```
#!/usr/bin/env python
import os
import subprocess
import datetime
import time
import json
import urllib2
import sqlite3
import sys
import RPi.GPIO as GPIO
import socket
from datetime import datetime
import iotsec_settings
from seciot import SecIOT
import json
import time
import Adafruit_Nokia_LCD as LCD
import Adafruit_GPIO.SPI as SPI
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont
def internet(host="8.8.8.8", port=53, timeout=3):
    """
    Host: 8.8.8.8 (google-public-dns-a.google.com)
    OpenPort: 53/tcp
    Service: domain (DNS/TCP)
    """
    try:
        socket.setdefaulttimeout(timeout)
        socket.socket(socket.AF_INET, socket.SOCK_STREAM).connect((host, port))
        return True
    except Exception as ex:
        print ex.message
        return False
def die():
#   GPIO.output(26,1)
#   time.sleep(1)
    GPIO.cleanup()

def do_reset():
    #Reset to factory!!
    die()


def start_gpio():
    GPIO.setmode(GPIO.BCM)

    #Input Pins
    TAMPER = 4
    RESET = 16
    BUTTON = 6
```

```
    #Output Pins
    BUZ = 26
    BUTTON_LED = 5
    LCD_BACKLIGHT = 20
    NET_STATUS = BUTTON_LED
    ERR_LED = 21

    #Display pins
    DC = 23
    RST = 24
    SPI_PORT = 0
    SPI_DEVICE = 0

    #Setup pins

    for pin in [TAMPER, RESET, BUTTON]:
        GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
    for pin in [BUTTON_LED, BUZ, LCD_BACKLIGHT, NET_STATUS, ERR_LED]:
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin,0)

    # Setup LCD Display
    disp = LCD.PCD8544(DC, RST, spi=SPI.SpiDev(SPI_PORT, SPI_DEVICE,
      max_speed_hz=4000000))
    disp.begin(contrast=60)
    disp.clear()
    disp.display()

    infofile = "info.json"
    guid = None
    if os.path.isfile(infofile):
        info=open("info.json", 'r')
        guid=json.loads(info.read())["guid"]


    try:
        while True:
            #Internet monitoring LED
            if(internet()):
                GPIO.output(NET_STATUS, 1)
            else:
                GPIO.output(NET_STATUS,0)

            #Tamper Detection
            if GPIO.input(TAMPER) == 0:
                GPIO.output(BUZ,1)
                subprocess.call(['/opt/seciot/tamper.sh'])
                GPIO.output(ERR_LED, 1)
            else:
                GPIO.output(BUZ,0)

            # LCD output
```

```
            if GPIO.input(BUTTON) == 0:
                if guid is None:
                    guid = SecIOT.new_guid(url = "osrsrv.aakportfolio.com")
                disp.clear()
                disp.display()
                # Create blank image for drawing.
                # Make sure to create image with mode '1' for 1-bit color.
                image = Image.new('1', (LCD.LCDWIDTH, LCD.LCDHEIGHT))
                # Get drawing object to draw on image.
                draw = ImageDraw.Draw(image)
                # Draw a white filled box to clear the image.
                draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT), outline=255,
                  fill=255)
                # Load default font.
                font = ImageFont.load_default()
                # Write some text.
                draw.text((0,0), guid[:14], font=font)
                draw.text((0,10), guid[14:28], font=font)
                draw.text((0,20), guid[28:], font=font)

                print guid

