Chapter 15
An Efficient Access Control Mechanism for Application Software Using the Open Authentication

Seon-Joo Kim, Jin-Mook Kim and In-June Jo

Abstract As the cloud service is developing, technologies such as virtualization/big data processing is being proposed, but security accidents are occurring constantly. Therefore, companies which are afraid of disclosure of their main data build their own private cloud service. However, in the virtualization software, it is difficult to control the execution of application software and there are also other issues, such as system resource waste, repeated user login execution. Hence, this paper considers executing application software in accordance with the user privilege, by using the Open Authentication and the virtualization technology. For this purpose, a proposed system was designed and implemented, and it was verified by simulation that the proposed system reduced the system resource usage and could execute application software on Web according to the user privilege. Our proposed scheme can support another private cloud service such as SaaS more efficiently.

Keywords Open authentication · OAuth · Private cloud · Virtualization · Modeling · Cloud service · SaaS

S.-J. Kim
SQEC, TTA 267-2 Seohyun-dong, Bundang-Gu, Seongnam-City, Gyunggi-do 463824, Korea
e-mail: sunjoo@tta.or.kr

J.-M. Kim (✉)
Division of Information Technology, Sunmoon University, 100 Kalsan-ri, Asan-si, Tangjeong-myeon 336708, Korea
e-mail: calf0425@sunmoon.ac.kr

I.-J. Jo
Department of Computer Engineering, Paichai University, 155-40 Baejae-ro, Seo-Gu, Daejeon 302735, Korea
e-mail: injune@pcu.ac.kr

H.-K. Jung et al. (eds.), *Future Information Communication Technology and Applications*, Lecture Notes in Electrical Engineering 235, DOI: 10.1007/978-94-007-6516-0_15,

15.1 Introduction

Recently, the cloud service for various mobile devices has been developed. To support diverse mobile devices, several technologies such as virtualization, provisioning and big data processing have been proposed. However, in spite of these technologies, security accidents are occurring constantly. Therefore, companies that are afraid of disclosure of their main data build their own private cloud service. But, this private cloud service structure has several issues. First, it wastes system resource by installing the Guest OS twice in the server. Second, to use application software, you have to get through the user login to the Guest OS repeatedly. Third, connection is possible only through remote desktop software. Finally, to reduce the repeated user login procedure, some companies build additionally Single Sign-On (SSO).

This paper suggests an efficient access control mechanism for application software using the OAuth in the SaaS cloud System (ACMOS). ACMOS is proposed to reduce the system resource and to use application software through Web with no need to install remote desktop.

15.2 Relevant Researches

15.2.1 Outline of Cloud Service

Cloud service is an internet-based computing technology and Web-based software service that installs software in a utility data server on internet and supports bringing into a computer or a mobile device and using the software, whenever necessary [1]. Although this clouds service is provided by famous IT companies, such as Google, Amazon, Apple, Daum, KT, Naver, security accidents are occurring constantly. Therefore, companies build their own private cloud service using virtualization software. However, this private cloud service structure has several issues.

First, it wastes system resource by installing the Guest OS twice in the server. Second, to use application software, you have to get through the user login to the Guest OS repeatedly. Third, connection is possible only through remote desktop software.

15.2.2 Outline of OAuth

The open authentication (OAuth) protocol is an authorization protocol that supports main data sharing between different Web sites without exposing main information of users, and it enables access to Web site or application software

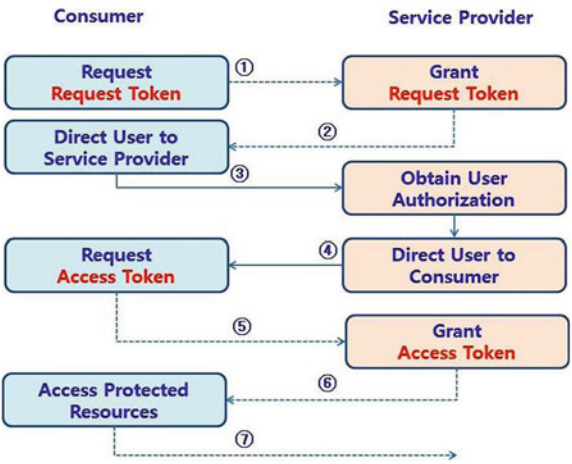

Fig. 15.1 OAuth action procedure, in the figure, service provider means a Web site, and consumer means the web site that approaches the service provider on behalf of general users

making use of a security token, without repeated procedures of recognition and authorization. This protocol was published as RFC 5849 in 2010 [2, 3].

