International Journal of Applied Engineering Research ISSN 0973-4562 Volume 11, Number 24 (2016) pp. 11913-11924

Flexible Access Control for Complex Business Application: A Case Study

Yonghan Lee[1] and Young-Gab Kim [2*]

[1] Global Business Services, IBM Korea,
Three IFC, 10 Gukjegeumyung-ro, Yeongdeungpo-gu, Seoul 07326, Republic of Korea.

[2] Department of Computer and Information Security, Sejong University,*
209, Neungdong-ro, Gwangjin-gu, Seoul 05006, Republic of Korea.
(Corresponding author)

Abstract

As computing environments have grown in size and complexity, business service also have become more diverse and complex. Therefore, in terms of information security, the access control become an important component of complex business applications. Unlike traditional access control approaches (e.g., MAC, DAC, and RBAC), access control decisions should take into consideration the context information such as time and location. As a result, a fine-grained access control mechanism that dynamically changes the privileges of a service based on the context information is required in a complex business applications. In this paper, we propose a model called CIAAC (Context Information-based Application Access Control), which is designed to separate context awareness and access control policies from business and processing logic. Moreover, we focus on the case studies which show how to apply the proposed approach to complex business applications, especially in the banking system.

Keywords: Access Control, Context-Aware, Business Logic, Complex Business Application

INTRODUCTION

As computing environments have grown in size and complexity, the problem of access control becomes more acute. Over the years, a number of access control mechanisms and models, like discretionary access control (DAC), mandatory access control (MAC), and role-based access control (RBAC) have been proposed [1, 2] and successfully adapted to various systems, including operating systems, document management systems, and large information systems. With the recent rise of ubiquitous, grid, and cloud computing systems, the need for more power access control models has become acute. To address this need, new research has focused on the use of "context" to define access control policies more easily and fully.

Briefly defined, "context," from the perspective of access control is an instance of a well-formed schema of subjects and objects. Here, objects are operations and static resources, and subjects are agents that want to use the objects. Because generally time and location are important properties of context, context can vary with the course of time and changing location. Further, it is a crucial task how to use captured context information for access control policies and rules. Therefore, a previous work [3] proposed a novel model named context information-based applicaton access control (CIAAC) model, which can provide access previlige for software applications with complex business logics based on context information. CIAAC is specifically designed to separate context awareness and access control policies from business and processing logic, allowing operators of business applications to change access control policies more freely in response to the external security environment. However, the previous work was only focused on the description of CIAAC

model with simple exampes. Therefore, in this paper, in order to show the effectiveness of the CIAAC model, we focus on a modeling and implementation of the CIAAC model with three case studies. In particular, we shows how context information can be integrated into software applications especially in banking system such that those applications maintain flexible access control.

This paper is organized as follows. Section 2 describes related studies of access control using context information and associated restrictions in implementing such access control in software applications. Section 3 describes briefly the CIAAC model proposed in previous work. Section 4 provides a case-study for the proposed model, illustrating the use of flexible access control in a banking application. Finally, Section 5 provides some concluding remarks.

RELATED WORK

Traditional Access Control Models

There are several general models for access control. One of the most common of these is known as DAC, as implemented in access control matrix (ACM) and Take-Grant systems. DAC defines subjects' access to objects as a set of relations that configured by the owners of the objects [1, 2]. Another prevalent general model is MAC, as implemented in Chinese-Wall and Bell-LaPadula (BLP) systems. In MAC, owners of objects do not configure relations between subjects and objects; rather, supervisors or super users define the access level of each subject and the producers or owners of objects define the sensitivity of those objects at each level [2, 4]. Both DAC and MAC lack flexibility in certain regards. For example, when large numbers of users are added or removed at a large enterprise organization, or when the organization shifts its access standards, there is no simple or sure way to update access controls under DAC or MAC.

An alternative to the DAC and MAC is known as RBAC. RBAC introduces the elements user, role, and permission, and policies are defined by mapping information between users and roles (user assignment, or UA) and between permissions and roles (permission assignment, or PA). This makes the RBAC model more flexible than DAC and MAC in configuring the access control policies of complex organizations [5], but also creates certain weaknesses. Suppose, for instance, that an emergency requires that we forcibly restrict service for requests meeting specific conditions, i.e., those coming from internet banks. You cannot define access control policies with just RBAC model. Further, it is difficult to define access control policies based on business requirements just like "deposit and withdraw service of teller terminals is available during only business hours" with generic RBAC model.

