Decentralized Access Permission Control using Resource-oriented Architecture for the Web of Things

Se Won OH*/**, Hyeon Soo KIM*

*Department of Computer Science & Engineering, Chungnam University, Yuseong, Daejeon, 305-764, Korea

** Electronics and Telecommunications Research Institute, Yuseong, Daejeon, 305-700, Korea

sewonoh@etri.re.kr, hskim401@cnu.ac.kr

Abstract— **As the today's Web provides open communication environment for a variety of web resources, the Web of Things (WoT) offers new opportunity and challenges about the interoperation among the smart things. The well-known Web technologies can leverage the Web-enabled things to publish and exchange their resource information over the Web, then the Web-enabled thing should cope with the security threat regarding the information exposures over the Web, particularly, access permissions to the thing's resource information. Thus, in this paper we analyse access permission control mechanism considering both the WoT characteristics and the REST-compliant resource-oriented Web architecture. In contrast to existing access control logics, the proposed mechanism utilizes not only the requester information such as the typical identity and the internet addresses, but also the context of the thing itself. Based on this mechanism, we present web-resource structure for access permission control, and describe an exemplary procedure in detail. This research contributes to the flexible and decentralized access permission control for WoT.**

Keywords— **WoT, IoT, Access Control, Access Permission, ROA, REST, Resource-oriented**

I. Introduction

Nowadays the number of smart things connected to the Internet even exceeds the population of human beings. As more smart things are capable of data communicating over the Internet, the concepts of the "Internet of Things (IoT)" or "Machine-to-machine (M2M)" are having been realized in many fields, as in [1]–[5], including smart homes, smart grid, remote healthcare, and logistics automation. Different sources predict that by 2020, as in [6] and [7], the number of connected things would be up to 212 billion, and the market size will be almost $9 trillion.

The vision of IoT depicts the interoperation among these networked devices as well as the global connectivity to the physical things in the real world. Some research activities, however, have mainly focused on establishing connectivity among these networked devices, for example, about how to make the embedded devices with an IP address interconnected on today's Internet. Whereas the Web-based IoT or the "Web of Things (WoT)" have been extensively studied for integrating smart things with the well-known Web technologies, as in [8]–[10]. The Web-enabled things can interoperate freely over the Web utilizing the open Web standards, since the World Wide Web (WWW, Web) can provide flexible, scalable communication channels for any web resources and web clients. Also, the scope of the traditional web services can be broadened into the physical-world, not only cyber-world. Moreover, the Web-enabled things can reuse and adopt the proven Web mechanisms such as discovery, searching, browsing, linking, and caching, as in [11]. Further, REST (Representational state transfer) architectural style with URIs (Uniform Resource Identifier), HTTP, and standardized media types, is very promising to make these things to share their data and resources over the Web.

A pressing open problem in this WoT environment is how to allow smart things to grant clients access to their own resources, since the Web-based open environment often leaves information vulnerable to disclosure resulting in security threat such as:

- Malicious clients and unwanted data sharing
- Attacks in any time and from anywhere
- Unpredictable work load and availability risk

Unfortunately, there are few studies on this issue and several system prototypes only provides inadequate functionality and security. In particular, from a security point, the existing access control mechanisms for the web resources, as in [8], [10], [12], requires either rigid access policy enforcement or accommodation of external access management procedures like CAS (central authentication service) system.

Thus, we generalize the notion of access permission control for the resources of web-enabled things in this paper. Then, we propose a decentralized access permission control mechanism for Web resources for WoT. The proposed mechanism adopts REST-style resource-oriented architecture for things, in order to enable a thing itself (or its owner) to manage access permissions to its own information resources by means of simple CRUD (Create, Read, Update, and Delete) actions. To explain the mechanism in detail, we describe the prototype module of resource access control as well as resource structure XML.

The remainder of this paper is organized as follows: Section 2 introduces the related study for this research. Section 3 proposes access permission control mechanism considering both the WoT characteristics and the resource-oriented Web architecture with the exemplary cases in detail. Then, concluding remarks are provided in Section 4.

