius** Cloud. This test bed also consists of 16 devices. The details on types of device and folders have been presented in Test bed on Particle Cloud.

To prepare the benchmark please perform the following steps:

1. Run this command on server before running a new test.
    - *$sudo rm -rf /iotcloud/test.iotcloud/**

2. Wire the sensor according to the pictures in *iotcloud/version2/src/others/setup* (please see the INO file to get exact pin information for proper wiring as well).

3. Use the library for DHT devices either from Arduino or PietteTech; copy the *.cpp* and *.h* files into *iotcloud/version2/src/C* (see Running a Basic Test).

4. Copy the INO file (and other related files such as the Makefile) into the same folder. Please adjust the Makefile to your specific configuration of Particle Photon device.
   Please take note that there are two versions of INO files for each sensor; use the one with the word “Fidelius” for this experiment.

5. Compile the INO file together with **Fidelius** to get the binary file and flash it onto a Particle Photon device.

6. Please repeat steps 2 and 3 for all of the 3 sensors.

7. We use the same **Fidelius** light bulb controller that we use in Running the Light Fan Test. The Android app has also been adjusted to control 2 light bulbs. Please unzip *iotcloud/version2/src/Control-2bulbs.zip*, and use Android Studios to compile and run it on an Android phone.

8. Execute/reset the Particle Photon devices that control the sensors and let the light bulb controller run on the Raspberry Pi.

9. The Particle Photon devices are going to update the cloud server periodically to report sensor readings.

10. The light bulb controller can be given inputs through the phone app mentioned in step 6 above.

Energy Measurements

The energy measurement experiment was also done on the same test bed with Particle Photon devices that control the 3 types of sensors. The experiment was performed also both on Particle and **Fidelius** clouds.

Information Leakage

The information leakage experiment was done using the files stored in *iotcloud/version2/src/others/functions*. In this experiment, we varied the lengths of different messages that we sent to the Particle Cloud and observed the network traffic trace captured using *tcpdump* on the router. We used *Wireshark* to scrutinize the network trace.

PyORAM Experiment

We also conducted an experiment to compare **Fidelius** and an ORAM implementation called PyORAM. This is a Python implementation of the PathORAM. Please see the code in *iotcloud/PyORAM*. For this experiment, we used *iotcloud/PyORAM/examples/ali.py*. We varied the block size and block count following the numbers we used for **Fidelius**.