IoT Bandwidth Locators for User Evaluation

Da Huo
School of Arts and Sciences
Rutgers University, Piscataway
dh637@scarletmail.rutgers.edu

Hairong Wang
School of Engineering
Rutgers University, Piscataway
hw385@rutgers.edu

Shruti Das
Clark School of Engineering
University of Maryland, College Park
shrutid647@gmail.com

Parul Puri
School of Engineering
Rutgers University, Piscataway
pp646@rutgers.edu

Weizhong Kong
School of Engineering
Rutgers University, Piscataway
kwz20112938@gmail.com

Perry Wu
School of Engineering
Rutgers University, Piscataway
perrywu6@gmail.com

Pranathy Veldandi
School of Engineering
Rutgers University, Piscataway
vpranathy@gmail.com

Daniel Like
School of Engineering
University of Pennsylvania, Philadelphia
dlike230@gmail.com

Joshua Guo
Montgomery High School
Skillman, New Jersey
jguo435@gmail.com

***Abstract* — Smart home automation can empower everyday tasks by enabling limited human intervention between the number of devices on the same network, or within the same bandwidth connection. These everyday home devices send data through a heterogeneous network of sensors and actuators to a centralized database which then appropriately generates responses for the "things" connected. Though beneficial, this may also lead to external parties being able to control this data so security is a major issue.**

***Index Terms* — IoT, TI Sensortag, LARS, SSH Tunneling**

I. Introduction

Internet of things (IoT) .

Our IoT simulation involves two separate teams. Our project team is the Blue Team, or the defense team, and the other team is the Red Team, or the attack team.

While one of our main objectives is to build an end-to-end IoT framework to collect and analyze sensor data from devices, the remainder of our objectives revolve around combating the objectives of the Red Team.

This paper describes the design of an end-to-end security conscious IoT framework for everyday home devices, using openHAB, which actively uses machine learning techniques to detect attacks on its data.

Our initial system was set up to incorporate two different wireless communication protocols: Bluetooth Low Energy and Z-wave.

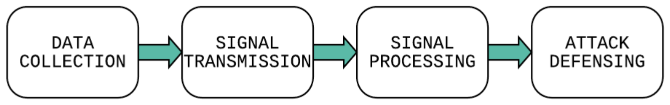

Figure 1: Block Diagram of the IOT Framework

II. Framework Architecture

A. *Overview:*

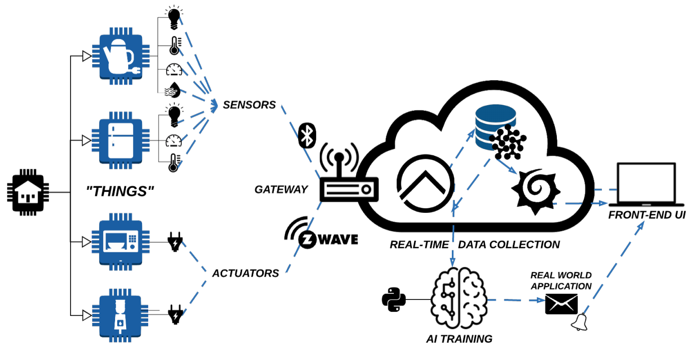

Figure 2: End-to-End Detailed Architecture

B. *Communication Protocols*

1) Bluetooth Low Energy
2) Z-wave

C. *Hardware Components*

1) Texas Instruments Sensortag CC2650
2) Aeotec Smart Switch 6

D. *Open Source Software Components*

1) openHAB: The open Home Automation Bus is an open source, technology agnostic home automation platform which runs as the center of a smart home. openHAB software integrates different home automation systems, devices and technologies into a single solution. It provides uniform user interfaces, and a common approach to automation rules across the entire system, regardless of the number of manufacturers and sub-systems involved.[7]

- *Architecture Overview:*
  openHAB is developed in Java and mainly based on the Eclipse SmartHome framework. It uses Apache Karaf together with Eclipse Equinox to create an Open Services Gateway initiative (OSGi) runtime environment. Jetty is used as an HTTP server. It is a highly modular software which can be extended to a variegated applications from creating common UIs to interacting with ever growing number of physical things through 'Add-Ons'.[7]