II. Related Study

Here we summarize the basic model of access control model and the existing researches on access permission management, with describing some challenges coming from the WoT characteristics.

A. Access Control Model

The very nature of access control suggests that there is an active subject requiring access to a passive object to perform some specific access operation, while a reference monitor between them grants or denies access, as shown in Figure 1.

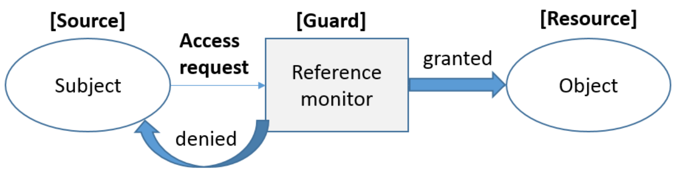


Figure 1. Access control model for providing computer system security

To certify the security level of information system, it is essential to control operations executed by subjects for preventing actions that could damage data and resources [12]. The reference monitor as a guard performs two operations: 1) to identify the subject who made the request, and 2) to decide who is allowed to do what to the object.

Most existing implementations for access control have been dependent on a kind of local database which can maintain either user passwords, group membership, and/or access control matrices. A few differences for security on the Web just includes 1) SSL usage for securing the user channels, 2) checking host information for authentication, and 3) centrally managed database for a domain access control.

To assure access control in a distributed system environment like the Web, it is necessary to handle all the information from disparate systems managed locally in past days. Likewise, the number (and diversity) of subjects/objects in the web of things make this access control issue more challenging.

B. Existing Access Control Logics in Computer Network

With the advancement of information security, in order to facilitate the management of access permission, many security model for computer network systems have been successfully coming up till now as the followings, to name just a few;

- Subject/object access control matrix [13]
- Multilevel security using information flow [14]
- Role-base access control (RBAC) [15]
- Attribute-based access control (ABAC) [16]

In particular, RBAC introduces the concept of role, in order to logically relate users and permissions. That is, the RBAC model assigns permissions to roles, and roles to the users. However, these model mainly focuses a kind of 'static' authorization assignment, while many practical information systems requires more flexible ways to update the subject's permissions. Thus, in consideration of dynamic changes in group structure, attributes of the subject, and trust level, recently a concept of dynamic authorization model has been suggested [17]. But any of the traditional research has yet specified access control logic for considering the context of the requested object.

III. Access Permission Control for WoT

A. WoT Access Control Requirements

In the WoT environment a thing is expected to communicate with other thing over the web eventually by itself. That is, each thing may be either subject or object interchangeably in order to perform their own tasks. Further, according to the REST-style web architecture, each thing can represent itself as web resources which can be identified with the unique URIs. According to the aforementioned issues, the access permission control for WoT should have the following requirements:

1) Each thing may publish its information as one or more web resource(s) over the web

2) Resources for a thing can be accessed by the basic HTTP/REST request from a subject (e.g. a HTTP client).

3) Permission assignment can be represented as a kind of resource, then the decision to grant access for a given request to the specific resource should be made using this resource.

B. Resource-oriented Architecture for WoT

Figure 2 presents the types of resource representation model and the basic four HTTP/REST methods for the CRUD operations to the REST resources. As shown as a) a resource has a unique URI and attribute(s) information, and a group of resources can be also represented as a kind of resource as outlined in b) and c).

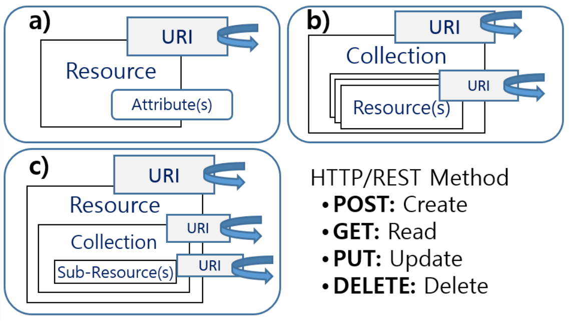


Figure 2. REST resource representation and HTTP method types

C. Proposed Mechanism for Managing WoT Access Control

Here we proposed 5 steps to assure any resource access request from a subject as summarized in Table 1. Particularly, the issue of how to decide if the resource access is granted or not is very closely associated with the Step 5. While we presume the advanced secure channels and cryptographic algorithms would improve the security level for the HTTP protocol communication, we are focusing the basic standard HTTP protocol in this research.

