Complex Event Processing for Object Tracking and Intrusion Detection in Internet of Things Environments

Shang-Nan Yin, Ho-Seok Kang, and Sung-Ryul Kim*
Department of Internet and Multimedia Engineering
Konkuk University, Seoul, 05029, Korea
{yinshangnan, hsriver}@gmail.com, and kimsr@konkuk.ac.kr

Abstract

With the development of Internet of Things (IoT), there have been more and more services and applications deployed in physical spaces and information systems. The massive number of sensors and devices are embedded in IoT environments, which produce huge amounts of data continuously for the IoT systems and platforms. Processing these data stream generated by the IoT networks with different patterns has raised new challenges for the real-time performance of intrusion detection system (IDS) in IoT environments, which has to react quickly to the hacking attacks and malicious activities to IoT. In this paper, a complex event processing (CEP) based IDS model for object detection tracking and intrusion detection in the IoT environments is proposed. Esper, an open source complex event processing engine is used to develop the model. In this model, the cincoming streams of data are detected by Esper engine according to the predefined EPL rules. And then, trigger corresponding listeners, the normal events are sent to the higher layer application as an new event through the adapter. In the alert event processor, the abnormal events are divided into the attack events and the other causes of abnormal events, such as sensor fault, transmission delay, etc.

Keywords: Intrusion Detection, IoT Security, Complex Event Process

1 Introduction

The Internet of Things (IoT) provides connectivity for anyone at any time and place to anything at any time and place. IoT devices are poised to become more pervasive in our lives than mobile phones and will have access to the most sensitive personal data such as social security numbers and banking information. As the number of connected IoT devices constantly increase, security concerns are also exponentially multiplied.

IDS aims to monitor system or network activities of malicious activities based on predefined policies and produce alerts and reports to administrators. For different security threats in IoT systems or platforms, effective IDS technology needs to be simple, real-time and accurately detected. IDS based on multi-agents make use of agents' autonomy, mobility, and independent logic processing capability [8]. Multi-agent IDS system consists of detection agent, host agent and network agent. IDS agents work at perceptive layer and communicate to each other collaboratively, which greatly reduce network load and time delay. Host agent on network terminals will manage detection agents, which is responsible for listening security events. Filtered data of host agent will be submitted to network agent and finally transmitted to the console. Compared with other IDS technology such as game theory model, Bayesian network and machine learning, multi-agent IDS performs better in real-time efficiency aspect (processing capability, concurrent and maximum TCP per second). IDS based on game theory model, Bayesian

*Corresponding author: Department of Internet and Multimedia Engineering, Konkuk University, 120 Neungdong-ro, Gwangjin-gu, Seoul, 05029, Korea, Tel: +82-2-450-4134 network, and machine learning needs to operate events on certain computation models, thus, they are relatively weak in real-time performance [10]. However, these kinds of IDS have better system robustness and lower network failure rate, since their centralized deployment mode.

Different from the computation distribution concept of multi-agent IDS, 6LoWPAN technology has been studied as a new way to provide IP identity and network connectivity to IDS of the perceptive layer of IoT, which is tiny, inexpensive and low-powered. [9] has implemented a novel intrusion detection system based on 6LoWPAN for the IoT in the Contiki OS detecting routing attacks such as spoofed or altered information and sinkhole, which supports more effective data transmission and quick response to malicious activities with limited energy and memory capacity. Event processing architecture has been used as another tool to solve the problem of events streams in IoT. [11] have designed and implemented IDS using information flow processing (IFP) [4] to extract event from multiple and distributed sources as soon as relevant information arrived, which promptly detect possible attacks with real-time actions and alerts. [5] takes advantages of collaborative mechanisms to enhance the efficiency of event processing workflow on the single node and has implemented an IDS for IoT platform Semantic Room which achieved high detection accuracy and small detection delays in security threats (e.g. port scans, botnets) and frauds [7].

In this paper, we propose a Complex Event Processing (CEP) [3] based solution for object tracking and intrusion detection system model in IoT environments. Complex Event Processing (CEP) technology provides new solutions in the field of complex pattern identifications and real-time data processing, which can be used to improve the performance of traditional IDS in IoT environments. This IDS model will real-time monitoring and tracking each layer event streams according to the predefined EPL rules in IoT environments. Rest of the paper is organized as follows: Section 2 introduces the layered structure of IoT and the complex event processing (CEP). Section 3 explains the proposed model and Section 4 has the conclusion.

