Policy-Driven Security Management for Fog Computing: Preliminary Framework and A Case Study

Clinton Dsouza Gail-Joon Ahn Marthony Taguinod
Laboratory of Security Engineering for Future Computing (SEFCOM)
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
{cvdsouza, gahn, mtaguino}@asu.edu

Abstract

With the increasing user demand for elastic provisioning of resources coupled with ubiquitous and on-demand access to data, cloud computing has been recognized as an emerging technology to meet such dynamic user demands. In addition, with the introduction and rising use of mobile devices, the Internet of Things (IoT) has recently received considerable attention since the IoT has brought physical devices and connected them to the Internet, enabling each device to share data with surrounding devices and virtualized technologies in real-time. Consequently, the exploding data usage requires a new, innovative computing platform that can provide robust real-time data analytics and resource provisioning to clients. As a result, fog computing has recently been introduced to provide computation, storage and networking services between the end-users and traditional cloud computing data centers. This paper proposes a policy-based management of resources in fog computing, expanding the current fog computing platform to support secure collaboration and interoperability between different user-requested resources in fog computing.

1. Introduction

Internet of Things (IoT) has gained considerable popularity from both academia and industry professionals. IoT combines information and computing processes to control very large collections of different objects [1]. The concept of IoT encompasses every smart connected device, including personal devices utilized by people in their everyday lives [2]. Such connectivity and shareability enable the proliferation of connected objects, and in turn create data explosion which comes from billions of devices located around the world. However, unless these disparate devices can work together to create meaningful information and services, all the data from devices may be meaningless [3]. Therefore, the integration must be conducted seamlessly and intelligently.

The increased use of a "pay-as-you-go" cloud computing model has reduced the overall cost of users owning and managing private data centers. This has given rise to increasing user demand for computing, networking, and storage resources as well as the need for efficient management and access to highly virtualized resources. However, current computing models cannot account for and handle the huge data load, and thus require an innovative approach with the capability in communicating more closely with physical devices by extending cloud computing services to the edge of the network. As a result, fog computing also termed edge-of-network has recently been introduced [7]. Fog computing is a virtualized platform providing computing, networking and storage services between end devices and traditional cloud computing data centers. The primary purpose of such an environment is to deliver seemingly infinite collection of resources for fulfilling customers' needs. Given the high volume of interplay required between the fog environments, interacting end-user devices, and cloud computing data centers, a number of security concerns need to be addressed. In particular, due to the distributed nature of the fog computing, a more dynamic and robust policy management is necessary to accommodate diverse security requirements. In this paper, we articulate research challenges in policy management for fog computing and propose a policy-driven security management approach including policy analysis and its integration with fog computing paradigm.

This paper is organized as follows. Section 2 briefly overviews fog computing model including architectural components. In Section 3, we discuss the proposed policy management framework for fog computing followed by a case study with the fog computing testbed in Section 4. Section 5 describes the related work and Section 6 concludes

IEEE IRI 2014, August 13-15, 2014, San Francisco, California, USA
978-1-4799-5880-1/14/$31.00 ©2014 IEEE

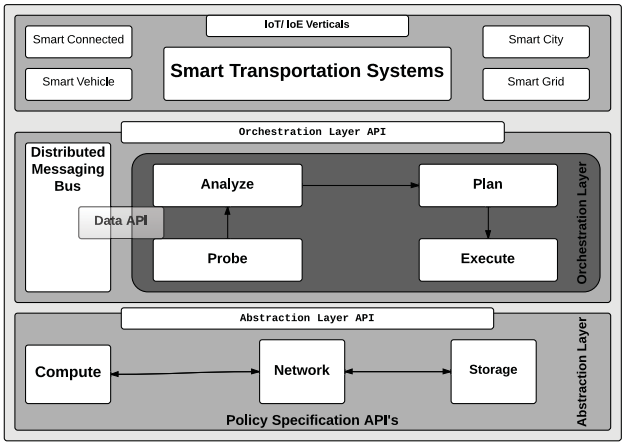

Figure 1. Fog Computing: Architectural Components [6]

the paper along with the future directions.