The action procedure of the open authentication protocol is shown in the figure (Fig. 15.1).

(1) Consumer requests for token from service provider using HTTP.
(2) Service provider receives token from consumer, and, after its verification, replies with the token containing secret information of consumer.
(3) To receive the access authorization from user, service provider redirects to the authorized URL.
(4) After receiving password put in by user, service provider redirects to the call back URL.
(5) Consumer requests for consumer_key, token value, signature, timestamp, access token containing nonce, from service provider.
(6) After inspecting the consumer signature, service provider delivers the access token.
(7) Consumer requests the protected resource using the received access token. The following is an example that consumer requests for the protected resource at service provider.

```
http://203.250.xxx.xxx/photos?file=vacation.jpg&size=original&oauth_consumer_key=dr32f5w3t4e2y09&oauth_token=necd744e10el3jjk&oauth_signature_method=HMAC-SHA1&oauth_signature=tR3%2BTy81lMeYAr%2FFid0kMTYa%2FWM%3D&oauth_timestamp=1210092096&oauth_nonce=kllo9940pd9333jh&oauth_version=1.0
```

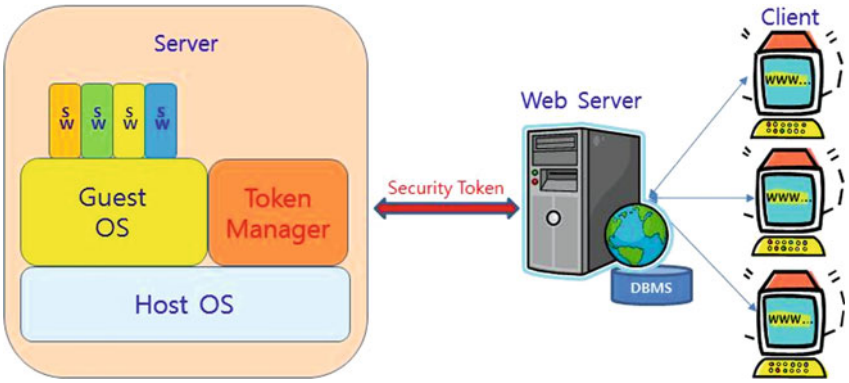

Fig. 15.2 ACMOS architecture

15.3 Proposed Mechanism

15.3.1 Basic Design of ACMOS

The overall architecture of ACMOS is shown below (Fig. 15.2).

A Web server provides web environment to a client, and requests, using a security token, the execution of Software installed on Guest OS. Host OS and Guest OS, Token Manager, Application are installed on the server. Host OS manages the execution of Guest OS and Token Manager, Token Manager verifies the security token received from the Web server, then the client examines the execution privilege of Guest OS and Application and performs the function of calling these. Guest OS receives the call of Token Manager and supports the execution of Application. Finally, the Application installed on Guest OS is executed on the user request.

The following (Fig. 15.3) is the marking of detailed modules of the proposed system, and the object of Token Manager is composed of SVM, SEM, GEM, SMM modules. Security Token Verify Module (SVM) is the module that examines the effectiveness of the security token received from the Web server and manages the security token. Session Management Module (SMM) is the module that maintains/manages the software session that is executed on the Client and Guest OS. Guest OS Execution Module (GEM) is the module that executes or ends Guest OS according to the request of SVM, and Software Execution Module (SEM) is the module that executes or ends Application Software according to the request of SVM.

Web Server is composed of UIM, IAM, SWM, STM, TMM, LMM. User Information Module (UIM) is the module that manages the user information, and user Identification and Authentication Module (IAM) is the module that recognizes and authorizes users according to registered information. Software access control Module (SWM) is the module that manages the list of Application Software

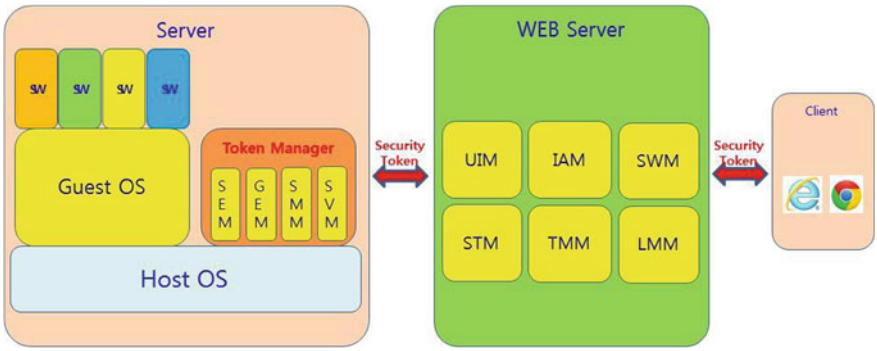

Fig. 15.3 Detailed modules of the proposed system

executable for each user, Security Token manager Module (STM) is the module that creates a security token according to the user execution privilege to Application Software, Transaction Management Module (TMM) is the module that manages the session while users use Application Software, and Log Management Module (LMM) performs the function of storing and inquiring various log data occurring on the Web server.