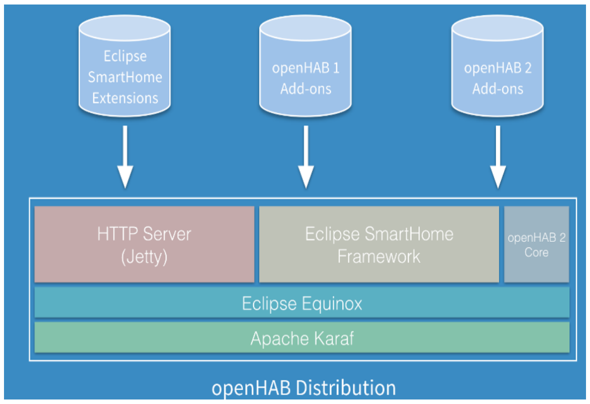

Figure 3: OpenHAB architecture overview

In openHAB things are the entities which can be physically added to the system like bluetooth sensors, Z-wave devices, web services etc. Bindings are the software adapters which make the things available in the home automation system. Items are the capabilities which can be utilized in our applications for either building any automation location or for creating customizable graphs/ UI. Items may have different states. The links are the glue between the items and the things. In our automation system, we use things like bluetooth sensors and Z-wave devices. The bluetooth sensors help us gather information about coffee machine and the refrigerator. These sensors are added as things and have different items like humidity, light, temperature, accelerometer etc. This sensor data can be used to detect temperature values, to learn patterns and help to predict when will the coffee be ready. The Z-wave controller communicates with the microwave, coffee machine, water cooler etc to provide us with data about those devices. Our system is able to receive the data from these things and has items like current consumption, power consumption, switch etc. We are able to automate the process of switching the devices On / Off at night by communicating with the nodes. We collect the bluetooth sensor data to detect light intensity at night and use that information to generate automation rules to switch off the Z-wave devices if there is no one in the break room. The rule engine in openHAB helps in the process of automation.

2) Influx Database:
InfluxDB is used as a data store for any use case involving large amounts of time-stamped data, including DevOps monitoring, log data, application metrics, IoT sensor data, and real-time analytics. InfluxDB helps us to persist time series data and conserve space on our system. It saves data for a defined length of time and auto expires unneeded data if any. It also provides SQL like language to query and interact with the data. We are able to persist all the items' states in the InfluxDB every minute. The InfluxDB can be linked and configured to OpenHAB which helps us to store and query all the data that openHAB monitors.[8]

3) Grafana:
Grafana is an open source software for time series analytics. It can be linked to many databases and helps to query, interact and display real time data. We configured the InfluxDB in Grafana and linked the database to openHAB, thus querying real time data from the database and displaying it through customizable graphs. We were able to query all the items available in openHAB and display their graphs on Grafana as all the data was persisting to InfluxDB and the database was configured in Grafana.[6]

III. Front End Application

The strategy we used to make the coffee email notification work is to constantly monitor the gradient of coffee pot's temperature. We take the gradient of temperature data over the past 5 minutes using the sliding window shown in Figure 3:

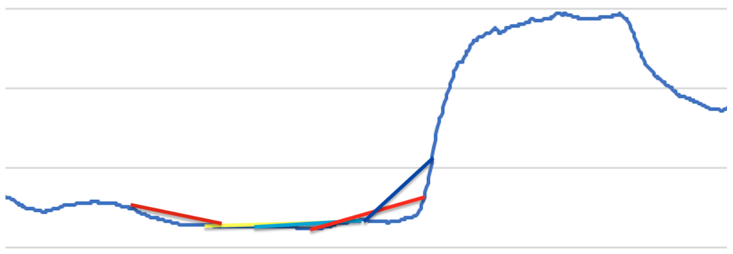

Figure 3: Gradient of Ambient Temperature

We analyzed temperature data over the past month to set the gradient threshold to be 0.003 C°/sec. When the gradient monitor detects a gradient greater than 0.003 C°/sec, the coffee machine is predicted to be on and making coffee. We will send out an email notification to everyone at this time.