2 Overview of Fog Computing

To understand diverse requirements for fog computing paradigm, we leverage Smart Transportation Systems (STSs) as an exemplary use-case for fog computing in this section. An abstract view of fog computing is illustrated in Figure 1. STSs are heterogeneous distributed systems designed for enabling real-time communication between commuters and smart systems and constantly monitoring traffic activities to allow traffic preemption and guide commuters safely. In addition, STSs are expected to collect environmental data which includes traffic density, vehicle routing, vehicle speed, pre-emptive emergency routing, and so on. Bonomi et al. [6] highlighted the key requirements for fog computing which are driven by a STS environment. Fog computing architecture comprises three core components as follows:

- Internet of Things (IoT) Verticals
- Orchestration Layer
- Abstraction Layer

Each layer comprises both virtual and physical components which contribute to the efficient and dynamic functionality of a fog system. Figure 2 provides another view of fog computing architecture and its components. As illustrated in Figure 2, the IoT verticals are defined as tenant applications or products which are rented for use. Given the flexibility of fog computing platform and its interoperable nature, multi-tenancy is unique and highly desirable feature supported by a fog computing environment. Multi-tenancy feature enables multiple clients to host their applications on a single server and these clients (tenants) host their applications on a single fog computing instance. The orchestration layer supports data aggregation, decisions, data sharing and migration, and policy management for fog computing. The details of this layer will be discussed in the subsequent section. Finally, the abstraction layer is responsible for exposing a uniform and programmable interface to the client and hides the platform heterogeneity. Similar to the cloud model, this layer relies on virtualization technologies and exposes generic APIs that clients can invoke whilst developing applications to be hosted in a fog computing platform.

We now present key features–in addition to those presented in [6]–which are targeted to not only support certain use-case scenarios but provide a generic direction to support a wide-range of users and reusable systems:

- *Fog Node* (FN) is defined as heterogeneous semi-virtualized components deployed in a variety of environments. These nodes can be deployed in the core, edge, access networks and endpoints of a fog environment with diverse programmable components. The primary focus of FN is to facilitate seamless and uniform resource management including management of the networking, computation and storage allocation of each node. Each node can be a "Tiny Cloud" providing short term support to users and application requests. The design of an FN should be in a modular form with the plug-in capability or new nodes when the current one fails.
- *Fog Instance* (FI) is a virtualized instance as a supporting module for FN in their resource management and service requests. FIs are spawned dynamically based on resource requirements and are intended to provide computing, networking and short term storage services. Figure 2 shows how FNs and FIs interact with each other including other objects.

The subsequent section further articulates the orchestration layer of the fog computing architecture and relevant issues in policy management.

2.1 Orchestration Layer and Policy-based services

The middle layer of fog computing consists of multiple components such as *Data* API which involves a generic framework on how data is shared across the platform and *Orchestration Layer* API which comprises of the core analytics and intelligence services for fog computing. The

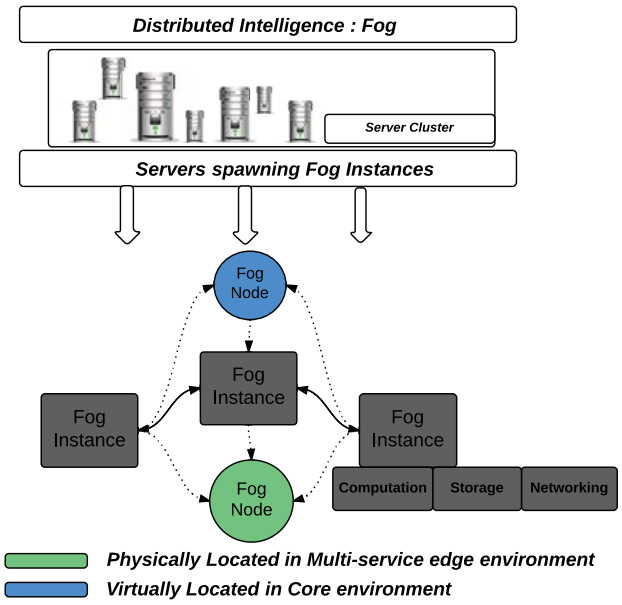

Figure 2. Fog Node Workflow

orchestration service as a whole focuses on the life-cycle management of services including policy services which need to be dynamic and are expected to account for the distributed nature of the fog computing paradigm. The Orchestration Layer API in particular focuses on four major functionalities: probing the application for data, analyzing the acquired data, planning the allocation and management of resources to manage the request, and enforcing a decision.