Overall, DAC, MAC, and RBAC all define access control policies using direct or indirect relations between subjects and objects.

Extended Access Control Model Using Context Information

Recent work has introduced the notion of "context" to access control models in order adapt to new environments, such as those involved in ubiquitous and cloud computing. Traditional RBAC models, in particular, have been extended to use context in different ways. Covington et al. [6] proposed the GRBAC (Generalized RBAC) model, which adds a context-oriented environment role to the familiar subject and object roles. Similarly, Park et al. [7] added a context role, which is mapped with organizational role to traditional RBAC model, and made use of context information to RBAC model. Lastly, Kim et al. [8, 9] proposed a new element, the SCM (State Checking Matrix), which to provide traditional RBAC with a validation mechanism based on context information about user-role and role-permission mappings.

These extended RBAC models allow greater flexibility in defining access control policies, but their continued reliance on the role and assignments of the RBAC model precludes a proper modeling of context and a consequent redesign of access control.

CONTEXT INFORMATION-BASED APPLICATION ACCESS CONTROL (CIAAC) MODEL

In this section, the CIAAC model proposed in the previous work [3] is explained briefly. The main goal of the CIAAC model is to provide access control model for software applications with complex business logics based on context information. As depicted in Figure 1, the CIAAC model is composed of three main components: context definition, access control policy, and access control controller (AC). A more detailed description will be presented in the following subsections.