In order to achieve real time data analysis regardless of sensor disconnection, we used a separate data analyzer server and transfer data through WebSocket.

For security purposes, when we send emails, we need to register our application to Google API and let the software send out the email.

IV. Artificial Intelligence Training

A. *Overview*

Artificial Intelligence and Machine Learning is used in this project to identify attacks from the Red Team. We trained regression models to predict future ambient temperatures. In the case of spoofed data from the Red Team, we would use our predictions to determine whether incoming data was likely to be falsified or not. In addition, we also monitored our data to detect data gaps, check results with other parameters, and account for additional noise.

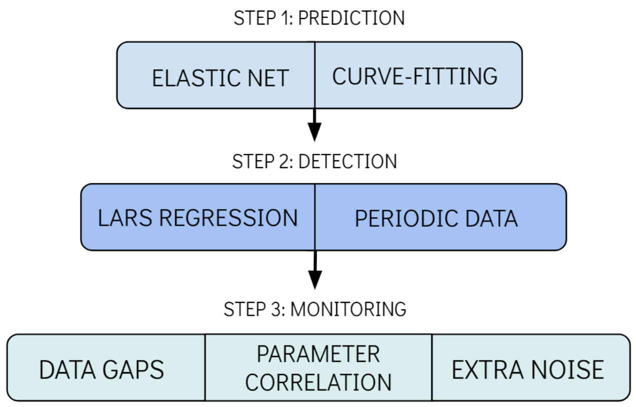

Figure 4:

B. *Training Models*

We started with three different regression models: Ridge, Lasso, and Elastic Net. These regression models are called *regularization methods*. They both minimize the sum of the squared error of the model on the training data while also reducing the complexity of the model (absolute sum of all coefficients). Ridge Regression retains all the features of the data and shrinks the coefficients. Lasso Regression only keeps very important variables and sets all other correlated ones to zero. The Elastic Net Regression is the combination of these two regressions. It groups correlated variables together, and an entire group of variables are included in the final model building if any one of the variables in the group is a strong predictor. This model is best used when training data has many data points (>10,000) because Ridge keeps too much data to be efficient, and Lasso removes too much necessary data. The Elastic Net model minimized the cost function to the greatest extent; therefore, the Elastic Net model was the optimal one [2]. However, we were able to reduce overlap of RMSE distributions of falsified and non-falsified data to the greatest extent with LARS, or Least-Angle Regression. This algorithm is also a linear regression algorithm, although it functions mainly by incrementally expanding the impact of highly explanatory variables [3].

C. *Predictor Evaluation*

We used the RMSE (root mean square error) loss function to evaluate the performance of the predictors we trained. This loss function computes the average of the squares of the absolute difference between predicted and observed values and returns the square root of this average. It is a useful measure of performance for regression because it provides the standard amount by which predicted data differs from observed data.

D. *Fake Data Simulation*

In order to test the accuracy of our anomaly detection techniques, we needed to be able to generate fraudulent data that appeared similar to the original data. To do this, we modified a multiplier for each data points, initialized at 1, by very small, random increments. This method ensured that the fabricated data would not have random spikes, and would thus look similar to the original data. Both the multiplier and the fake data were confined to specific bounds to ensure that the simulated fraudulent data did not deviate very significantly from the real data.

E. Prediction

For the first method, We trained an Elastic Net regression model to predict a given sensor value for the coffee pot using all sensor values for sensor readings a minute in the past. For example, one instance of this model predicted future ambient temperature based on object temperature, lighting, pressure, humidity, and predicted ambient temperature for the previous minute. The use of predicted ambient temperature, rather than measured ambient temperature, was important for this regression as it ensured that the model would rely only on recent measured data and previous predictions. A model that used measured instead of predicted ambient temperature data from the previous minute to predict current ambient temperature could reach extremely high levels of accuracy just by producing a value very close to the previous value. Thus, using predicted ambient temperatures resulted in predictions whose accuracy gave a far more valuable representation of the predictive value of the given sensor readings. While this specific model was not particularly useful for making long-term predictions as it required recent sensor data for each individual prediction, it proved useful for the purpose of anomaly detection.