The structure of the security token used in ACMOS is as follows. This token is an object that contains access privilege to Guest OS and Application, as follows:

$$ST = \{UserInfoKey \ \& \ oauth\_signature\_method \ \& \ oauth\_signature \ \& \ oauth\_timestamp \ \& \ oauth\_nounce \ \& \ UserPermission\}$$

UserInfoKey is a 128 bit string that hashed UserInfo.

oauth_signature_method is an algorithm that signs on the security token, and is fixed to HMAC-SHA1 for use.

oauth_signature is the signed value on the security token.

oauth_timestamp is the time stamp value at the time of creating the security token.

oatuth_nounce is a series of letters randomly generated to protect from the retransmission attacks at the time of requesting the user authorization.

userPermission is a series of letters that contains user information, information of the access privilege to Guest OS and application software.

15.3.2 The Proposed System Procedure

The following (Fig. 15.4) shows the action procedure of ACMOS.

(1) A Client connects to a Web server and puts in ID/Password.

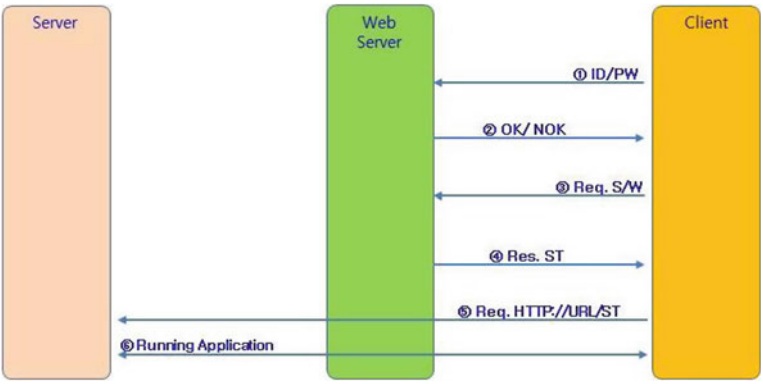

Fig. 15.4 The ACMOS action procedure

(2) The Web server examines the ID/PW and delivers the result to the Client. At this moment, the web server's deliver the list of Application Software to a right client and deliver an error message to an invalid user.
(3) A right Client requests the execution of Application Software.
(4) The Web server creates a security token, which contains ID, IP Address & MAC Address, Software Permission, Guest OS Permission, Timestamp, Nounce, and delivers it to the Client.
(5) The Client requests, in the URL format (ex. http://203.250.143.54/ security token series of letters), the server to execute Application Software.
(6) To verify the effectiveness of the security token, the server confirms the security token signature value, time stamp value, Nonce. If the security token is effective, the server delivers the Application Software list to the Client.

15.4 Verification and Consideration

To verify the effectiveness of ACMOS, two simulations were performed. The first simulation measured the system response time of the existing system and ACMOS. The second simulation measured the response time and the system resource change of the proposed system in accordance with the change of user number (1 user, 5 users, 10 users, 25 users).

In the first scenario, the response time of ACMOS (7.3 s) was 3.5 times longer than the existing system (2.1 s). The system resource was substantially reduced from the existing system to CPU use rate in average (3.7 %), memory usage (2.9 GB), HDD usage (38.9 GB), as shown in (Fig. 15.5). It is considered that the response time was longer because the TCP/IP-based socket communication was changed to HTTP and the OAuth-based security token creation/verification