TABLE 1. STEPS TO ASSURE RESOURCE ACCESS CONTROL

Order	Step	To filter out
1	To filter TCP/IP packets	Unallowable domain access
2	To parse HTTP/REST request	Invalid request; abnormal parameters
3	To check HTTP header for basic authentication	Unverified client
4	To check whether the requested resource exists or not	Requests for the expired/outdated or irrelevant resource
5	To check the assigned access permission for the request operation	Unassigned access permission for the requested operation

Figure 3 describes an exemplary case for a HTTP/REST requests using the typical access control matrix, where a thing may have N REST resources (objects) which can be requested from the other M things (subjects). While the access permissions about Object 1 might be simply designed for each subject (1 to M), this model do not provide scalable function due to the overhead for maintaining and searching in the M by N matrix. Additional classification of subjects into the several associated groups based on Role-based Access Control (i.e. the role of SuperUser equals {Subject 1, Subject 2, …}) might alleviate the burden, but each thing still needs to maintain the lists of roles, and compare the assigned access right from each REST requests for CRUD operations.

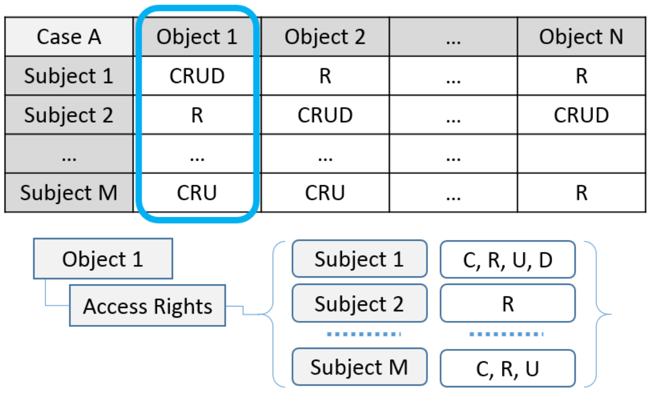

Case A | Object 1 | Object 2 | … | Object N
Subject 1 | CRUD | R | … | R
Subject 2 | R | CRUD | … | CRUD
… | … | … | …
Subject M | CRU | CRU | … | R
Object 1
Access Rights
Subject 1
C, R, U, D
Subject 2
R
Subject M
C, R, U

Figure 3. Access matrix example focusing on Object 1

Then, we slightly modify the matrix of Figure 3 in order to concentrate with the each CRUD operations as shown in Figure 4. And here we got insights that a specific permission policy can be applied to more than one REST requests, for instance, Create and Update for the Object 1. Moreover, we now generalize the permission policy by replacing the subjects with the generic conditions which accommodate not only the subject information like typical identifiers for the requester (i.e. IP address, hostname, and domain information), but also the context information about the thing itself (i.e. time duration, point of time, location, device hardware state, capabilities, configuration limit value).

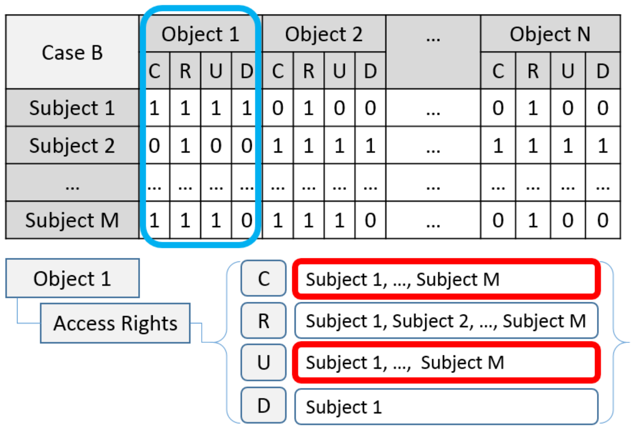

Case B | Object 1 | Object 2 | … | Object N
C R U D | C R U D | C R U D
Subject 1 | 1 1 1 1 | 0 1 0 0 | … | 0 1 0 0
Subject 2 | 0 1 0 0 | 1 1 1 1 | … | 1 1 1 1
… | … … … … | … … … … | … | … … … …
Subject M | 1 1 1 0 | 1 1 1 0 | … | 0 1 0 0
Object 1
Access Rights
C
Subject 1, …, Subject M
R
Subject 1, Subject 2, …, Subject M
U
Subject 1, …, Subject M
D
Subject 1