Given the numerous interacting components in the orchestration layer, policy-based service is a dominant component that needs further in-depth analysis and development. In [6], Bonomi et al. proposed several components in the service orchestration layer including foglet software agent, distributed databases, policy-based service orchestration, and scalable messaging bus and introduced a policy-based orchestration framework including the specification of abstract requirements that need to be accomplished by the proposed framework. Based on such service modules, we attempt to extend the framework by defining a policy management module that complements the existing framework and provides policy-based security management for the fog computing paradigm.

The policy management module involves the presence of a decision maker engine that enables the defined policy-based orchestration framework to enforce policies based on actual interactions. Additionally, our policy module attempts to concentrate on intricate communication modules which makes the fog computing architecture unique. Our module extends and covers the requirements set by the current framework thus providing assurance to the fog computing paradigm.

3 Policy-Driven Security Management: Preliminary Framework

As discussed in the previous section, policy collaboration is an important component in the orchestration layer of a fog computing model. The concept of policy collaboration is introduced with the goal to support secure sharing and communication in a distributed environment. However, unlike previous approaches which focused on policy collaboration from a singular perspective, a fog computing architecture involves not only the virtual component interaction, but also a physical component interaction with each other as well as associated virtual components. This would imply that FNs and FIs would communicate with physical devices and cloud computing data centers in parallel. These multi-level collaboration requirements give rise to a new set of security problems involving identity management, resource access management, distributed decision enforcement, dynamic load-balancing, quality of security and service, and so on. The policy management module as part of the extended framework helps address the highlighted security issues.

Our policy management module, as depicted in Figure 3, is primarily dedicated to support the orchestration layer of the fog architecture, thus the bulk of the computation occurs within the fog nodes and the supportive fog instances depending on the policies to be enforced and services being requested for provisioning task. The extended policy management framework consists of modules wherein each module plays a specific and decisive role and can be plugged and played in real-time based on configurations defined by administrators of corresponding hosted applications and the owner of a particular fog environment.

We summarize each module and their responsibilities in the framework as follows:

Policy Decision Engine This module is programmed to make aggregated decisions based on data provided by all attached components. Based on the services requested and the target user, the policy decision engine analyzes the rules defined in the Policy Repository and generates a decision which is then later enforced.

Application Administrator The multi-tenant nature of the fog computing paradigm raises the requirement for an administrator to define policies and rules that

bind a user to a particular applications and allow for secure collaboration and migration of client data across multiple FNs owned by a particular application or multiple applications.

Policy Resolver The policy framework adopts attribute-based security framework wherein all users are authenticated and identified based on a set of attributes which they present. One or more of these attributes will be used to validate the identity of the user and thus the policy resolver consists of multiple subcomponents to verify the request of the user based on his attributes and entitle the presented identity with associated rules and policies.

(a) *Attribute Finder* This module analyzes the set of attributes presented by a user and queries the attribute database to determine the identity of the user. The attribute finder then sends the result to the Policy Resolver for further aggregation.

(b) *Attribute Database* A repository of user attributes identifying a users access privileges against a requested resource.

Policy Repository A secure repository consisting of rules and policies which are referred by the Policy Decision Engine while the policy decision is made.

(a) *Policy Rules* The set of policies/rules that defines multi-level policy domains of a fog computing environment covering operational, security-related, and network management requirements. Once user attributes are matched by the Policy Resolver, the Policy Decision Engine will refer to the policy rules to determine (i) the influence of governance and provisioning that allows the user to access the requested applications and (ii) what constraints need to be enforced based on the user's access privileges.