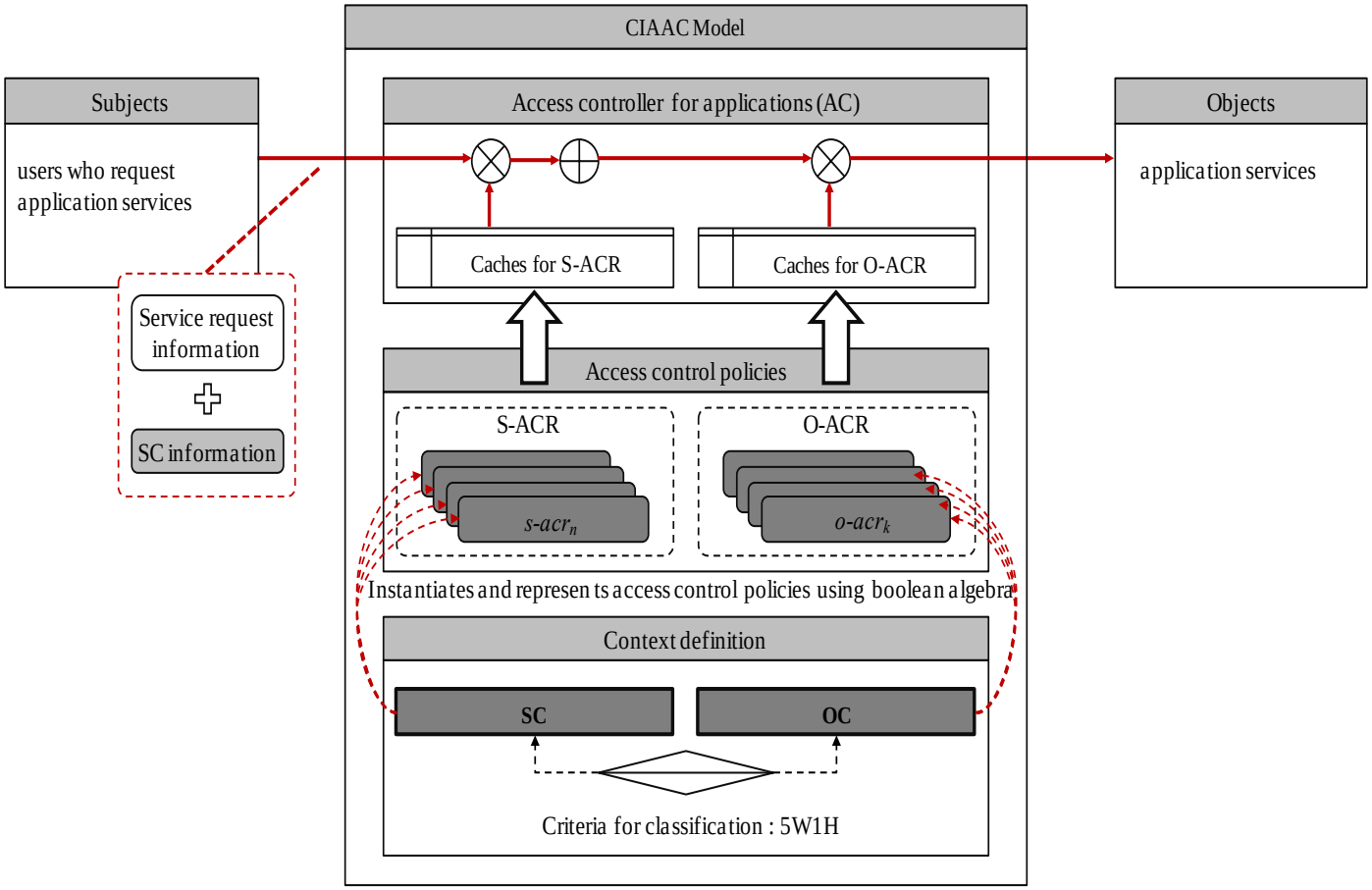

Figure 1: CIAAC Model for complex application

Context Definition

In the CIAAC model, context can be split into two parts: subject context (SC) and object context (OC). SC is the context that includes the subject and any information on the subject's surroundings that the subject should submit when requesting access to an object. Similarly, OC is the context that includes the object and all properties of the object that represent its business logical orientation.

We must establish context criteria that are mutually exclusive and collectively exhaustive. For this, we propose the familiar 5W1H (who, when, where, what, why, and how) as good criterion for context modeling. Table 1 shows candidate attributes for SC and OC based on 5W1H criteria, especially in banking systems.

Table 1: Candidates for SC and OC attributes based on 5W1H criteria

Criteria	Candidates for SC Attributes	Candidates for OC Attributes
Who	• Requester's ID • Requester's position in company • Requester's department	• Requester's department
When	• Date to request • Time to request	• Accessible on holiday • Accessible after working time
Where	• Requester's IP address	• Accessible IP address
What	• Transaction ID to request	• Transaction ID
How	• Channel ID	• Accessible channel
Why	• Normal or cancelling	• Possible to cancel

Access Control Policy

An access control model must define the policies to decide whether a specific subject can access a specific object. Whereas other access control models use methods like ACM, ACL, roles, and views, our CIAAC model defines policies as Boolean algebra expressions. In the CIAAC model, policies defined by expressions based on SC are called subject access control rules (S-ACR) and those based on OC are called object access control rules (O-ACR). Once access policies are defined, the access control management system can decide, using context information submitted by the subject, whether the access request is permitted.

Access control rules are crucial to the CIAAC model, as they must replace the representation of permissions as relations between subjects and objects. Hence, S-ACR and O-ACR clearly, and must be easily implemented. Like ACL and ACM, CIAAC uses a 2-dimesional matrix to define S-ACR and O-ACR as depicted in Figures 2 and 3. S-ACR is a set of policies to constrain a subject who lacks access rights from accessing an object, either temporarily or permanently.

And Condition

	S-ACR	SC1	SC2	SC3	SC4	SC5	SC6	SC7	SC8	SC9
		Requester ID	Requester Position	Department ID	Date	Time	IP	Transaction ID	Channel ID	Cancelling
Or Condition	$s\text{-}acr_1$	[.]*	[.]*	[.]*	[.]*	[.]*	[.]*	SVC1101	ATM	[.]*
	$s\text{-}acr_2$	9	[.]*	[.]*	[.]*	[.]*	[.]*	[.]*	[.]*	[.]*
	$s\text{-}acr_3$	[.]*	[.]*	[.]*	20150930	19 |2[0-3]	[.]*	SVC1102	[.]*	[.]*

Figure 2: Example of S-ACR in banking system

And Condition

O-ACR	Transaction Id	Cancelling Possible	Department			Accessible Time		Accessible Channel		
			Sales	General affair	R&D	Holiday	After working time	Teller Terminal	ATM	Internet Banking
	-	OC1	OC2	OC3	OC4	OC5	OC6	OC7	OC8	OC9
$o\text{-}acr_1$	SVC1101	Y	Y	N	N	N	Y	Y	Y	Y
$o\text{-}acr_2$	SVC1102	Y	Y	Y	Y	Y	Y	Y	N	N
$o\text{-}acr_3$	SVC1103	N	Y	Y	Y	Y	Y	Y	N	N
$o\text{-}acr_4$	…									

Figure 3: Example of O-ACR in banking system

Given the access policy for each SC denoted as a regular-expression pattern matching function *m*, once we denote each context value the subject submits as $cv_1, cv_2, cv_3, \ldots, cv_n$. We can also denote each context policy in S-ACR as $scp_1, scp_2, scp_3, \ldots, scp_n$. The resulting matching function *m* can be represented as follows:

- $m_n = f\ (cv_n, scp_n)$: pattern matching function of *n*-th context
- Let the following denote the result of the *m* functions related by the *AND* operation:
- $s\text{-}acr_n = and\ (m_1, m_2, m_3, \ldots, m_n)$: each S-ACR
- S-ACR will then be the result of the *OR* operation applied to the above:
- $S\text{-}ACR = or\ (s\text{-}acr_1, s\text{-}acr_2, s\text{-}acr_3, \ldots, s\text{-}acr_n)$: entire S-ACR

As for the O-ACR, because object rules may reflect business logic, business logic developers mainly define O-ACR in isolation with business logic. Generally, O-ACR should keep going permanently before changing business requirements. Each access policy for each object context should be implemented using different method (o_n) each other. So each access policy for subject requests with subject context can be represented as follows:

- $o_n = f_n\ (cv_1, cv_2, cv_3, \ldots cv_m)$ where *n* is number of object context and *m* is number of subject context.

ACR will then be the result of the AND operation applied to the on above:

- $ACR = and\ (o_1, o_2, o_3, \ldots, o_n)$

Access Controller

Based on S-ACR and O-ACR above, the AC can properly authorize service requests. There are three components to the AC: a parser for context information submitted by the subject as part of the service request, a rule checker for S-ACR, and a rule checker for O-ACR.

Once a context data format (e.g., XML, fixed-length byte arrays, name-value pairs, etc.) is chosen, we can implement that can efficiently extract the value of any piece of information. If we choose an XML format, the parser will not need to know the type of each context value in advance, since XML can carry type information alongside values. If we choose fixed-length format, on the other hand, we will have to supply type information (e.g., string, number, length of decimal point, etc.) from without.

The checker for S-ACR can be implemented easily. Once context information is parsed into values, the checker merely asks whether a designated context value pattern-matches the context policy (i.e., regular expressions) in S-ACR. Again, note that for a single request, each S-ACR rule, and if just one is matched, the request is blocked.

It is difficult to generalize implementation of the checker for O-ACR, since the O-ACR is part of business logic and one

object context policy corresponds to several pieces of context information. Furthermore, a single request may require any number of objects, and thus cover a large amount of business logic. Unlike S-ACR, the checker for O-ACR uses just one O-ACR to validate the request, so if only requester's context information meets with the O-ACR of requested service, the request is allowed to access requested service.

The AC should be designed to process authorization as fast as possible, regardless of the size and complexity of S-ACR and O-ACR. Using a memory cache to store S-ACR and O-ACR policies at runtime helps to speed the checking processes. To obtain a detailed description of the CIAAC model, please refer to [3].

CASE STUDY IN BANKING SYSTEM

In order to illustrate the motivations for our research, the CIAAC model is applied to the banking system, which has many service channels. First, we will explain how the banking system architecture as a whole is structured and where the CIAAC fits into it. Second, we will discuss several complex banking scenarios that require access control, and demonstrate why CIAAC is helpful in these scenarios.

General Architecture of Banking System

The overall architecture of a banking system is illustrated in Figure 4. A banking system has many service channels, split into internal and external channels according to the agents of management. Internal channels are managed by the bank itself and include teller terminal, automated teller machine (ATM), internet banking server, and others. External channels are managed by organizations other than the bank. Requests coming in from these various channels are received by a multi-channel integration (MCI, also known as a front-end processor, or FEP) system, which splits them into internal MCI and external MCI. The main system for business processing is known as the "core banking" system. This system comprises a number of framework-based applications, supporting things such as deposits, loans, foreign exchanges, external interfaces, etc. Business intelligence is meant to analyze the processing results of the core banking system. Extract-Transform-Load (ETL) and Change Data Capture (CDC) transfer transaction results from core banking to the business intelligence area. Finally, the banking system will likely include many internal legacy business systems with enterprise application integration (EAI) connections between them.