Figure 4. Access matrix example focusing on the each CRUD operations

Now, we can represent an access rights for each CRUD operation to Object 1 as the following:

Permission Flag C[/R/U/D] = {Condition about Subjects} ∩ {Condition about the requested Object / Thing}

For example, a thing may allow the 'Create' operation by judging from the combinative results throughout the several conditions as specified in Table 2; C1) if the requester is from the specific domain like '*.cnu.ac.kr', and C2) if the current time is between 9 AM and 3 PM, and C3) if the CPU usage of the requested thing is currently less than 90%.

TABLE 2. EXAMPLE OF MULTIPLE CONDITIONS ON ACCESS CONTROL

Condition	Type	Value
C1	(Subject) Domain	*.cnu.ac.kr
C2	TimeBetween	(09:00:00, 15:00:00)
C3	(Thing) State	(CPU, LESS-THAN, 90%)

Here, we suggests that any implementation should settle the rules when there is more than one condition. And the basic cases would be from the following two cases with the assumption that all the conditions has the equal priority level:

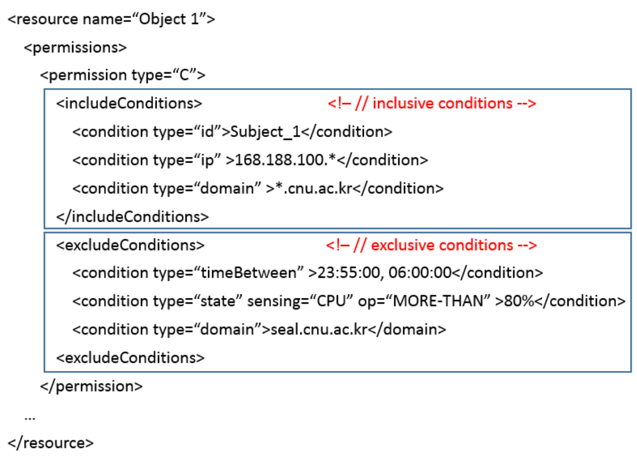
- Permission is granted only when all the conditions meet
- Permission is granted if any of the conditions meets

Further, for more elaborated access control, the conditions can be divided into two types: inclusive and exclusive. In this research, we chose the rule: Permission is granted if any of inclusive conditions meets when any of exclusive condition does not meet. Then, the above expression can be represented:

Permission Flag C[/R/U/D] = {Union of inclusive conditions} - {Union of exclusive conditions}