Policy Enforcer: The most active component of the policy management framework is the Policy Enforcer. The Policy Enforcer module resides within either a virtual instance such as an FN, an FI, and cloud computing data center, or within physical device such as a mobile device, GPS system, and a connected vehicle.

3.1 Use-case Scenarios: Smart Transportation Systems

As we briefly introduced in Section 2, our use-case scenario is based on Smart Transportation Systems (STSs). We first demonstrate the role of fog computing in supporting STSs. STSs are intelligent and adaptive systems that

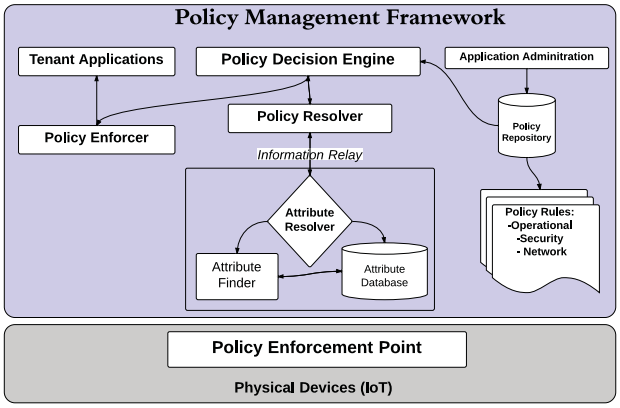

Figure 3. The Extended Policy Management

accommodate dynamic traffic changes and provide real-time traffic information to travelers by considering potential conflicts and safety issues. STSs consist of diverse interacting components and each component requests and provides multiple resources and data information: Smart Traffic Lights (STLs), Connected Vehicle (CV), Emergency Connected Vehicle (ECV), and pedestrian with smart devices. In the use-case scenarios in this paper, STLs play an important role as FIs by relaying communication data between FNs and other physical smart devices. STLs can be considered as a System of Systems of traffic lights in an urban transportation system. Based on these components, we consider the following use-case scenarios:

Scenario 1: Bob leaves for work at 7:30 AM and is required to reach his office by 8:00 AM. His office address is stored in the GPS system linked to his CV. When his CV approaches the first STL, it communicates with the STL and receives an optimum route to its destination based on the estimated time of arrival. As he approaches the next STL, based on the current traffic condition factoring in his intended arrival time, the system will update the GPS with the same or alternate route for the requested travel.

Scenario 2: While Bob is on his way to his office, a firetruck, which is categorized as an ECV, travels on the same route to respond a reported emergency. This ECV will provide the nearest STL with its final destination and by doing so, the corresponding STL will also update Bob's GPS to inform him of an approaching ECV so that he could either prepare to pull over or provide an alternate route to avoid the ECV.

Scenario 3: A school bus travels to the designated school, Tempe High School and on its route it makes multiple stops to pick up students thus affecting the traffic flow. The school bus will communicate its final

destination with the first STL and the STL will then notify all adjoining STLs of the approaching school bus. Since its route is pre-determined, STLs notify all connected vehicles of an approaching school bus and advise an appropriate precaution. The current route is the same as Bob's route and his GPS system will warn him of an approaching school bus.

Scenario 4: Bob exceeds the speed limit and is fast approaching an STL where a pedestrian is about to cross. The STL detects Bob's CV and notifies him via GPS of a probable collision detection. At the same time, the STL notifies the pedestrian of an approaching CV and updates its traffic information to alert adjacent STLs.

The above-mentioned scenarios require real-time data analytics and aggregation along with dynamic relays of information between the smart vehicles, traffic lights, FNs and FIs. To manage the secure collaboration and continuous connectivity between all interacting instances, a robust policy management framework is critical to ensure interoperability in a fog ecosystem as well as ensure secure communication among the different entities. To enable such complex collaboration and data sharing, we need to evaluate and resolve conflicts and anomalies dynamically whilst directing the requested information securely to its final destination. In addition, it is necessary to articulate the relevant policy requirements to support our scenarios. We categorize such requirements into three primary non-functional requirements:

1. Operational Requirements: focusing on enforcement of operational constraints for interacting components in a fog ecosystem to prevent misuse and any potential breach of unauthorized data.

