4M-02

A Code Generating and Verification Method from DSM Language for Secure IoT Applications

Tianxiang Zou† Hao Feng† Kenji Hisazumi‡ Akira Fukuda‡
Graduate School of ISEE, Kyushu University† Faculty of ISEE, Kyushu University‡

1 Introduction

Nowadays, the use of Internet of Things (IoT) is becoming more and more popular. IoT applications often involve payment, housing, navigation, shopping and healthcare in our lives. In these IoT applications, thousands of devices are connected together through the internet to enable real-time data exchange.

One of the most representative examples in IoT applications is Smart House Application[1]. In this application there are a large number and varieties of devices connected to each other via the Internet. There are temperature sensors, actuators, alarms and so on, and usually, there are more than 10 devices in one room. On the one hand, by observing and analyzing the data obtained from these devices, users can switch actuators for a more comfortable living environment. On the other hand, if these data are leaked or tampered with, they may pose a threat to the users' safety.

2 Motivational Example

2.1 Scenarios

For temperature sensor in the room. By analyzing temperature data in the room, can make user's live more comfortable. As for single man, even if the raw data is seen by other people, may not affect his individual life. But for the single woman, if the raw data is leaked, through the data, other people can easily speculate the woman's presence and absence during this period. Once such data is obtained by criminals, it is likely to pose a threat to the safety of the woman. However, as for the average data in one day, others can not speculate user's lifestyle easily, even if the data is leaked, it does not pose a threat to the user's safety. Thus, data related to privacy and security need to be processed according to different users and environments.

Moreover, for data storage, if the data is stored on a safe cloud server with public key certification, such as google, the security of the data will be guaranteed. However, if data is stored on a server which has unknown security, it can result in data leakage, and may pose a threat to the safety of users. Thus, the users also need to have the ability to check and select the method and location to store data.

2.2 User Demand

Users can not only choose the default data processing method provided by system, but also choose other methods according to their own needs, and also can check and choose where to save the data. Moreover, before execution on devices, users can verify that the program meets their needs.

3 Related Research

3.1 Middleware

In order to develop IoT applications effectively and safely, a key technology in the realization of IoT system is middleware, which is usually described as a software system designed to be the intermediary between IoT devices and applications[2].

3.2 Domain-Specific Modeling Language

To simplify the problem domain and develop on different platforms, it is usually to use domain-specific modeling language (DSML) for development. DSML is language usually designed with a specific purpose[3]. DSL methodology uses models to describe the individual components of the domain system, which is usually based on graphical or textual description.

3.3 Existing Services

Here are two main existing open-source middleware used DSML for IoT smart house application: openHAB and Node-red.

openHAB: openHAB represents a local general-purpose middleware used DSML for smart home gateways and is fully based on the Eclipse SmartHome project, which allows building smart home solutions as the most common application area of IoT[4]. By using GUI dashboard, users can easily view the data collected by sensors and control the devices in their houses. However, the security of openHAB is currently undefined[4], also, openHAB does not have any data processing functions.

Node-RED: Node-RED is an open-source IoT middleware platform used DSML from IBM, it offers rapid prototyping, unique integration of social network injections, event-driven templates and real-time interaction with physical devices[5]. Users can easily wire devices together by using Flow-based programming in GUI dashboard. The security method of Node-RED is Bcryptjs & AES[5]. However, Node-RED does not have a ready-to-use data processing

method. If users want to process data, they need to enter the code themselves.

3.4 Development Demand

Therefore, for our system for data privacy and security in smart house, it is important to provide varieties ready-to-use privacy and security data processing functions. Similarly, it is important to provide diverse data storage methods and locations. Moreover, it must be easy to use.

4. Proposed Method

Next, we will introduce the proposed method to implement our proposal system. Our proposal system not only need to meet the user demand mentioned above, but also need to meet development demand. For easy to develop, we decide to use DSML to build a middleware with GUI modeling panel which can automatically generate executable code. For providing varieties of functions and storage methods, we create varieties of feature icons. To verify the program meets the user's needs, the user can model the needs and verify the code generated by the middleware.

4.1 Eclipse Sirius

Sirius is a graphical DSML Tools in Eclipse. We propose to use Sirius to build a GUI modeling penal, define the relationship among the functions and create icons for functions. Developers can drag and drop the icons for the required features and connect them with lines to complete the modeling. And in this process, the developers do not need to enter any code.

4.2 Code Generation

Next, we propose to use python to analyze the model in Sirius. Through model analysis, we can get the key information, such as function name, parameters and order of functions used. Then we use the library called jinja2 in python for code assembly. Jinja2 is a modern and designer-friendly templating language for Python. It can generate any kinds of code through templates. Jinja2 can use the key information to find specific code templates, and assemble them, and finally generate executable code.

4.3 Runtime Verification

Finally, we need to verify before the code is deployed to the device for execution. In smart house application, the devices in the house may have a default driver or some programs provided by service provider. Here we propose to use Lustre for verification. Lustre is a synchronous declarative language for programming reactive systems. And it is also a dataflow language with textual and time-discrete. Using Lustre for verification, not only can check whether the code is correct, but also can check whether the logic and execution order meet user's need.

5. Modeling

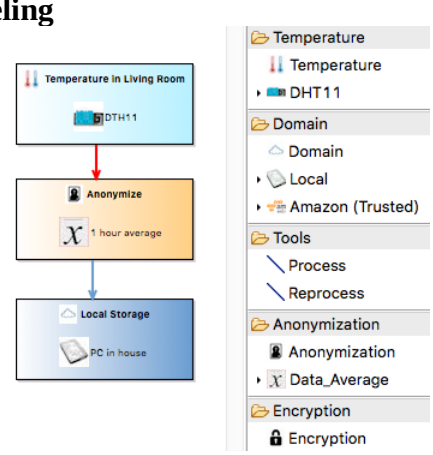

Figure 1. Development Modeling Panel

The developer panel of the proposal system is shown in the figure 1. As is shown in the picture, we want to collect the temperature data collected in living room and calculate the average number for one hour, and save them in local PC. thus, we drag and drag the icons and connect them with line. Finally, when we run the code generator, the executable code and Lustre verification file will be generated automatically.

On the other hand, the User can fill out a drop-down list to define their own privacy and security needs for the program before the program is deployed on the devices and executed. After filling out, a Lustre file will be generated. This file can be verified with the Lustre verification file which generated during modeling. When the needs are fully met, the program can pass the verification and can be allowed to execute.

6. Conclusion

This study proposes to use Sirius, python and jinja2, and Lustre to implement the Smart House application system for data processing and needs verification for privacy and security. As for future works, we will add more functions to this system and evaluate the system through case study.

Reference

[1] Pankesh Patel et al., Enabling high-level application development for the Internet of Things, in The Journal of System and Software 103, pp. 62-84, 2015.
[2] Anne H. Ngu et al., IoT Middleware: A Survey on Issues and Enabling Technologies, IEEE Internet of Things Journal, VOL. 4, No. 1, 2017.
[3] Viyović et al., Sirius: A rapid development of DSM graphical editor, INES 18th, pp. 233-238, IEEE, 2014.
[4] Velázquez et al., Securing openHAB Smart Home through User Authentication and Authorization, 2018.
[5] Scott et al., A comparative study of open-source IoT middleware platforms, KTH, Stockholm, 2018.