We made a separate program to give us useful predictions of future data. This program uses measured data to predict data 15 minutes in the future. This program is effective because it can make predictions farther into the future without new sensor data. This prediction algorithm performed with an RMSE loss value of 1.83 after modifying the hyperparameters of the Elastic Net regression algorithm.

We also applied another methodology with higher accuracy: trend fitting. Before that we tried various algorithms(LSTM, LR .etc) for prediction. They all have same problem: huge accuracy fluctuation may happen when dealing with different test datasets. So we moved our focus from learning the feature of the whole day to simulating the data trend within a certain range near the current data point. So firstly we achieved fitting within time t-14 to t; then we make a prediction of temperature of time (t+19) which is 20 mins later . Finally we achieved that with only 0.6 RMSE loss. We also tried various degrees of trend fitting and we found that high degrees may have better performance in certain intervals, but have larger oscillation in the total period, so we used only one-degree fitting instead.

F. Anomaly Detection

Using the trained Elastic Net regression model from section B, we were able to classify incoming sensor data as valid or fraudulent. This was done by randomly generating falsified and non-falsified testing data sets and computing the model's performance on each of these data sets. The performance score used was the RMSE loss function. We then graphed the distributions of RMSE loss values for falsified and non-falsified data sets, and found that the overlap between these distributions was minimal. This made it possible to identify a RMSE loss cutoff that could be used to reliably distinguish between legitimate and fraudulent data, which performed with 95% accuracy. We were able to increase this level of accuracy to 100% by using LARS regression, predicting data 15 minutes into the future and using two consecutive data points to make this prediction. This 100% accuracy does not necessarily extend to all fraudulent data and instead only applies to our method of generating falsified data [4].

Additionally, we implemented two other ways to detect the fake data attack. The first one is use a searching dictionary. Attacker may cut certain interval from our normal dataset and duplicate it as fake data stream. To deal with that, for each data point with a time step T, we collected 100 points before T and then store them as a list. So finally we utilized a dictionary with the mapping relationship of (temperature of T, 100 temperatures sample before T). In this way, for each new input, we traverse all dictionary to check whether there is new repeat interval. The second method is check the zero-crossing frequency of temperature. Attacker may add some noise like sin() or cos() or square wave to the existed data stream to make it looks similar to normal data. To defend that, we defined a sliding window of length 100. Let it move from the start to the end of the current dataset. We calculated the maximum and minimum "count" of gradient changing. So when a new input comes, we compared the new calculated "count" with the safety threshold to find out whether any "Additional Noise Attack" happens.

G. Gap Detection

In order to detect when the Red Team jams our data, we designed a program that will notify when there are significant gaps in data collection. The sensors collect data every 10 seconds which get fed to a real time CSV file. We can use the program to find how many significant gaps in the data there was in a certain time frame and how long each gap was. The length of a gap to be considered significant is decided by the user. Small gaps in the data appear frequently as a cause of imperfections in the sensors and weak signal, and these gaps should not be confused with actual jamming attacks from the Red Team.

V. Future Work

A. Email Notification

Currently we use a Python SMTP interface to login to a Gmail account and send out an email notification when making coffee. For security purposes, we needed to register our application with the Gmail API and let the software send emails with special authentication keys instead of a username and password. [5] Email notifications were sent out whenever anomalous data was detected or gaps in data were detected.

VI. References

[1] H. Kopka and P. W. Daly, *A Guide to LATEX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] J. Shubham. “A Comprehensive Beginners Guide for Linear, Ridge and Lasso Regression.” *Analytics Vidhya*, 27 Apr. 2018.

[3] Efron, Bradley, et al. “Least Angle Regression.” *The Annals of Statistics*, vol. 32, no. 2, 2004.

[4] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. ACM Computing Surveys, 41:1-58, 2009. Y. Cheng and G. Church

[5] https://developers.google.com/gmail/api/guides/sending

[6] https://grafana.com/grafana

[7] https://www.openhab.org/docs/

[8] https://www.influxdata.com/time-series-platform/influxdb/