2. Network Requirements: focusing on maintenance of secure communication channel, network load balancing, and network QoS requirements.

3. Security Requirements: focusing on authenticating and authorizing access requests between various fog components and smart devices as well as ensuring policy specifications are met for multi-tenant applications in the fog computing environments.

To support these requirements and to ensure that a uniform yet secure collaboration is maintained while communicating in dynamic and distributed environment, the policy specifications are defined and each specification also attempts to capture constraints associated with devices or instances in physical or virtual in the fog computing environments. The generalized specification leverages the following two schemas:

- *Data schema*: specifies a set of defined attributes associated with a physical or virtual component.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Document created by: Clinton Dsouza;
    Gail-Joon Ahn, SEFCOM-ASU -->
  <Specification-1 Target="STL1.0"
      Requester="CV01"
      Resource="Authentication-Device">
    <Attributes Authentication="X.509"
        UUID="CV01" GPS-Lat="33.4545"
        GPS-Long="-111.98787"
        Time="7:30:00pm">CV01</Attributes>
  </Specification-1>
  <Specification-2 Target="FN01"
      Requester="STL1.0"
      Resource="Authentication-User">
    <Attribute Security_Token="X.509"
        UUID="STL1.0" Location="Tempe,AZ"
        Time="7:30:01">STL1.0</Attribute>
  </Specification-2>
  <Specification-3 Target="FN01"
      Requester="FN02"
      Resource="Security-Data_Migration">
    <Attribute Security_token="X.509"
        UUID="CV01" Destination="Mesa,AZ"
        Origin="Tempe,AZ"
        Target_Subject="FN02"
        FN02:Security_Token="X.509"
        FN02:Destination="Mesa,AZ">
  </Attribute>
</Specification-3>
```

Figure 4. Data Schema Specification

- *Policy schema*: specifies a set of defined conditions associated with a requested action which are required to be fulfilled for the transaction.

The values obtained from the above mentioned schemes are used to determine the event, condition and action being performed by the smart components. Additionally, the diverse nature of the smart devices coupled with heterogeneous fog computing system requires multiple policy enforcement points. This implies that multi-dimensional specification of policies is needed. We further introduce another level of policy specification to address this issue in both Data and Policy schemas as follows:

- *Virtual specifications*: consisting of virtualized instances such as FNs, FIs and centralized cloud-based data repositories.

- *Physical specifications*: consisting of connected smart devices also known as Internet of Things.

To further support the realization and implementation of a robust policy management framework and its associated policy modules, we also adopt eXtensible Access Control

Markup Language (XACML) [9] to define a more formalized and refined operational, security and network policy specifications. Figure 4 shows a snippet of data schema specification.

4 Implementation and Evaluation

We have implemented a preliminary testbed which not only focuses on the policy decision enforcement but also simulates a STS based on the use-cases detailed in Section 3.1. The testbed utilizes Google Maps API for displaying alternative routes as well as real-time route of the user. On the user-end we designed a mobile application that collects user data within the time interval [every second to five minutes]. The collected data is stored in a database to be later used for data aggregation. When a user provides his final destination, the web-application determines a list of alternative routes apart from the user-chosen route. As the user approaches a STL, his route changes based on updated traffic information that the STL receives. The STL plays the role of a Policy Enforcer wherein all necessary policy decisions are routed to the STL for the enforcement on either a CV or a pedestrian. The policy management framework has been also implemented with key elements introduced earlier. To support policy enforcement capability in our implementation, we also utilized ARM-based computers called Raspberry-Pi, which can be emulated to behave as STLs and thus behaves as policy enforcement points. In addition, a generic policy decision engine is built based on OpenAZ API. The testbed consists of five main components:

- User UI : is the front-end display which a user can use to view his current position and get alerts on policy decisions being pushed to his device.