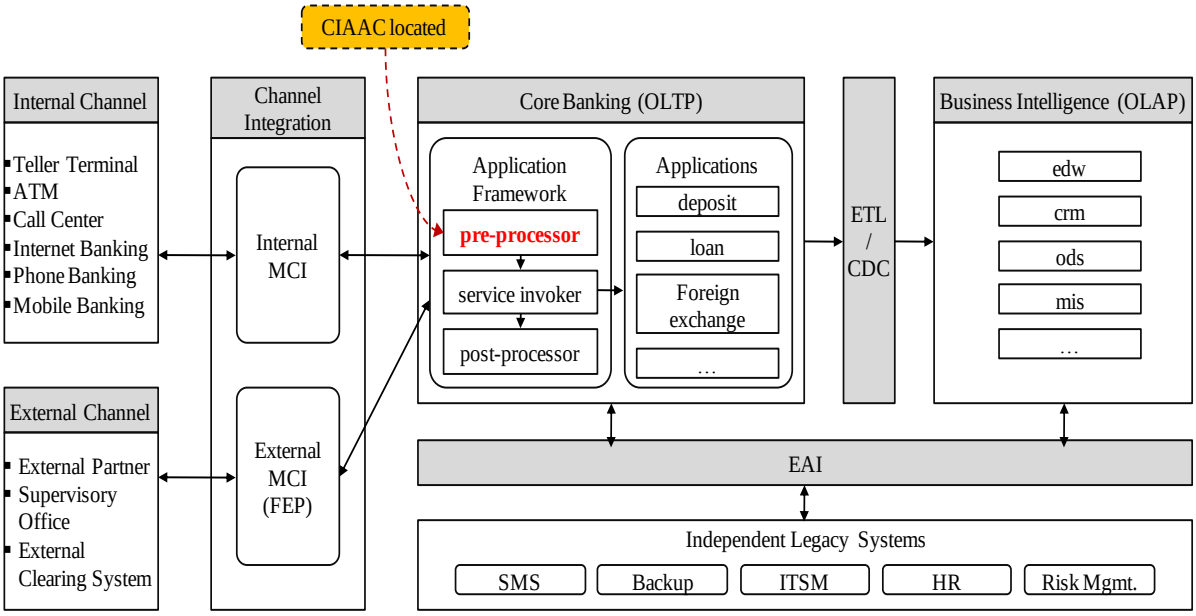

Figure 4: Typical architecture of a banking system

Location of the CIAAC module in Banking System

Modules for access control should be located at a point that all service requests must pass through. The channel integration area is such a position, but as shown in Figure 4, when channel integration systems are separated into two subsystems (i.e., internal MCI and external MCI), access control modules must be applied to each integration system independently, thereby splitting central policy authority.

Most application frameworks have filtering areas that perform common pre-processing and post-processing for all service requests. The pre-processor executes a pre-defined filter module, and the post-processor executes filter modules after processing. That is, in this paper, the pre-processor is selected as an ideal location for access control, as both internal and external channels are routed through this area.

Context Definition for Banking System

The major benefits of the CIAAC model are ease, intuitiveness, and flexibility, but good context modeling is needed to take advantage of these benefits. Obviously, the context must include all elements needed to determine access control, but it should also be kept as small as possible, to keep performance up. Eliminating duplicate elements and/or reusing elements where possible can help in this regard.

As previously mentioned, CIAAC context is split into SC and OC. When a customer of a bank requests a cash withdrawal over ATM, the ATM transfers the service request to the banking system with pre-defined SC information such as IP address, machine number, account number, and withdrawal amount. The SC we designed for a banking system is given in Figure 5. OC, on the other hand, is supplied by a service profile repository, and supplies the information shown in Figure 6.

Criteria	Element of SC	Element ID	Address
Who	User ID	USR_ID	Unique user ID of core banking system
	Department Id	DEPT_ID	Department ID of user
	Bank code	BNK_CD	Unique ID of bank
	SSO session key	SSO_KEY	Unique session ID
When	Request date	REQ_DT	Service request date
	Request time	REQ_TM	Service request date
Where	IP address	IP_AD	IP address of user machine
	MAC address	MAC_AD	MAC address of user machine
	Machine number	MCN_NUM	Unique machine ID
	Environment	ENV_CD	D : Development, T : Test, R : Real
What	Request service ID	REQ_SVC_ID	ID of service which user requests
	Request type	REQ_TY	0 : simple request, 1: begin of bulk requests, 2:middle of bulk requests, 9 : end of bulk requests
	Transaction Type	TRX_TY	Q : Request new transaction, R : Response for request
	Request serial number	REQ_SER_NUM	Request serial number of bulk requests
How	Global ID	GLOB_ID	Unique ID of service request
	First transmission channel	FST_TS_CH	The first channel of service request
	Previous transmission channel	PRV_TS_CH	Previous via-channel
	Previous transmission node	PRV_TS_ND	Node number of previous via-channel (multi-node env.)
	Screen number	SCR_NUM	Screen number of request service
	Login with previous date	PRV_DT_LGI	Whether user requests service as previous date or not
Why	Cancelling transaction	CNC_TS	Whether the request is for cancelling previous transaction or not
	Correction transaction	CRC_TS	Whether the request is for correction previous transaction or not