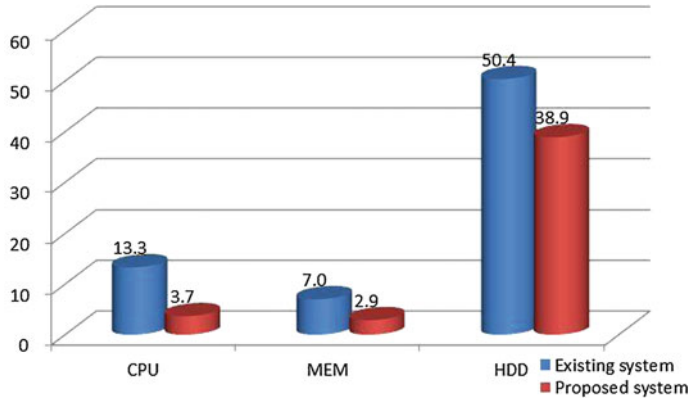

Fig. 15.5 System resource measurement

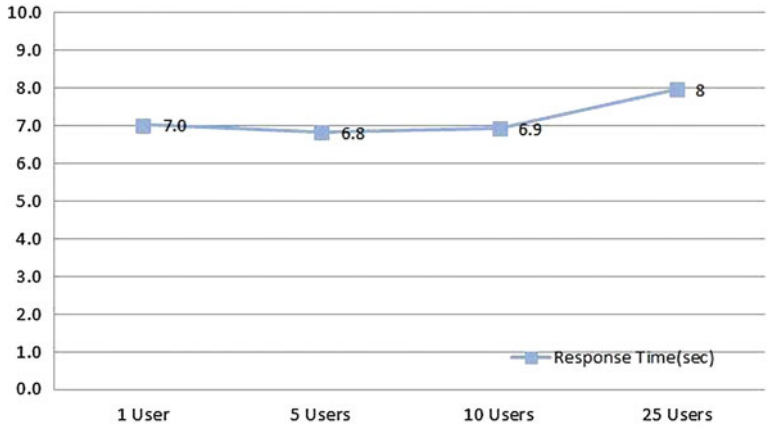

Fig. 15.6 Measurement of response time in accordance with the change of user number

procedure was complicated, and the number of Guest OS decreased from the existing system (4) to ACMOS (2).

In the second scenario, as the number of users changes (1 user, 5 users, 10 users, 25 users), the response time was measured as 7.0, 6.8, 6.9, 8.0 s, as shown in (Fig. 15.6), and as for the system resource change, the CPU usage was measured as (8.6, 14.0, 25.6, 29.2 %), as shown in (Fig. 15.7), but the memory usage (2.9 GB) and HDD usage (38.8 GB) showed no change.

The above measurement in the proposed system comes from average measured in each system for each user, which exhibited no difference in the response time from 1 to 10 users but 1 s difference for users above 25. This shows that the performance of the proposed system can handle stably only up to ten simultaneous

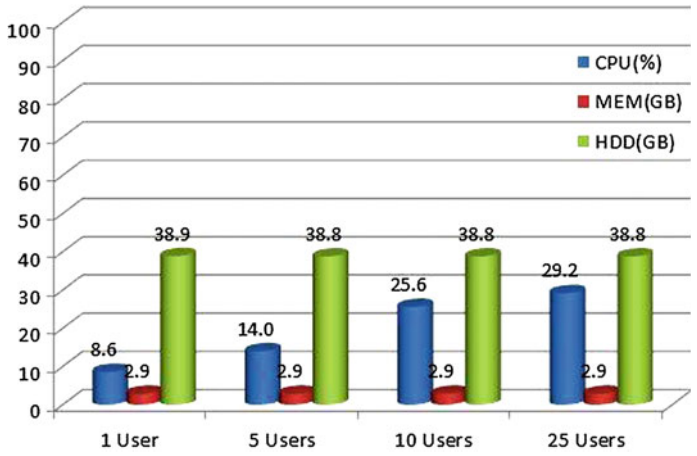

Fig. 15.7 Measurement of system resource in accordance with the change of user number

users. And, in the system resource change rate, the CPU use rate increases as the number of users increases, but the memory/HDD usage showed no change because of the system resource restriction managed by the virtualization Software.

15.5 Conclusions

This paper proposes ACMOS based on an open protocol, to control the execution of application software in accordance with the user privilege. ACMOS installs software in integration on the Guest OS and is designed to execute software through login once. Comparison of the existing system and the proposed system confirmed that system resource usages (CPU, RAM, and HDD) are reduced but the response time becomes slightly longer in the proposed system. And, it was possible to reduce repeated user login and to use application software through Web.

As a future research work, security verification for using the open authentication protocol and continuous improvement on the late response with the increasing number of users will be needed.

References

1. OAuth Administrator manual, http://www.wikipedia.org
2. Getting Started OpenAuth, http://dev.aol.com/
3. OAuth Core 1.0, http://oauth.net/core/1.0