2 Related Work

2.1 A Layered IoT System

The research and applications of Internet of Things (IoT) are attracting more and more attentions and the security issues have become increasing prominent. Figure 1 shows a common layered structure of an IoT.

1. Perception Layer: The perception layer is also known as ‘Device Layer’. It consists of the physical objects and sensor devices. The sensors can be RFID, 2D-barcode, or Infrared sensor depending upon objects identification method. This layer basically deals with the identification and collection of objects specific information by the sensor devices. Depending on the type of sensors, the information can be about location, temperature, orientation, motion, vibration, acceleration, humidity, chemical changes in the air etc. the collected information is then passed to Network layer for its secure transmission to the information processing system.

2. Network Layer: The network layer can also be called ‘Transmission Layer’. This layer securely transfers the information from sensor devices to the information processing system. The transmission medium can be wired or wireless and technology can be 3G, UMTS, Wifi, Bluetooth, infrared, ZigBee, etc depending upon the sensor devices. Thus, the Network layer transfers the information from Perception layer to Middleware layer.

3. Middleware Layer: The devices over the IoT implement the different type of services. Each device connects and communicates with only those other devices which implement the same service type.

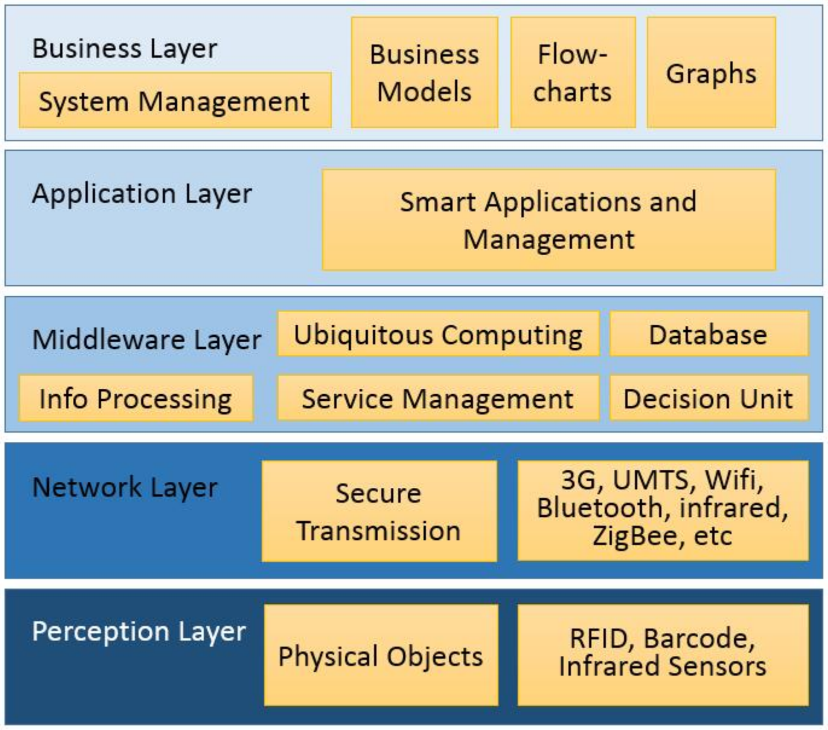

Figure 1: The Structure of An IoT

This layer is responsible for the service management and has a link to the database. It receives the information from Network layer and store in the database. It performs information processing and ubiquitous computation and takes an automatic decision based on the results.

4. Application Layer: This layer provides global management of the application based on the objects information processed in the Middleware layer. The applications implemented by IoT can be smart health, smart farming, smart home, smart city, intelligent transportation, etc.

5. Business Layer: This layer is responsible for the management of overall IoT system including the applications and services. It builds business models. Graphs, flowcharts etc based on the data received from Application layer. The real success of the IoT technology also depends on the good business models. Based on the analysis of results, this layer will help to determine the future actions and business strategies.