Figure 5: Subject context design for banking system

Criteria	Element of OC		Element ID	Address
Who	Permitted departments	Sales	PD_A	Departments that are permitted to use this service
		General affairs	PD_B	
		Personnel affairs	PD_C	
		Call center	PD_D	
		Electronic banking	PD_E	
		Product development	PD_F	
	Permitted roles	Teller	PR_A	Roles that are permitted to use this service
		Manager	PR_B	
		Department leader	PR_C	
		Staff	PR_D	
		Sales	PR_E	
		Credit assessment	PR_F	
When	Available begin time		AV_BG_TM	Time when this service is available from
	Available end time		AV_ED_TM	Time when this service is available to
	Holiday available or not		HLDY_AV_YN	Whether this service is available in holiday or not
Where	Permitted channel	Internal terminal	PC_A	Service channels that are permitted to use this service
		Internet banking	PC_B	
		ATM	PC_C	
		Mobile banking	PC_D	
		Call Center	PC_D	
What	Request service ID		REQ_SVC_ID	Unique ID of this service (Transaction ID)
	Service name		SVC_NM	The name of this service
	Application code		APP_CD	The application code of this service
	Service info.	Simple inquiry	SI_A	The properties of this service
		Modification	SI_B	
		Deposit/Repay	SI_C	
		Withdraw/Pay	SI_D	
		Open accounts / Register	SI_E	
		Close accounts	SI_F	
		Cancel / Delete	SI_G	
		Money transaction	SI_H	
		Charge fee	SI_I	
How	Available or not		AV_YN	Whether object context information of this service is available or not
	Required approval		REQ_APRV	Whether approval of manager for this service transaction is needed or not
Why	Revocable or not		RVCB_YN	Whether the result of this service is revocable or not
	Correctable or not		CRTB_YN	Whether the result of this service is correctable or not

Figure 6: Object context design for banking system

Case Study

Case 1: Access control for applications when the business date is changed:

Business date management is an important task in any banking system. Therefore, banking applications use special business date that the status of business is reflected as basis date information because the date information provided by mechanically system (i.e., system standard clock) does not consider the status of business (e.g., whether the settlement accounts of the day is completed or not and whether all the transactions of the day is processed properly or not). Even if the system time is 00:00, the business date of the banking system will not change until all transactions for the day are processed properly. Thus, applications for date management in a banking system should advance the business date only after checking that all the jobs of the day have been successfully completed, at which point all online transactions should be blocked, to ensure data compatibility for all

transactions. To complicate matters further, consider that an external channel carries both inbound transactions and outbound transactions, and one banking system can request a deposit to another banking system, and vice versa. While online transactions are blocked in our banking system, the response for deposit that our banking system has requested prior to blocking may be passed through out banking system. If these responses are also blocked, changing business date of the day is not completed permanently because the transaction with external bank can be finished, when response of the other bank is processed. Hence, we design the access control policies for changing business date as follows:

- All online requests from internal channels are blocked
- All online requests from external channels are blocked except for responses of external organizations to our requests

Access control policies for changing business date have nothing to do with service properties. Furthermore, system administrator or automated scheduler system should apply the access control policies in proper condition and time. For our CIAAC model, we apply the s-acr's in Figure 7, where "MC" and "FP" indicate the system IDs of "MCI" and "FEP" respectively. At the first s-acr (i.e., s-arc#1), all requests from internal channels are blocked, and at the second s-acr (i.e., s-acr#2), all new transaction requests from external channels are blocked. Because S-ACR uses the negative access control policy, service requests are blocked if SC information in the request satisfies the first or second s-acr.