                # Display image.
                disp.image(image)
                disp.display()
                GPIO.output(LCD_BACKLIGHT, 1)
                GPIO.output(BUTTON_LED, 1)
                time.sleep(10)
            else:
#           GPIO.output(BUTTON_LED, 0)
                disp.clear()
                disp.display()
                GPIO.output(LCD_BACKLIGHT, 0)


            if GPIO.input(RESET) == 1:
                print "reset"
                steptime = datetime.now().microsecond + 5000
                while GPIO.input(RESET) and (steptime < datetime.now().microsecond):
                    time.sleep(0.05)
                if GPIO.input(RESET):
                    do_reset()
                time.sleep(0.05)
    except KeyboardInterrupt:
        die()
if __name__ == "__main__":
    start_gpio()
```

```
IOT_MOBILE_DEVICE   = 0
IOT_HOME_NODE       = 1
IOT_ZWAVE_NAME      = "ZWAVE"
IOT_ZWAVE_LOCATION  = "NOWHERE"
#cafile="/opt/seciot/osrsrv.aakportfolio.com.crt"
#cafile="./osrsrv.aakportfolio.com.crt"
#cafile="./aakportfolio.com.crt"
#cafile="./spr.pem"
#cafile="/etc/ssl/certs/ca-certificates.crt"
#cafile="./ca-bundle-client.crt"
cafile=True
protocol='https'
server_host = "osrsrv.aakportfolio.com"
```

```
#!/usr/bin/env python
import datetime
import time
import json
from iotsec_settings import *
import sqlite3
import sys
import requests
class SecIOT():
    def __init__(self, auth, home_or_mobile, guid = None, statefile="state.db"):
        self.server_host = server_host
        self.home_or_mobile = home_or_mobile
        self.auth = auth
        self.state_file = statefile
        self.protocol = protocol

        if guid is None:
            #Restore node stuff
            info=open("info.json", 'r')
                    guid=json.loads(info.read())["guid"];
        try:
            self.getvalues()
        except:
            self.dbconn = sqlite3.connect(self.state_file)
            c = self.dbconn.cursor()
            c.execute("create table valuetables(valname text,valvalue text,valtime
              text,modified integer);")
            c.execute("CREATE UNIQUE INDEX I1 ON valuetables(valname);")
            self.dbconn.commit()

        service_path = self.get_server_path(home_or_mobile)
        if service_path == None:
            raise "fail"
        else:
            self.service_path = service_path

        self.guid = guid

    @staticmethod
    def geturl(action):
            return '%s://%s/%s/' % (protocol, server_host, action)


    @staticmethod
    def new_guid(url):
            return '%s://%s/%s/' % ('https', self.server_host, action)

            url = SecIOT.geturl('guid/new')
        print url
        r = requests.get(url, verify=cafile)
        self.guid = r.text
        print self.guid
```

```
        return self.guid

    def get_server_path(self, home_or_mobile):
        if home_or_mobile == IOT_MOBILE_DEVICE:
            service_path = "mobile"
        elif home_or_mobile == IOT_HOME_NODE:
            service_path = "home"
        else:
            service_path = None
        return service_path

    def poll_state(self, poll_type):
        url = SecIOT.geturl('poll')
        headers = {'Content-Type' : 'application/json'}
        payload = {'guid':self.guid, 'home_or_mobile': poll_type}

        print url

        r = requests.post(url, data=json.dumps(payload), headers=headers,
          verify=cafile)

        print r.text

        for v in r.json()['state']:
            t = self.getvalueall(v[0])
            print t,v
            beforetime = time.mktime(datetime.datetime.strptime(t[2], "%Y-%m-%d %H:%M:
              %S").timetuple())
            currenttime = time.mktime(datetime.datetime.strptime(v[2], "%Y-%m-%d %H:%
              M:%S").timetuple())
            if beforetime < currenttime:
                self.setvalue(v[0],v[1])

        return r.json()



    def push_state(self):
        self.dbconn = sqlite3.connect(self.state_file)
        c = self.dbconn.cursor()
        c.execute("select * from valuetables where modified=1")
        rc = c.rowcount
        print "rowcount %d" % rc

        c.execute("update valuetables set modified=0")
        self.dbconn.commit()

        url = SecIOT.geturl('push')
        headers = {'Content-Type' : 'application/json'}
        payload = {'guid':self.guid, 'home_or_mobile': self.home_or_mobile,
          'state':self.getvalues()}
```

```
        print url

        r = requests.post(url, data=json.dumps(payload), headers=headers,
          verify=cafile)
        print r.text
        self.dbconn.close()
        return r.json()


    def getvalues(self):
        self.dbconn = sqlite3.connect(self.state_file)
        c = self.dbconn.cursor()
        c.execute("select * from valuetables")
        t = c.fetchall()
        self.dbconn.close()