- Collection Data Container: is a data object which consists of road information such as traffic delays, road closures, and current road usage by emergency vehicle.
- Google Directions Services: are third-party services which are utilized to obtain multiple possible routes based on the data being collected in the collection data container.
- STLs: are semi-virtual FNs to provide computation, networking and storage capabilities to users and handle requested services.
- Connected Vehicle: is a smart vehicle completely controlled by the user which establishes secure communication with STLs and requests services.

Also, we utilize and configure the instances in a single OpenStack project to simulate FIs and host the web applications. The FIs can also behave as supportive services when the node instance is running out of memory.

4.1 Evaluation and Results

As mentioned above, our testbed is capable of evaluating one main component which we have deeply analyzed: the Policy Enforcement Point (PEP). The PEP is one of the primary components in the proposed policy management framework. To evaluate the functionality of the PEP in our testbed, we utilized the OpenAZ open-source implementation framework to analyze the security policies as shown in Table 1. The policies have been specified in XACML and were parsed through to extract attribute values before a decision is made. We measured the performance of the PEP based on the specified XACML policy set. Additionally, based on the use-case scenarios, we also captured vital information such as time taken by the vehicle to approach a STL, the distance of a vehicle from the STL, and the number of vehicle attempting to communicate with a single STL. For brevity, we partially describe our evaluation results focusing on scenario 1 mentioned in Section 3.1. The intuition behind our evaluation is to measure the performance of the PEP with the following two different conditions:

1. Light traffic load: simulates when a few vehicles are attempting to validate themselves with the STL.
2. Heavy traffic load: simulates when there are high volume requests from vehicles attempting to be validated themselves.

The complexity in loading each rule is created by the conditional statements defined in a single rule. For example, in Table 1, Rule 1 has a set of conditional evaluations for four attributes, while Rule 4 has a set of conditional evaluations for seven attributes which obviously increase the evaluation time at the PEP. Table 2 summarizes the evaluation results obtained from two security resources associated with the number of rules tested upon them. The ultimate goal of a fog system is to communicate decisions and service requests with users in real-time. Although our results show a near real-time policy enforcement case, given a situation wherein there might exist thousands of policy sets and corresponding rules, a definite lag can be expected. The lighter load of rules are executed in lesser amount of time than that with a higher load of rules.

5 Related Work

Due to its infancy, there exist very few related work in fog computing. In [6] and [7], Cisco research team introduced the concept of fog computing and its underlying architecture with concise definitions, and use-cases to

Table 1. Sample security rules

Rule #	Subject(s)	Resource (App)	Conditions(Attributes)	Requester	Action
1	STL1.0 | FN1.0 | STL2.0	Device Authentication	UUID: CV4898, Time: 7:30pm, Auth.: X.509 Token, Curr. Location: 243 Apache Blvd., Tempe,AZ	CV01	GRANT
2	FN-ab-01	User Authentication	UUID: STL1.0, Time: 7:30:01pm, Auth: X.509 Token, Curr. Location: 243 Apache Blvd, Tempe, AZ	STL1.0	GRANT
3	FN-ab-01	Instance Authentication	UUID: FI-ab-01, Time: 7:30:02pm, Auth: X.509 Token,Service_Request: Data_Migration	FI-ab-01	DENY
4	FN-hk-02 — FI-hk-02	Data Migration Authentication	UUID: FN-ab-02, Target UUID: FN-hk-1.2, Service_Requester: FN-ab-02, Time: 7:30:04pm, Location: Tempe, AZ, Auth: X.509 Token, Service_Request: Instance_Authentication	FN-ab-02	GRANT

Table 2. Evaluation Results

Security Resources	Light Load (10 - 30 rules) (ms)	Heavy Load (40 - 60 rules) (ms)
User Authentication	221 ms	259
Device Authentication	267 ms	329

promote the efficiency and necessity of such a dynamic platform. Also, Madsen et al. evaluated the fog computing platform from a very abstract perspective but have provided certain interesting evaluation criteria that a fog platform should meet such as M2M interactions and reliability protocols which a fog system should utilize [10]. However they did not present any conclusive implementation and evaluation results of any reliability tests.