"OR" logical condition

Element of SC	Element ID	*s-acr#1*	*s-acr#2*
User ID	USR_ID		
Department Id	DEPT_ID		
Bank code	BNK_CD		
SSO session key	SSO_KEY		
Request date	REQ_DT		
Request time	REQ_TM		
IP address	IP_AD		
MAC address	MAC_AD		
Machine number	MCN_NUM		
Environment	ENV_CD		
Request service ID	REQ_SVC_ID		
Request type	REQ_TY		
Transaction type	TRX_TY		**Q**
Request serial number	REQ_SER_NUM		
Global ID	GLOB_ID		
First transmission channel	FST_TS_CH		
Previous transmission channel	PRV_TS_CH	**MC**	**FP**
Previous transmission node	PRV_TS_ND		
Screen number	SCR_NUM		
Login with previous date	PRV_DT_LGI		
Cancelling transaction	CNC_TS		
Correction transaction	CRC_TS		

"AND" logical condition

Figure 7: Example of S-ACR policies for changing date in banking system

Case 2: Access control for applications handling privacy information :

Some banking applications may also support internal matters, such as the management of employee information. Because employee privacy has become so important lately, access control policies are needed when working with social security numbers, salaries, performance assessments, medical records, and so forth.

A "retrieve personnel information" service should be permitted only to employees belonging to the personnel department, and only over internal terminal channels. Accordingly, as depicted in Figure 8, our access control policies are as follows:

- Service Id: HRM10110
- Service Name: Retrieve personnel information"
- Who to permit : Any persons belonging to the personnel department
- What channel to permit : Internal terminal

These access control policies can be decided by properties of the applications. Developers should implement the application according to design documents of the application with excluding access control policies. After implementation, developers should check the application whether to work properly as intended, then the developer can define access control policies using the OC information shown in Figure 8.

Service Id	Service name	…	Permitted Departments					…	Permitted Channels					…	Available or not
			PD_A	PD_B	PD_C	PD_E	PD_F		PC_A	PC_B	PC_C	PC_D	PC_E		
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
HRM10110	**Retrieve personnel information**	**…**	**N**	**N**	**Y**	**N**	**N**	**…**	**Y**	**N**	**N**	**N**	**N**	**…**	**Y**
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

Personnel affairs

Internal terminal

Figure 8: Example of O-ACR policies for applications managing privacy information

Case 3: Urgent defense against DDoS attack :

There are a number of emergencies that can threaten stable bank system operation, including request capacity overflow, or simple component failure. In such cases, system operators and administrators must urgently constrain service requests so as to minimize damage to the system.

In this section, we explain how to defend against distributed denial of service (DDoS) attack using S-ACR in CIAAC. Suppose that an Internet banking system is under DDoS attack from external anonymous sources. The Internet banking system is at capacity and unmanageable. Due to the service requests flooding in from this system, the load on the core banking system increases rapidly and continuously, and service response times begin to lengthen. To prevent systemic paralysis, system operators must promptly intervene and temporarily block service requests from the Internet banking system to the core banking system. As a first step, they register the s-acr that blocks all such request as follows:

- Block all requests for which the first transmission channel is Internet banking (IB)
- After searching system logs, system operators find that the target service of the DDoS attack is a "Retrieve account balance" service and the service ID is "DPM32001". They alter the registered s-acr to block just those requests produced by the attack
- Block all requests for which the first transmission channel is Internet banking (IB)
- Block all requests for service ID "DPM32001"

As a result, the core banking system is stabilized and, after mitigating the DDoS attack, the registered s-acr is relieved.