        return t

    def setvalue(self, name, value):
        self.dbconn = sqlite3.connect(self.state_file)
        c = self.dbconn.cursor()
        c.execute("insert or replace into valuetables
          (valname,valvalue,valtime,modified) values (?,?,datetime
          ('now','localtime'),1)",[name,value])
        self.dbconn.commit()
        self.dbconn.close()

    def getvalue(self, name):
        self.dbconn = sqlite3.connect(self.state_file)
        c = self.dbconn.cursor()
        c.execute("select * from valuetables where valname=?",[name])
        t =  c.fetchone()[1]
        self.dbconn.close()
        return t

    def getvalueall(self, name):
        self.dbconn = sqlite3.connect(self.state_file)
        c = self.dbconn.cursor()
        c.execute("select * from valuetables where valname=?",[name])
        t =  c.fetchone()
        self.dbconn.close()
        return t

    def clearvalues(self):
        self.dbconn = sqlite3.connect(self.state_file)
        c = self.dbconn.cursor()
        c.execute("delete from valuetables")
        self.dbconn.commit()
        self.dbconn.close()
```

```
if __name__ == "__main__":

#   foobar = SecIOT("osrsrv.aakportfolio.com","sfasdasdfa","enc_key" , IOT_HOME_NODE);
    foobar = SecIOT("enc_key" , int(sys.argv[2]), guid="sfasdasdfa")
#   foobar = SecIOT("127.0.0.1:5000","sfasdasdfa","enc_key" , IOT_HOME_NODE);
#   foobar = SecIOT("127.0.0.1:5000","sfasdasdfa","enc_key" , IOT_MOBILE_DEVICE);

#   status = int(sys.argv[1])
#   print status

    push = True

    if int(sys.argv[1]) == 1:
        status = True
    elif int(sys.argv[1]) == 0:
        status = False
    elif int(sys.argv[1]) == -1:
        push = False

    if push:
        foobar.setvalue("switch1",status)
        foobar.setvalue("switch2",status)
        foobar.setvalue("switch3",status)
        foobar.setvalue("switch4",status)
        foobar.setvalue("switch5",status)
        print foobar.push_state()

    print "home", foobar.poll_state(IOT_HOME_NODE)

    print "mobile", foobar.poll_state(IOT_MOBILE_DEVICE)

    print foobar.getvalues()
```

```
#!/usr/bin/env python
import openzwave
import unicodedata
from openzwave.node import ZWaveNode
from openzwave.value import ZWaveValue
from openzwave.scene import ZWaveScene
from openzwave.controller import ZWaveController
from openzwave.network import ZWaveNetwork
from openzwave.option import ZWaveOption
from louie import dispatcher, All
import sys
import time
from device import IOTDeviceController
import iotsec_settings
from seciot import *
from daemon import Daemon
import json

class ZWDeviceController(IOTDeviceController):
    def louie_network_started(network):
        print("Hello from network : I'm started : homeid %0.8x - %d nodes were found."
           % \
            (network.home_id, network.nodes_count))

    def louie_network_failed(network):
        print("Hello from network : can't load :(.")

    def louie_network_ready(network):
        print("Hello from network : I'm ready : %d nodes were found." %
          network.nodes_count)
        print("Hello from network : my controller is : %s" % network.controller)
        dispatcher.connect(louie_node_update, ZWaveNetwork.SIGNAL_NODE)
        dispatcher.connect(louie_value_update, ZWaveNetwork.SIGNAL_VALUE)

    def louie_node_update(network, node):
        print('Hello from node : %s.' % node)

    def louie_value_update(network, node, value):
        print('Hello from value : %s.' % value)


    def __init__(self, name, location, nodefilename="/opt/seciot/nodenames.json"):
        IOTDeviceController.__init__(self, name)

        #Restore node stuff
        nodefile=open(nodefilename, 'r')
        nodejson=nodefile.read()
        self.node_dict = json.loads(nodejson)



        #Init options
```

```
        device="/dev/ttyACM0"
        sniff=300.0
        options = ZWaveOption(device, \
         config_path="/opt/openzwave/config", \
         user_path=".", cmd_line="")
        options.set_logging(False)
        options.set_console_output(False)
        options.lock()
        #Create a network object
        self.network = ZWaveNetwork(options, autostart=False)
        self.network.set_poll_interval(10,True)
        #We connect to the louie dispatcher
        dispatcher.connect(self.louie_network_started,
          ZWaveNetwork.SIGNAL_NETWORK_STARTED)
        dispatcher.connect(self.louie_network_failed,
          ZWaveNetwork.SIGNAL_NETWORK_FAILED)
        self.network.start()
        #We wait for the network.
        print "***** Waiting for network to become ready : "
        for i in range(0,300):
            if self.network.state>=self.network.STATE_READY:
                print "***** Network is ready"
                break
            else:
                sys.stdout.write(".")