D. Representation of WoT Access Control Resource

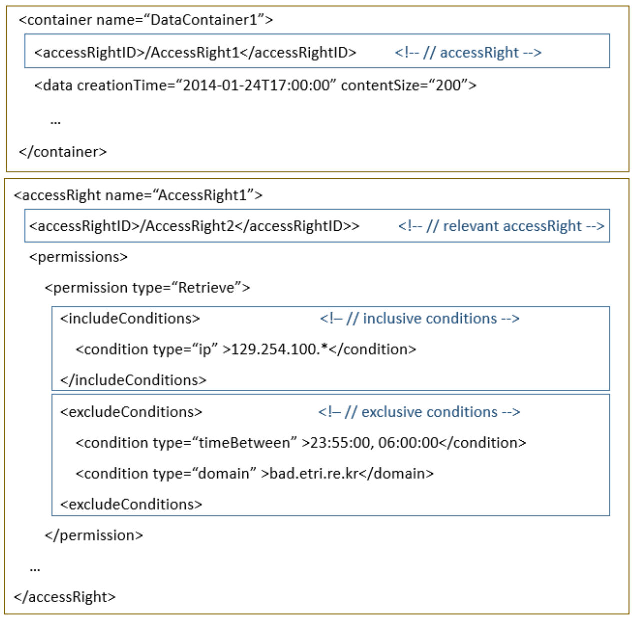
The Figure 5 outlines the permission policy about 'Create' operation request to Object 1, which consists of 3 inclusive and 3 exclusive conditions. Then, Object 1 can be accessed only if any of 3 inclusive conditions (i.e. Subject id, Subject IP address, Subject domain), meet when any of 3 exclusive conditions (i.e. Thing operation duration, Thing CPU usage, Subject domain) do not meet. Likewise other three operation (i.e. R/U/D) can be represented as the part of the permission policy.



```
<resource name=“Object 1”>
  <permissions>
    <permission type=“C”>
      <includeConditions>                    <!-- // inclusive conditions -->
        <condition type=“id”>Subject_1</condition>
        <condition type=“ip” >168.188.100.*</condition>
        <condition type=“domain” >*.cnu.ac.kr</condition>
      </includeConditions>
      <excludeConditions>                    <!-- // exclusive conditions -->
        <condition type=“timeBetween” >23:55:00, 06:00:00</condition>
        <condition type=“state” sensing=“CPU” op=“MORE-THAN” >80%</condition>
        <condition type=“domain”>seal.cnu.ac.kr</domain>
      <excludeConditions>
    </permission>
  …
</resource>
```

Figure 5. Representation of access permission for Create operation

Furthermore, the above permission policy also can be represented as one of web resource (Object), which will make same policy be utilized to the multiple resources readily. Figure 6 show two web resources, the above one is for DataContainer1 and the below one is for AccessRight1 which is used for access policy of DataContainer1 just with referencing AccessRight1's URI. The AccessRight1 also can be managed by another access policy using AccessRight2. Eventually, only with these resource models, a thing can manage access permission without any external authorization server's support. Depending on the implementations of WoT, a specific access policy can be consistently applied to the multiple things, since this web-resource access policy can be easily referenced by its URI.



```
<container name=“DataContainer1”>
  <accessRightID>/AccessRight1</accessRightID>      <!-- // accessRight -->
  <data creationTime=“2014-01-24T17:00:00” contentSize=“200”>
    …
</container>

<accessRight name=“AccessRight1”>
  <accessRightID>/AccessRight2</accessRightID>>      <!-- // relevant accessRight -->
  <permissions>
    <permission type=“Retrieve”>
      <includeConditions>                    <!-- // inclusive conditions -->
        <condition type=“ip” >129.254.100.*</condition>
      </includeConditions>
      <excludeConditions>                    <!-- // exclusive conditions -->
        <condition type=“timeBetween” >23:55:00, 06:00:00</condition>
        <condition type=“domain” >bad.etri.re.kr</domain>
      <excludeConditions>
    </permission>
  …
</accessRight>
```

Figure 6. Representation of WoT access control resource

IV. Conclusions

In this paper, we specified the requirements for the access permission control to the resources of web-enabled things. Then, we propose an access permission control mechanism considering for the WoT characteristics. The proposed mechanism adopts resource-oriented architecture for Web-enabled things, utilizes the access conditions regarding both the requester information and the context of the thing itself.

The access permission policy can be easily applied to multiple resources and things, since the policy can be represented also as a web resource. Moreover, there is no need to have other protocol for adjusting access policy. Thus, the proposed mechanism contributes to the flexible and decentralized access permission control for the Web of Things.

Acknowledgment

This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korea government [13ZC1100 (2013~2015) – Unit Research Project 3, Development of USN/WoT Convergence Platform for Internet of Reality Service Provision, 14ZC1310].

References

[1] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac. “Internet of Things: Vision, Applications and Research Challenges,” *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, Sept. 2012.

[2] L. Atzori, A. Iera, and G. Morabito. "The Internet of Things: A Survey," *Computer Networks*, vol. 54, no. 15, pp. 2787– 2805, Oct. 2010.