2.2 Complex Event Processing

Complex Event Processing (CEP) is an emerging technology to filter and process events in real-time scenarios. The basic information processed in CEP is defined as events and any phenomenon happened in physical spaces can be modeled as an event. CEP allows users to specify the events according to their own interest, such as transmission events in wireless sensor network or operating events in enterprise applications. Events have various relationships among them and complex events usually consist of multiple simple events. CEP technology can be used to detect meaningful complex relationships among events and respond to them as quickly as possible in real-time data, which makes it easier to identify event streaming with temporal and spatial constraints (e.g. `‘‘A then B within 5 minutes within 10 miles’’`). To some extent, CEP is a set of tools, practices, and techniques used to drive external information applications and systems [1]. CEP engine works a bit like a database turned upside-down which stores quires constraints and runs the data through. Thus, CEP has the advantages as follows: high throughput (information rate from 1k to 100k), Low latency (ranging from a few milliseconds to a few seconds) and complex pattern identifications (event relationships, event aggregations, and time or length windows, etc.). Typical examples of CEP applications are business process tools, finance management, network monitoring and sensor network automation [6].

Vertical causality: tracking events up and down the layers. Activity at each layer is translated into activities at the layers below and conversely. Those lower-level activities must complete successfully in order for the higher-level activities to also complete successfully. However, an activity at the top causes activities at successively lower levels, which in turn cause other activities to happen at the top. We call this vertical causality.

Event aggregation: making high-level sense out of low-level events. Event aggregation will in turn depend upon technology for recognizing patterns of events in large amounts of lower-level event traffic, in real time. And it depends first on an ability to express patterns consisting of multiple events together with their common data and timing. If event aggregation is implemented properly, it can give us the ability to track the lower-level events that were aggregated to create a high-level event [2].

Esper is an open-source Java-based software product for CEP and Event Stream Processing (ESP) that analyzes series of events for deriving conclusions from them. Esper provides a rich Event Processing Language (EPL) to express filtering, aggregation, and joins, possibly over sliding windows of multiple event series. It also includes pattern semantics to express complex temporal causality among events (followed-by relationship). Esper provides a highly scalable, memory-efficient, in-memory computing, SQL-standard, minimal latency, real-time streaming-capable Big Data processing engine for any-velocity online and real-time arriving data and high-variety data, as well as for historical event analysis. In this paper, we use esper to develop the IDS model.

3 New IDS Model in IoT

In this section, we propose object tracking and intrusion detection system model based on CEP in the IoT environments, and then introduce this model in detail. Figure 2 shows this model.

Event. An event is an object that is a record of an activity in a system. The event signifies the activity. An event may be related to other events. There are particular attributes or data in the event. For example, a temperature sensor event attributes are as follows:

```
Event{
     EventId;
     ObjectId;
     SensorId;
     Data;
     TimePro;
     Location;
     ......;
}
```