                sys.stdout.flush()
            time.sleep(1.0)
        #We update the name of the controller
        self.network.controller.node.name = name
        self.network.controller.node.location = location


    def export(self):
        #list of devices
        #each device has zwavename, friendly name, device type, statepool,
          currentstates (timestamped)
        #return self.network.nodes
        return self.node_dict.keys()

    def readState(self, node):
        print "read",self.node_dict
        print "read",node
        if(type(node) in [str,unicode]):
            nodenum = self.node_dict[node]
        else:
            print type(node)
            nodenum = node
        mynode = self.network.nodes[nodenum]
        state = None
        for switch in mynode.get_switches():
            state =  mynode.get_switch_state(switch)
            print node
```

```
            print state
        return state;


    def setState(self, node, state):
        print "set",self.node_dict
        print "set",node
        if(type(node) in [str,unicode]):
            nodenum = self.node_dict[node]
        else:
            nodenum = node
        if state in [1, "1", u'1', True]:
            boolstate = True
        elif state in [0, "0", u'0', False]:
            boolstate = False
        else:
            print state
            print type(state)
            raise error("bad")
        print state
        for switch in self.network.nodes[nodenum].get_switches():
            self.network.nodes[nodenum].set_switch(switch,boolstate)


def zwservice():
    #Create controller and network service
    zwave = ZWDeviceController(iotsec_settings.IOT_ZWAVE_NAME,
      iotsec_settings.IOT_ZWAVE_LOCATION)
    zwavepoll = SecIOT("enc_key", IOT_HOME_NODE)
    i = True
    while True:
        i = not(i)
        #for node in zwave.export():
        #   if not zwave.readState(node) == zwavepoll.getvalue(node):
        #       zwavepoll.setvalue(node, zwave.readState(node))

        if i == False:
            #print zwavepoll.poll_state(IOT_HOME_NODE)
            print zwavepoll.poll_state(IOT_MOBILE_DEVICE)
            for node in zwave.export():
                                anode =node #unicodedata.normalize('NFKD',
                    node).encode('ascii','ignore')
                                zwave.setState(anode, zwavepoll.getvalue(anode))
            time.sleep(0.5)
        else:
                        for node in zwave.export():
                anode = node#unicodedata.normalize('NFKD', node).encode
                  ('ascii','ignore')
                                if not zwave.readState(anode) == zwavepoll.getvalue
                    (anode):
                    state = zwave.readState(anode)
                    print anode,":",state
```

```
                    zwavepoll.setvalue(anode, state)
            zwavepoll.push_state()
            time.sleep(.5)
        time.sleep(2)
if __name__ == "__main__":
    zwservice()
```

```
from iotsec_settings import *
from flask import Flask,request,json
import os
import hashlib
import sqlite3

SQL_DB_PATH = '/var/www/seciotcloud/seciotcloud/iotsec.db'
#SQL_DB_PATH = '/Users/phar/andrew/iotsec.db'


#import pycrypto


app = Flask(__name__)



#
#
#  /service/{GUID}
#

def check_db():
    try:
        conn = sqlite3.connect(SQL_DB_PATH)
        c = conn.cursor()
        c.execute("select * from iotdata");
    except:
        conn = sqlite3.connect(SQL_DB_PATH)
        c = conn.cursor()
        c.execute("create table iotdata(base_guid text,client_or_server
          integer,msg_time text,state text, node_map text,UNIQUE
          (base_guid,client_or_server)  ON CONFLICT REPLACE);")
        conn.commit()

@app.route("/poll/", methods=['GET', 'POST'])

def poll(override = -1, push = 0):
    HOME_OR_MOBILE = request.json['home_or_mobile']
#   print type(HOME_OR_MOBILE)
    if (push == 0):
        print HOME_OR_MOBILE
#   if(override == -1):
        if(0 == 1):
            print "not redirected"
    else:
#       print "redirected"
        HOME_OR_MOBILE = str(override)
    check_db()
    conn = sqlite3.connect(SQL_DB_PATH)
    c = conn.cursor()
```

```
#   print request.get_data()
    c.execute("select state from iotdata where base_guid=? and client_or_server=?;" ,
      (request.json['guid'], str(HOME_OR_MOBILE)))
    data = c.fetchone()
#   print "data:"
#   print type(data)
#   print data

    if(not(data is None)):
        return data
    elif (override == -1):
        if (request.json['home_or_mobile'] == str(IOT_HOME_NODE)):
            return poll(override = str(IOT_MOBILE_DEVICE))
        elif (request.json['home_or_mobile'] == str(IOT_MOBILE_DEVICE)):
            return poll(override = str(IOT_HOME_NODE))
    else:
        print "Data is NONE!"
        data = "{}"
    return data

@app.route("/push/", methods=['GET', 'POST'])

def push(redir = 0):
    check_db()
    conn = sqlite3.connect(SQL_DB_PATH)
    c = conn.cursor()
    print "push"
    print request.json['home_or_mobile']

    if 'state' in request.json:
        c.execute("insert or replace into iotdata (base_guid,
          client_or_server,msg_time, state) values (?,?,datetime
          ('now','localtime'),?);" , [request.json['guid'], request.json
          ['home_or_mobile'], request.get_data()])
        conn.commit()
    return poll(push = 1, override = int(request.json['home_or_mobile']))



@app.route("/nodes/get", methods=['POST'])
def get_nodes():
    guid = request.args.get('guid')
    check_db()
    conn = sqlite3.connect(SQL_DB_PATH)
    c = conn.cursor()
    c.execute("select node_map from iotdata where base_guid=?;", (guid,))
    return c.fetchone()



@app.route("/nodes/set", methods=['POST'])
def set_nodes():
```

```
    guid = request.json['guid']
    check_db()
    conn = sqlite3.connect(SQL_DB_PATH)
    c = conn.cursor()
    if request.method == 'POST':
        c.execute("insert or replace into iotdata (base_guid, node_map) values
          (?,?);" , [guid, request.json[node_map]])
    return get_nodes()


@app.route("/")

def main():

    return "Hello, world!!!"