In addition, there exist numerous approaches related to policy management in distributed computing environments including [11]. There have also been significant advances in the area of policy conflict detection and resolution in relation with network policy such as [4] [5] [12] [13]. The novel policy conflict and anomaly detection techniques coupled with the resolution strategies have been proposed in [13]. However, given the distributed nature of fog computing, there is a need to enhance existing policy management approaches for supporting such dynamic fog ecosystems.

6 Conclusion

In this paper we have outlined key characteristics of *Fog Computing* and have identified challenges in policy management that are critical for supporting secure sharing, collaboration and data reuse in a heterogeneous environment. We outlined a set of use-case scenarios and have shown the necessity of policy management as a core security management module in a fog ecosystem. Also, we have proposed a preliminary policy management framework accompanied with policy specification criteria and relevant schemas. In addition, we have demonstrated the feasibility and practicality of our approach through a proof-of-concept implementation of a fog computing environment based on use-case scenarios.

As part of future work, we will further address a sophisticated way to detect policy conflicts and resolve the detected conflicts. In particular, we will attempt to define a set of anomalies that should be addressed in fog computing environments. In addition, we will extend our policy management framework to support more complicated use-cases along with diverse devices so that we can measure the effectiveness of our approach with more realistic testbed.

Acknowledgment

This work is partially supported by grants from Cisco Inc. The authors would like to thank Dr. Rodolfo Milito at Cisco for his continuous support and valuable feedback in this project. In addition, we would also like to thank Jeong-Jin Seo at SEFCOM for his valuable contribution to the initial development of the IoT testbed that supports and simulates a STS environment.

References

[1] N. Bari, G. Mani, and S. Berkovich. Internet of things as a methodological concept. In *Computing for Geospatial Research and Application (COM.Geo), 2013 Fourth International Conference on*, pages 48–55, July 2013.

[2] Dave Evans. The internet of everything: How more relevant and valuable connections will change the world. Cisco IBSG, 2012.

[3] Robert De La Mora. Cisco iox: An application enablement framework for the internet of things. January 2014.

[4] A.A. Mansor, W.M.N.W. Kadir, T. Anwar, and S. Sahibuddin. Analysis of adaptive policy-based approach to avoid policy conflicts. In *Software Engineering Conference (APSEC), 2012 19th Asia-Pacific*, volume 1, pages 754–759, Dec 2012.

[5] Zhengping Wu and Yuanyao Liu. Dynamic policy conflict analysis for collaborative web services. In *Network and Service Management (CNSM), 2010 International Conference on*, pages 338–341, Oct 2010.

[6] Preethi Natarajan Flavio Bonomi, Rodolfo Milito and Jiang Zhu. Fog computing: A platform for internet of things and analytics. *Big Data and Internet of Things: A Roadmap for Smart Environments, Studies in Computational Intelligence*, 546:169–186, 2014.

[7] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[8] C.E. Rubio-Medrano, C. D'Souza, and Gail-Joon Ahn. Supporting secure collaborations with attribute-based access control. In *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on*, pages 525–530, Oct 2013.

[9] Erik Rissanen et al. extensible access control markup language (xacml) version 3.0. *Oasis Standard*, 2013.

[10] Henrik Madsen, Bernard Burtschy, G. Albeanu, and Fl. Popentiu-Vladicescu. *Reliability in the utility computing era: Towards reliable Fog computing*, pages 43–46. IEEE, 2013.

[11] L. Teo and Gail-Joon Ahn. Towards effective security policy management for heterogeneous network environments. In *Policies for Distributed Systems and Networks, 2007. POLICY '07. Eighth IEEE International Workshop on*, pages 241–245, June 2007.

[12] Hongxin Hu, Gail-Joon Ahn, and K. Kulkarni. Discovery and resolution of anomalies in web access control policies. *Dependable and Secure Computing, IEEE Transactions on*, 10(6):341–354, Nov 2013.

[13] Hongxin Hu, Gail-Joon Ahn, and K. Kulkarni. Detecting and resolving firewall policy anomalies. *Dependable and Secure Computing, IEEE Transactions on*, 9(3):318–331, May 2012.