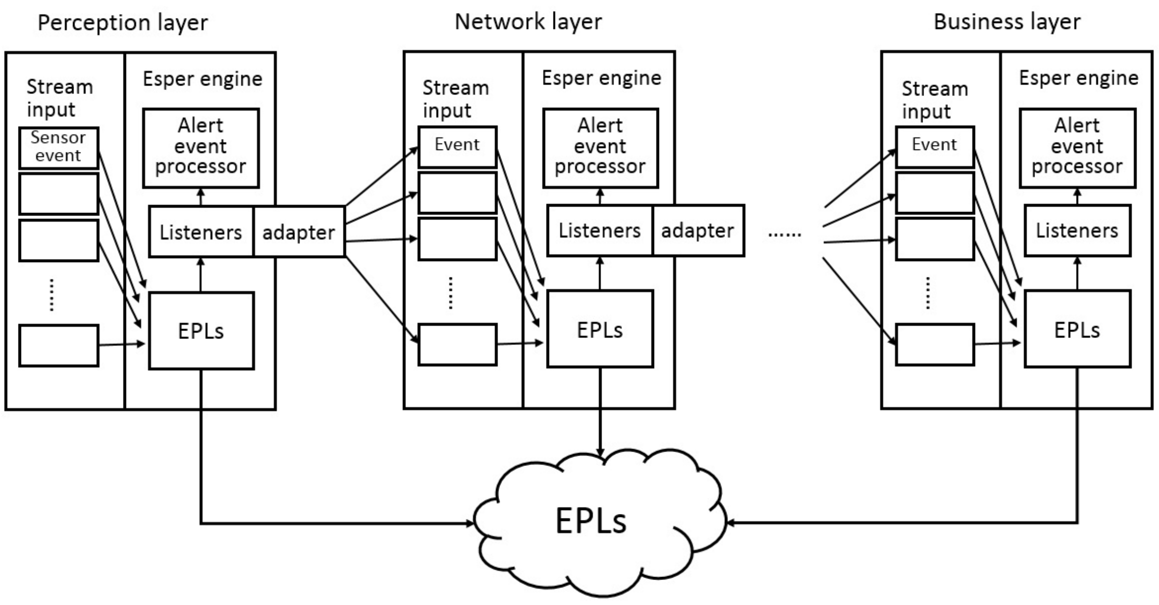

Figure 2: Proposed IDS Model in IoT

Every event has `EventId` is a unique identifier field. `ObjectId` shows sensor's type, is temperature in here. `SensorId` is sensor id. `Data` is the value of the sensor. `TimePro` means the time of this event produced. `Location` is the sensor position.

Esper Engine. Esper engine is the CEP engine. The incoming streams of data are processed by this engine according to the predefined rules (EPLs). In this IDS model the step involved in event processor are as follows:

Step 1: Initialization and launch the esper engine, and define the types of event attributes.

Step 2: Generate EPL rules. Run the queries (EPLs) continuously on the event streams as and when they arrive.

Step 3: The events are sent to the esper engine. Real-time detection events through EPL rules and then trigger corresponding listeners.

Step 4: Through the adapter, the normal events are sent to the higher layer application as an event. The abnormal events send to the alert event processor.

Step 5: Distinguish between the attack events and the other causes of abnormal events.

Following JAVA code snippet is used for getting an Engine instance.

```
EPServiceProvider esperEngine = EPServiceProviderManager. getDefaultProvider();
EPAdministrator admin = esperEngine.getEPAdministrator();
EPRuntime runtime = esperEngine.getEPRuntime();
String sensorEvent = sensorEventType.class.getName();
```

The input to the CEP engine is the event streams. `sensorEvent` has defined the name of the event stream. In esper engine, we can also input multiple event streams. `sensorEventType` is the JAVA class that contains event attributes and event types. An event can be sent to the esper engine as follows:

```
runtime.sendEvent(event);
```

The events are sent to the esper engine, and then detection the events by defining the EPL rules. In this model, we store all the EPL rules in the database. When the esper engine startup to call corresponding EPL rules in the EPL database. Therefore, the EPL rules are the most key component for this intrusion detection system.

Examples of EPL rules: Consider the simple example of identifying the people who enter into some of the privileged locations into which they are not supposed to enter. This event occurs in the perception layer. RFID tags have a priority associated with them. People are given a RFID based on their privilege. These event attributes are as follows:

```
Event{
     EventId;
     ObjectId;
     SensorId;
     Rfid;
     SensorPriority;
     TimePro;
     Location;
     ......
}
```

The EPL rule statement of the engine as follows:

```
String epl = ‘‘select SensorId, Rfid, SensorPriority from sensorEvent’’;
EPStatement stat = admin.createEPL(epl);
```

Here `SensorPriority` is the location priority.

This is just a simple EPL rule statements. We can create an EPL rule statement which through pattern matching between a pre-specified spatiotemporal pattern and the incoming data streams. The event attributes can contain some network data, such as IP address, port, protocol, packets and so on. Therefore, we can also create the EPL by the intrusion detection algorithm.

Esper engine provides different ways for the application to receive the results of the detections. Through the adapter, the normal events are sent to the higher layer application as an event. The abnormal events send to the alert event processor. For the abnormal event, we discretion through the EPL statements, maybe it is an attack behavior or sensor fault. All the detection results output by the listener. Below mentioned is the code snippet for adding a listener.

```
stat.addListener(new priorityListener());
```

The detection results trigger the different listener. The normal events trigger the listener to generate a new event and are sent to the higher layer by the adapter. In the higher layer, this new event will be sent to another esper engine for detection. In the process of the whole, esper will real-time tracking events.