@app.route("/guid/new")
def new_guid():
    check_db()
    conn = sqlite3.connect(SQL_DB_PATH)
    c = conn.cursor()
    while True:
        guid = str(hashlib.sha1(os.urandom(100)).hexdigest())
        c.execute("select state from iotdata where base_guid=?;", (guid,))
        if(c.fetchone() is None):
            c.execute("insert or replace into iotdata (base_guid, client_or_server)
              values (?,?)", (guid,'0'))
            conn.commit()
            return guid



if __name__ == "__main__":

    app.run(host="0.0.0.0",port=80,debug=True)
```

File - C:\DEV\Github\seciot\seciotapp\app\src\main\java\com\aakportfolio\aakatz3\seciotapp\seciot.java

```
 1 package com.aakportfolio.aakatz3.seciotapp;
 2
 3 import android.content.Context;
 4 import android.content.Intent;
 5 import android.os.AsyncTask;
 6 import android.util.Log;
 7
 8 import org.json.JSONArray;
 9 import org.json.JSONException;
10 import org.json.JSONObject;
11
12 import java.text.SimpleDateFormat;
13 import java.util.ArrayList;
14 import java.util.Calendar;
15 import java.util.HashMap;
16
17
18 /**
19  * Created by Andrew on 9/16/2016.
20  */
21 public class seciot {
22     private static final String SERVER_URL = "https://osrsrv
   .aakportfolio.com/";
23     private String guid;
24     private ArrayList<iotnode> nodes;
25     private SecureAPI secureAPI;
26
27     SecIOTPollCompleteListener secIOTPollCompleteListener;
28
29     public void updateGuid(String guid){
30         this.guid = guid;
31     }
32
33     public seciot(String guid, SecureAPI secureAPI,
   SecIOTPollCompleteListener listener){
34         this.guid = guid;
35         nodes = new ArrayList<>(10);
36         for(int i = 0; i < 10; i++){
37             nodes.add(new iotnode((i + 1), null, false));
38         }
39        this.secureAPI = secureAPI;
40         secIOTPollCompleteListener = listener;
```

```
41      }
42
43      public seciot(SecureAPI secureAPI,
    SecIOTPollCompleteListener listener){
44          this.guid = null;
45          nodes = new ArrayList<>(10);
46          for(int i = 0; i < 10; i++){
47              nodes.add(new iotnode((i + 1), null, false));
48          }
49          this.secureAPI = secureAPI;
50          secIOTPollCompleteListener = listener;
51      }
52
53      public ArrayList<iotnode> getNodes(){
54          return nodes;
55      }
56      public void setNodes(ArrayList<iotnode> nodes){
57          this.nodes = nodes;
58          if(secIOTPollCompleteListener != null){
59              secIOTPollCompleteListener.performUpdate();
60          }
61      }
62      public void setState(int node, boolean state){
63          nodes.get(node).setState(state);
64      }
65      public void pushState() throws JSONException {
66          //YYYY-MM-DD HH:MM:SS (24)
67          SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-
    dd HH:mm:ss");
68          String formattedDate = df.format(Calendar.
    getInstance().getTime());
69          ArrayList<JSONArray> switches = new ArrayList<>();
70          for(iotnode n : nodes) {
71            /*  ArrayList<Object> switchObj = new ArrayList
    <>();
72              switchObj.add(n.friendlyName);
73              if(n.state){
74                  switchObj.add("1");
75              } else {
76                  switchObj.add("0");
77              }
78              switchObj.add(formattedDate);
```

```
 79              switchObj.add(0);
 80              switches.add(new JSONArray(switchObj));*/
 81              //switches.add(new JSONArray("[\"" + n.
    friendlyName + "\"," + (n.state ? "\"1\",\"" : "\"0
    \",\"") +  formattedDate + "\",0]"));
 82              switches.add(new JSONArray("['" + n.