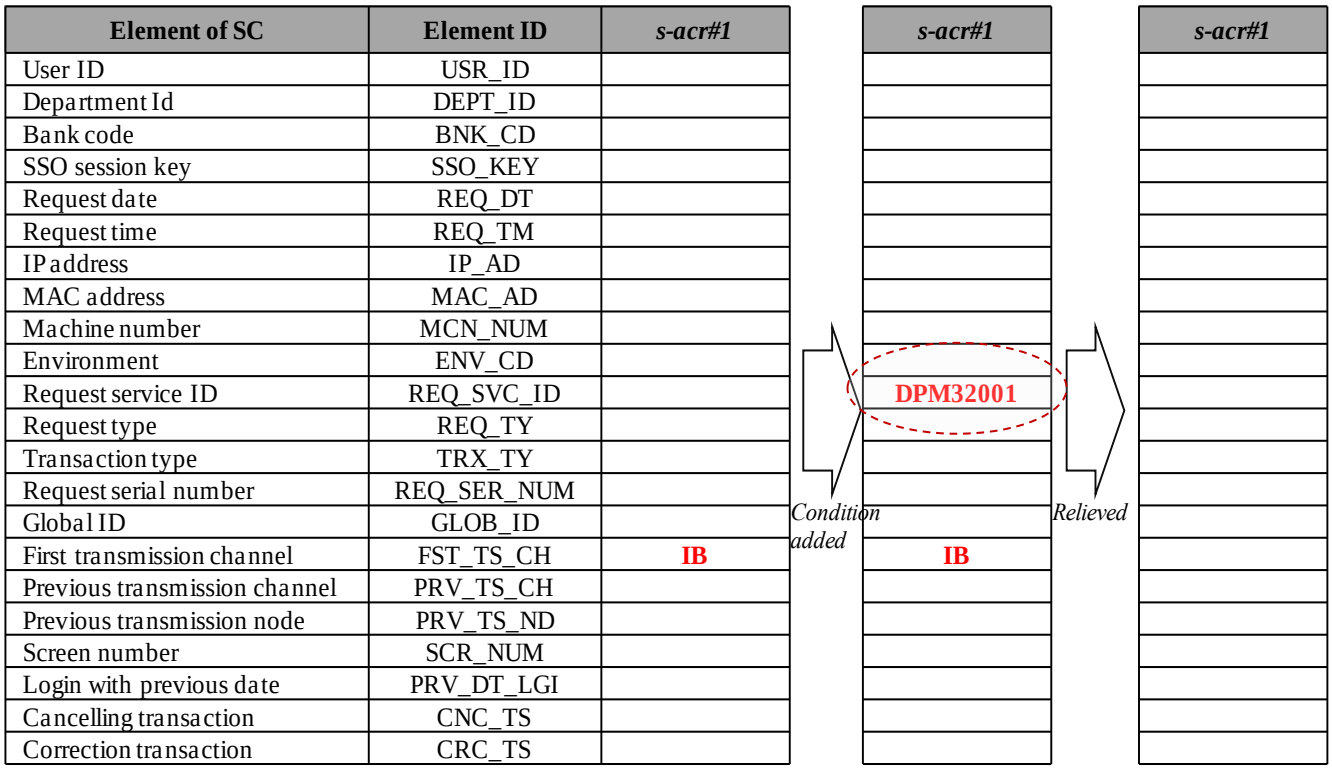

Figure 9: Example of S-ACR policies change for urgent, targeted service blocking

CONCLUSION

In this paper, we presented briefly a context information-based applicaton access control model, which can provide access previlige for complex business applications based on context information by separating context awareness and access control policies from business and processing logic. Furthermore, the proposed model was verified by running three case studies. Although improved performance is not the primary objective of CIAAC, we can significantly reduce, and perhaps eliminate, access control bottlenecks by caching the S-ACR and O-ACR matrices,

In this paper, we limited our rules composition to the use of Boolean AND and OR operators. Though we feel that this limitation keeps things clean and simple, we realize that it may also lead to unnecessarily long and complex expressions. To evaluate this potential drawback, we intend to test CIAAC extensively via reference implementation, and to explore the use of Boolean algebra to reduce complex expressions.

ACKNOWLEDGEMENTS

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No.R7115-16-0002, Wise-IoT).

REFERENCES

[1] Dey, A. K., 2001, “Understanding and using context,” Personal and Ubiquitous Computing, 5(1), pp. 4-7

[2] Kizza, J. M., 2007, “A guide to computer network security,” Springer-Verlag

[3] Kim, Y.-G. and Lee, Y., 2016, “Context information-based application access control model,” Proc. of the 10th International Conference on Ubiquitous Information Management and Communication, Danang, Vietnam, Article No.75

[4] Kim, K., Ko, H., and Hwang, H., 2007, “Context RBAC/MAC access control for ubiquitous

environment," Lecture Notes in Computer Science, 4443, pp.1075–1085

[5] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E., 1996, "Role-based access control models," IEEE Computer, 29(2), pp.38-47

[6] Covington, M. J., Moyer, M. J., and Ahamad, M., 2000, "Generalized role-based access control for securing future applications,". Proc. of the 23rd National Information Systems Security Conference, Baltimore, Maryland, pp.40-51

[7] Park, S.-H., Han, Y.-J., and Chung, T.-M., 2006, "Context-role based access control for context-aware application,".Lecture Notes in Computer Science, 4208, pp.572–580

[8] Kim, Y.-G., Moon, C.-J., Jeong, D., Lee, J.-O., Song, C.-Y., and Baik, D.-K., 2005, "Context-aware access control mechanism for ubiquitous applications," Lecture Notes in Artificial Intelligence, 3528, pp.236–242

[9] Kim, Y.-G. and Cha, S., 2013, "Dynamic access control policy based on RBAC for ubiquitous applications," Information-An International Interdisciplinary Journal, 16(9(B)), pp.7175-7190