[3] I.G. Smith, Ed., *The Internet of Things 2012 - New Horizons*, Internet of Things European Research Cluster (IERC) Cluster Book, 2012. Available: http://www.Internet-of-things-research.eu/pdf/IERC_Cluster_Book_2012_WEB.pdf

[4] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K.D. Johnson, “M2M: From Mobile to Embedded Internet,” in *IEEE Communications Magazine*, vol. 49, no. 4, pp. 36–43, 2011.

[5] E. Lee, H. Lee, K. Lee, and J. Park, “Automating Configuration System and Protocol for Next-Generation Home Appliances,” *ETRI Journal*, vol. 35, no. 6, pp. 1094-1104, Dec. 2013.

[6] (2013) IDC - Press Release. [Online]. Available: http://www.idc.com/getdoc.jsp?containerId=prUS24366813

[7] (2013) CISCO - Internet of Things. [Online]. Available: http://www.cisco.com/web/solutions/trends/iot/overview.html

[8] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, "From the Internet of Things to the Web of Things: Resource-oriented Architecture and Best Practices," in *Architecting the Internet of Things*, pp. 97– 129, Springer, 2011.

[9] S. Duquennoy, G. Grimaud, and J.-J. Vandewalle, “The Web of Things: Interconnecting Devices with High Usability and Performance,” in *International Conference on Embedded Software and Systems (ICESS)*, 2009, pp. 323–330.

[10] D. Zeng, S. Guo, and Z. Cheng, “The Web of Things: A Survey,” *Journal of Communications*, vol. 6, no. 6, pp. 424-438, Sept. 2011.

[11] *Framework of the web of things*, ITU-T Y.2063, 2012.

[12] B.W. Lampson, “Computer security in the real world,” *Computer*, vol. 37, no. 6, pp.37-46, June 2004.

[13] B.W. Lampson, “Protection,” *ACM SIGOPS Operating Systems Review*, vol. 8, no. 1, pp. 18-24, Jan. 1974.

[14] A.C. Myers and B. Liskov, “A decentralized model for information flow control,” *ACM SIGOPS Operating Systems Review*, vol. 31, no. 5, pp. 129-142, Dec. 1997.

[15] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman, “Role-based access control models,” *Computer*, vol. 29, no. 2, pp.38-47, Feb. 1996.

[16] M.A. Al-Kahtani and R.S. Sandhu, “A Model for Attribute-Based User-Role Assignment,” in *Proc. 18th Computer Security Applications Conference*, 2002, pp. 353-362.

[17] J. Liu, C. Liu, D. Jiao, and J. Chen, “The Research of a Multi-Factor Dynamic Authorization Model,” in *IEEE 9th International Conference on e-Business Engineering (ICEBE),* 2012, pp. 201-205

Se Won OH is a senior member of engineering and research staff working for ETRI, and also under a PhD course at Chungnam National University (CNU), Korea. He received He received the BS (1999) and the MS degree (2001) from Pohang University of Science and Technology (POSTECH), Korea. Since 2001, he has been involved in several research projects on software platform (such as RFID Event Management System, USN Middleware Platform, Software System Infrastructure) which integrates legacy applications with various data resources including RFID tags and Sensor Node. His recent research interests are Internet of Things (IoT) and Web of Things (WoT), particularly about handling Web-enabled devices. He has also participated standardization activities on automatic identification and data capture (AIDC) techniques as a member of JTC 1/SC 31, and as a secretary for JTC 1/SC 31/WG 6 (Mobile Item Identification and Management). He is a member of KICS.

Hyeon Soo Kim is a professor at Chungnam National University (CNU), Korea. He works for Department of Computer Science and Engineering at CNU. His current research areas include Software Engineering (Software Testing, Software Architecture, Software Maintenance, and Software Reengineering) and Applications (Nuclear Engineering) as well as Distributed Computing (J2EE/EJB, .NET, Web service, SOA, Internet of Things, and Large Scale Data Processing). He received the BS degree (1988) from Seoul National University (SNU), Korea, and the MS (1991) and the PhD degree (1995) from Korea Advanced Institute of Science and Technology (KAIST), Korea. He is a member of KIISE and KIPS.