    friendlyName + "'," + (n.state ? "'1','" : "'0','") +
    formattedDate + "',1]"));
 83          }
 84
 85          JSONArray arr = new JSONArray(switches);
 86
 87          String url = SERVER_URL + "push/";
 88
 89          JSONObject postParams = new JSONObject();
 90
 91          postParams.put("guid", guid);
 92          postParams.put("home_or_mobile", Integer.toString(
    iotsettings.IOT_MOBILE_DEVICE));
 93          postParams.put("state", arr.toString());
 94
 95          String jsstr = postParams.toString();
 96
 97          jsstr = jsstr.replace("\\\"", "\"");
 98          jsstr = jsstr.replace("\"[", "[");
 99          jsstr = jsstr.replace("]\"", "]");
100
101          postParams = new JSONObject(jsstr);
102
103          Log.d("json", postParams.toString());
104
105          SecIOTHelerParams secIOTHelerParams = new
    SecIOTHelerParams(secureAPI, url, postParams, null);
106
107          new SecIOTHelper().execute(secIOTHelerParams);
108          //return null;
109      }
110
111
112
113      public void pollState(SecIOTPollCompleteListener
    listener) throws Exception{
```

```
114         pollState(listener, iotsettings.IOT_HOME_NODE);
115     }
116
117     public void pollState(SecIOTPollCompleteListener
    listener, int homeOrMobile) throws Exception{
118         String url = SERVER_URL + "poll/";
119
120         JSONObject postParams = new JSONObject();
121
122         postParams.put("guid", guid);
123         postParams.put("home_or_mobile", Integer.toString(
    homeOrMobile));
124         Log.d("json", postParams.toString());
125
126         SecIOTHelerParams secIOTHelerParams = new
    SecIOTHelerParams(secureAPI, url, postParams, listener);
127
128         new SecIOTHelper().execute(secIOTHelerParams);
129     }
130     private class SecIOTHelerParams{
131         SecureAPI secureAPI;
132         JSONObject jso;
133         String url;
134         SecIOTPollCompleteListener listener;
135
136         SecIOTHelerParams(SecureAPI secureAPI, String url,
    JSONObject jso, SecIOTPollCompleteListener listener){
137             this.secureAPI = secureAPI;
138             this.jso = jso;
139             this.url = url;
140             this.listener = listener;
141         }
142     }
143
144     private class SecIOTHelper extends AsyncTask<
    SecIOTHelerParams, JSONObject, JSONObject> {
145         SecIOTHelerParams myParams;
146
147
148         @Override
149         protected JSONObject doInBackground(
    SecIOTHelerParams... params) {
```

```
150             myParams = params[0];
151             try{
152                 //HashMap<String, String> map = new HashMap
    <>();
153                 //map.put("a","b");
154                 //return params[0].secureAPI.HTTPSPOSTJSON(
    params[0].url, map);
155                 return myParams.secureAPI.HTTPSPOSTJSON(
    myParams.url,myParams.jso);
156             }catch(Exception e){
157                 e.printStackTrace();
158                 return null;
159             }
160         }
161 
162         @Override
163         protected void onPostExecute(JSONObject v) {
164             if(v != null) {
165                 Log.d("json", v.toString());
166                 updateState(v);
167             } else {
168                 Log.d("json", "NULL");
169             }
170         }
171     }
172 
173     public void updateState(JSONObject recievedStates){
174         JSONArray array = null;
175         try {
176             array = recievedStates.getJSONArray("state");
177         } catch (JSONException e) {
178             e.printStackTrace();
179         }
180 
181         if(array != null){
182             ArrayList<iotnode> nodes = new ArrayList<>();
183             for (int i = 0; i < array.length(); i++){
184                 try {
185                     Log.d("JSO-sw", array.get(i).toString()
    );
186                     JSONArray node = ((JSONArray) array.get
    (i));
```

```
187                     Log.d("JSO-sw", node.get(1).toString())
    ;
188                     nodes.add(new iotnode(i + 1, node.get(0
    ).toString(), (node.get(1).toString().equals("1"))));
189                 } catch (JSONException e) {
190                     e.printStackTrace();
191                     Log.d("JSO-sw", "err");
192                 }
193             }
194             if(nodes.size() > 0){
195                 setNodes(nodes);
196             }
197         }
198     }
199
200     interface SecIOTPollCompleteListener{
201         public void performUpdate();
202     }
203
204 }
205
```

File - C:\DEV\Github\seciot\seciotapp\app\src\main\java\com\aakportfolio\aakatz3\seciotapp\iotnode.java

```java
package com.aakportfolio.aakatz3.seciotapp;

/**
 * Created by Andrew on 10/6/2016.
 */

class iotnode {
    int index;
    String friendlyName;
    boolean state;

    public iotnode(int index, String friendlyName, boolean state){
        this.state = state;
        this.index = index;
        this.friendlyName = friendlyName;
    }

    boolean getState(){
        return state;
    }
    void setState(boolean state){
        this.state = state;
    }

    public String toString(){
        if(friendlyName == null){
            return "Node " + Integer.toString(index);
        } else {
            return friendlyName;
        }
    }
}

```

```java
package com.aakportfolio.aakatz3.seciotapp;


import android.content.Context;
import android.util.Log;

import org.json.JSONObject;

import java.io.BufferedInputStream;
import java.io.BufferedWriter;
import java.io.ByteArrayOutputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.URL;
import java.net.URLEncoder;
import java.security.KeyStore;
import java.security.cert.Certificate;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.util.Map;
import java.util.Scanner;

import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.TrustManagerFactory;

/**
 * Handles secure server communications
 * SecureAPI singleton class
 * Created by Andrew Katz
 */
public class SecureAPI {

    private static SecureAPI mySecureAPI = null;
    private SSLSocketFactory mySocketFactory = null;

```

```
43
44
45      public static SecureAPI getInstance(Context c){
46          if( mySecureAPI == null){
47              try {
48                  mySecureAPI = new SecureAPI(c.
    getApplicationContext()
49                          .getResources().openRawResource(R.