Esper provides runtime configuration. Through a Java Management Extension (JMX) MBean, we can remote dynamic statement management. Therefore, when adding a new sensor, changing the event type, adding the new EPL statements or adding the new listener and so on, we don’t need to restart the esper engine.

We need to configure esper engine according to this IDS model. We will improve the intrusion detection system model by a large number of experimental. We will futher research detection algorithm that can be used in the EPL rules, in order to build the more powerful esper engine.

4 Conclusion

In this paper, we present intrusion detection system model based on complex event processing in the IoT environments. This IDS model will real-time monitoring and tracking each layer event streams according to the predefined EPL rules in IoT environments. Trigger corresponding listeners to process the events. We describe the structure and process of this model in detail. The Internet of things (IoT) is vast amounts of distributed systems. The complex event processing (CEP) will play a vital role in the IoT environments. In the future, We will build and test this IDS model in the reality IoT environments in order to further improve the intrusion detection system model. We believe that can use complex event processing technology to accurately predict attack behavior in the future.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2016R1D1A1B02011964).

References

[1] D. Anicic, S. Rudolph, P. Fodor, and N. Stojanovic. Stream reasoning and complex event processing in etalis. *Semantic Web - On linked spatiotemporal data and geo-ontologies*, 3(4):397–407, October 2012.

[2] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Professional, 2002.

[3] G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 44(3):1–70, June 2012.

[4] A. Halinen, J.-A. TÖrnroos, and M. Elo. Network process analysis: An event-based approach to study business network dynamics. *Industrial Marketing Management*, 42(8):1213–1222, November 2013.

[5] S. Jajodia, S. E. Noel, and E. B. Robertson. Intrusion event correlation system. Technical Report US 12/758,135, George Mason Intellectual Properties, Inc., 2012.

[6] J. Lang and J. JanÍk. Reactive distributed system modeling supported by complex event processing. In *Proc. of the 3rd Eastern European Regional Conference on the Engineering of Computer Based Systems (ECBS-EERC'13), Budapest, Hungary*, pages 163–164. IEEE, August 2013.

[7] G. Lodi, L. Aniello, G. A. D. Luna, and R. Baldoni. An event-based platform for collaborative threats detection and monitoring. *Information Systems*, 39:175–195, January 2014.

[8] Z. Ran. A model of collaborative intrusion detection system based on multi-agents. In *Proc. of the 2012 International Conference on Computer Science and Service System (CSSS'12), Nanjing, China*, pages 789–792. IEEE, August 2012.

[9] S. Raza, L. Wallgren, and T. Voigt. Svelte: Real-time intrusion detection in the internet of things. *Ad hoc networks*, 11(8):2661–2674, November 2013.

[10] M. Yun and B. Yuxin. Research on the architecture and key technology of internet of things (iot) applied on smart grid. In *Proc. of the 2012 International Conference on Advances in Energy Engineering (ICAEE'10), Beijing, China*, pages 69–72. IEEE, June 2010.

[11] C. V. Zhou, C. Leckie, and S. Karunasekera. A survey of coordinated attacks and collaborative intrusion detection. *Computers and Security*, 29(1):124–140, February 2010.

Author Biography

Shang-Nan Yin is a Ph.D. student of the division of Internet and Multimedia Engineering at Konkuk University, Seoul, Korea. He received his Master degree in Internet and Multimedia Engineering at Konkuk University, Seoul, Korea. His recent research interests are distributed algorithms, information security and data mining.

Ho-Seok Kang is a postdoctoral researcher of the division of Internet and Multimedia Engineering at Konkuk University, Seoul, Korea. He received his Ph.D. degree in computer engineering at Hongik University, Korea. His recent research interests are in network security, network protocol, mobile security, distributed algorithms and cloud computing.

Sung-Ryul Kim is a professor of the division of Internet and Multimedia Engineering at Konkuk University, Seoul, Korea. He received his Ph.D. degree in computer engineering at Seoul National University, Korea. His recent research interests are in cryptographic algorithms, distributed algorithms, security in general, cloud computing, and data mining.