    raw.cabundle));
50              }catch (Exception e){
51                  Log.d("SecureAPI Exception",e.getMessage());
52              }
53          }
54          return mySecureAPI;
55      }
56      public static SecureAPI getInstance() throws Exception {
57          if( mySecureAPI == null){
58              throw new NullPointerException();
59          }
60          return getInstance(null);
61      }
62      private void createSocketFactory(InputStream is) throws
    Exception {
63          CertificateFactory certificateFactory =
    CertificateFactory.getInstance("X.509");
64          Certificate certificate = certificateFactory.
    generateCertificate(is);
65          Log.d("SecureAPI", "ca=" + ((X509Certificate)
    certificate).getSubjectDN());
66
67
68          KeyStore keyStore = KeyStore.getInstance(KeyStore.
    getDefaultType());
69          keyStore.load(null, null);
70          keyStore.setCertificateEntry("ca", certificate);
71
72          TrustManagerFactory trustManagerFactory =
73                  TrustManagerFactory.getInstance(
    TrustManagerFactory.getDefaultAlgorithm());
74          trustManagerFactory.init(keyStore);
75
76          SSLContext sslContext = SSLContext.getInstance("TLS"
```

```
 76 );
 77         sslContext.init(null, trustManagerFactory.
    getTrustManagers(), null);
 78
 79         mySocketFactory = sslContext.getSocketFactory();
 80     }
 81
 82     private String getPostString(Map<String, String> params
    ) throws Exception {
 83         StringBuilder stringBuilder = new StringBuilder();
 84         boolean first = true;
 85         for(Map.Entry<String, String> p : params.entrySet()
    ) {
 86             if(first)
 87                 first = false;
 88             else
 89                 stringBuilder.append("&");
 90
 91             stringBuilder.append(URLEncoder.encode(p.getKey
    (), "UTF-8"));
 92             stringBuilder.append("=");
 93             stringBuilder.append(URLEncoder.encode(p.
    getValue(), "UTF-8"));
 94         }
 95         return stringBuilder.toString();
 96     }
 97
 98     private String getResponseFromStream(InputStream is) {
 99         Scanner scanner = new java.util.Scanner(is).
    useDelimiter("\\A");
100         return scanner.hasNext() ? scanner.next() : "";
101     }
102
103     private SecureAPI(InputStream inputStream) throws
    Exception {
104         this.createSocketFactory(inputStream);
105     }
106
107
108
109     public JSONObject HTTPSPOSTJSON(String urlString,
    JSONObject json) throws Exception {
```

```
110         URL url = new URL(urlString);
111         HttpsURLConnection httpsURLConnection = (
    HttpsURLConnection) url.openConnection();
112 /*
113         httpsURLConnection.setSSLSocketFactory(
    mySocketFactory);
114         httpsURLConnection.setReadTimeout(10000);
115         httpsURLConnection.setConnectTimeout(15000);
116         httpsURLConnection.setUseCaches(false);
117         httpsURLConnection.setRequestMethod("POST");
118         httpsURLConnection.setRequestProperty("Accept", "
    application/json");
119         httpsURLConnection.setRequestProperty("Content-Type
    ", "application/json");
120        // httpsURLConnection.setRequestProperty("Host", "
    osrsrv.aakportfolio.com");
121         httpsURLConnection.setDoInput(true);
122         httpsURLConnection.setDoOutput(true);
123 */
124         httpsURLConnection.setSSLSocketFactory(
    mySocketFactory);
125         httpsURLConnection.setReadTimeout(10000);
126         httpsURLConnection.setConnectTimeout(15000);
127         httpsURLConnection.setRequestMethod("POST");
128         httpsURLConnection.setRequestProperty("Content-Type
    ", "application/json");
129         httpsURLConnection.setDoInput(true);
130         httpsURLConnection.setDoOutput(true);
131
132         //String request = URLEncoder.encode(json.toString
    (), "UTF-8");
133         String request = json.toString();
134
135         Log.d("request", request);
136
137
138        httpsURLConnection.connect();
139
140         DataOutputStream outputStream = new
    DataOutputStream(httpsURLConnection.getOutputStream());
141        // outputStream.write(json.toString().getBytes("UTF-
    8"));
```

```
142
143         //OutputStreamWriter wr= new OutputStreamWriter(
    httpsURLConnection.getOutputStream());
144         outputStream.writeBytes(request);
145         //wr.write(json.toString());
146         //wr.flush();
147         //wr.close();
148         outputStream.flush();
149        /* outputStream.
150         BufferedWriter bufferedWriter =
151                 new BufferedWriter(new OutputStreamWriter(
    outputStream, "UTF-8"));
152         bufferedWriter.write(json.toString());*/
153         //bufferedWriter.flush();
154         //bufferedWriter.close();
155         outputStream.close();
156
157         httpsURLConnection.connect();
158
159         String response = "";
160
161         try{
162             response = getResponseFromStream(
    httpsURLConnection.getInputStream());
163             response = response.replace("\\\"", "\"");
164             response = response.replace("\"[", "[");
165             response = response.replace("]\"", "]");
166         } catch(FileNotFoundException e){
167             e.printStackTrace();
168             response = getResponseFromStream(
    httpsURLConnection.getErrorStream());
169             Log.d("httpserror", response);
170         }
171
172         return new JSONObject(response);
173     }
174
175 }
176
```

```java
package com.aakportfolio.aakatz3.seciotapp;

/**
 * Created by Andrew on 10/7/2016.
 */

public class iotsettings {
    public static final int IOT_HOME_NODE = 1;
    public static final int IOT_MOBILE_DEVICE = 0;
}

```

```java
package com.aakportfolio.aakatz3.seciotapp;

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.SharedPreferences;
import android.provider.Settings;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.KeyEvent;
import android.view.View;
import android.view.inputmethod.EditorInfo;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.preference.PreferenceManager;
import android.widget.TextView;
import android.widget.Toast;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.lang.reflect.Array;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity
implements View.OnClickListener, seciot.
SecIOTPollCompleteListener{

    LinearLayout controlLayout;
    EditText guid_text;
    Button guid_submit;
    SharedPreferences prefs;
    String guid;

    SecureAPI secureAPI;

    seciot foobar;

```

```
41      @Override
42      protected void onCreate(Bundle savedInstanceState) {
43          super.onCreate(savedInstanceState);
44          setContentView(R.layout.activity_main);
45
46          prefs = getPreferences(Context.MODE_PRIVATE);
47          secureAPI = SecureAPI.getInstance(this);
48
49          guid = prefs.getString("guid", null);
50
51          foobar = new seciot(guid, secureAPI, this);
52
53
54          controlLayout = (LinearLayout) findViewById(R.id.
    control_layout);
55          guid_submit = (Button)findViewById(R.id.guid_submit)
    ;
56          guid_submit.setOnClickListener(this);
57          findViewById(R.id.pull).setOnClickListener(this);
58          findViewById(R.id.push).setOnClickListener(this);
59          guid_text = (EditText) findViewById(R.id.editText);
60          guid_text.setOnEditorActionListener(new TextView.
    OnEditorActionListener() {
61              @Override
62              public boolean onEditorAction(TextView v, int
    actionId, KeyEvent event) {
63                  if(actionId == EditorInfo.IME_ACTION_DONE){
64                      guid_submit.performClick();
65                      return true;
66                  }
67                  return false;
68              }
69          });
70
71          if(guid != null){
72              guid_text.setText(guid);
73              try {
74                  foobar.pollState(this);
75              } catch (Exception ex){
76                  ex.printStackTrace();
77              }
78          }
```

```
 79      }
 80
 81
 82
 83      public void performUpdate(){
 84          controlLayout.removeAllViews();
 85
 86
 87          for(iotnode n : foobar.getNodes()){
 88              CheckBox cb = new CheckBox(this);
 89              n.index = foobar.getNodes().indexOf(n);
 90              cb.setText(n.toString());
 91              cb.setChecked(n.getState());
 92              final int idx = n.index;
 93              cb.setOnCheckedChangeListener(new
    CompoundButton.OnCheckedChangeListener() {
 94                  @Override
 95                  public void onCheckedChanged(CompoundButton
     buttonView, boolean isChecked) {
 96                      //foobar.setState(idx, isChecked);
 97                      Log.d("box", ""+idx);
 98                      try {
 99                          foobar.setState(idx, isChecked);
100                          foobar.pushState();
101                      } catch (Exception e) {
102                          e.printStackTrace();
103                      }
104                      Toast.makeText(getApplicationContext(),
     buttonView.getText(), Toast.LENGTH_SHORT).show();
105                 }
106             });
107             controlLayout.addView(cb);
108         }
109     }
110
111     @SuppressLint("CommitPrefEdits")
112     @Override
113     public void onClick(View v) {
114         switch(v.getId()){
115             case R.id.guid_submit:
116                 guid_text.setEnabled(!guid_text.isEnabled()
    );
```

```
117                 guid = guid_text.getText().toString();
118                 prefs.edit().putString("guid",guid).commit(
    );
119                 foobar.updateGuid(guid);
120                 try {
121                     foobar.pollState(this);
122                 } catch (Exception ex){
123                     ex.printStackTrace();
124                 }
125                 break;
126             case R.id.pull:
127                 try {
128                     foobar.pollState(this);
129                 } catch (Exception ex){
130                     ex.printStackTrace();
131                 }
132                 break;
133             case R.id.push:
134                 try {
135                     foobar.pushState();
136                 } catch (Exception ex){
137                     ex.printStackTrace();
138                 }
139                 break;
140             default:
141                 Toast.makeText(this, "Not Implemented",
    Toast.LENGTH_SHORT).show();
142         }
143     }
144 }
145
```