http://dx.doi.org/10.5392/JKCA.2013.13.12.568

개방형 인증 프로토콜을 이용한 가상 운영체제에 설치된 SW 접근통제 방안

Access Control Method for Software on Virtual OS Using the Open Authentication Protocol

김선주*, 조인준**
한국정보통신기술협회*, 배재대학교 컴퓨터공학과**

Sun-Joo Kim(sunjoo@tta.or.kr)*, In-June Jo(injune@pcu.ac.kr)**

요약

최근 들어 국내외 대형 IT 기업들은 하드웨어 또는 소프트웨어 기반의 기술을 활용하여 다양한 클라우드 서비스를 제공하고 있다. 이러한 클라우드 서비스 기술을 통해 사용자로 하여금 지리적 위치나 PC, 모바일 장비 등의 제약 없이 다양한 클라우드 서비스를 이용할 수 있는 장점을 가지고 있다. 이러한 클라우드 서비스를 제공하기 위한 주요 기술로는 가상화, 프로비저닝, 빅데이터 처리 기술 등이 있다. 이러한 기술들이 있음에도 불구하고, 다양한 유형의 보안 사고가 지속적으로 발생하고 있다. 이로 인해 주요 데이터의 외부 노출을 꺼리는 기업들은 하드웨어와 소프트웨어를 직접 구축 및 운영하는 사설 클라우드 서비스를 직접 구축하고 있다. 이때, 가상화 기반 환경에서 사용자의 권한에 따라 소프트웨어를 다르게 제공하고자 하는 경우 사용자 권한에 따라 소프트웨어의 실행을 제어하기 어렵고, 시스템 리소스 낭비 문제, 반복적인 사용자 로그인 수행 등 여러 가지 문제점이 있다. 이에 본 논문에서는 개방형 인증 프로토콜과 가상화 기술을 접목하여 사용자 권한에 따른 응용 소프트웨어를 제공하는 방안을 제안 하였다.

■ 중심어 : | 클라우드서비스 | 사설 클라우드 | 가상화 | 접근통제 | 개방형 인증 프로토콜 |

Abstract

In recent years, IT companies offer various cloud services using hardware-based technologies or software-based technologies. User can access these cloud services without the constraints of location or devices. The technologies are virtualization, provisioning, and big data processing. However, security incidents are constantly occurring even with these techniques. Thus, many companies build and operate private cloud service to prevent the leak of critical data. If virtual environment are different according to user permission, many system are needed, and user should login several virtual system to execute an program. In this paper, I suggest the access control method for application software on virtual operating system using the Open Authentication protocol in the Cloud system

■ keyword : | Cloud Service | Private Cloud | Virtualization | Access Control | Open Authentication Protocol |

I. 서 론

2008년 애플에서 출시한 아이폰은 단순히 휴대전화 기능을 뛰어 넘어, 사용자의 취향에 따라 소프트웨어를 선택적으로 설치하여 사용할 수 있는 모바일기기라는 점에서 큰 이슈가 되었다. 이에 따라 사용자들은 시간과 장소

접수일자 : 2013년 09월 16일
수정일자 : 2013년 10월 18일
심사완료일 : 2013년 10월 21일
교신저자 : 조인준, e-mail : injune@pcu.ac.kr

에 구애받지 않고 언제든지 인터넷에 접속하여 다양한 클라우드 서비스를 제공 받을 수 있는 기술이 발전하였다. 즉, 클라우드 서비스는 인터넷 기반의 컴퓨팅 기술로 인터넷 상의 유틸리티 데이터 서버에 소프트웨어를 설치해 놓고 필요할 때 마다 컴퓨터나 모바일기기로 불러와서 사용할 수 있도록 지원하는 웹 기반의 소프트웨어 서비스이다[7]. 즉, 인터넷을 통해 사용자들에게 IT 자원을 제공하는 서비스이다.

이러한 클라우드 서비스는 제공 방식에 따라 Infrastructure as a Service(이하 'IaaS'라 함), Platform as a Service(이하 'PaaS'라 함), Software as a Service (이하 'SaaS')로 분류하거나 서비스 관리 주체에 따라 [표 1]과 같이 공용 클라우드 서비스, 사설 클라우드 서비스, 하이브리드 클라우드 서비스로 분류할 수 있다[3][4].

표 1. 클라우드 서비스 개요

구분	개요
공용 클라우드 서비스	- 서비스 제공자가 IT자원을 구축하고, 기업/사용자가 서비스로 이용하고 사용료를 지불하는 방식 - 제3자에 의한 기업/사용자의 주요 데이터 노출, 저작권 등의 보안 위험 존재 - 아마존 EC2/EC3, 구글 앱스, 다음넷 다음클라우드, KT 유클라우드 등
사설 클라우드 서비스	- IT자원을 기업이 직접 구축하고, 서비스를 직접 운영하는 방식 - 기업의 주요 데이터 노출 위험이 줄어듦 - IT자원 구매와 유지보수 비용이 많이 소요됨 - IBM, HP, VMWare, EMC 등
하이브리드 클라우드 서비스	- 공용 클라우드 서비스와 사설 클라우드 서비스의 장점을 결합한 방식 - 기업의 주요 데이터는 직접 구축한 IT자원을 활용하고, 그 이외의 부분은 공용 클라우드 서비스를 활용 - IT자원의 구매와 유지보수 비용이 소요됨 - 시스코, 오라클, 인텔 등

이와 같은 클라우드 서비스는 2010년 1.5억 달러에서 연평균 49% 성장하여 2015년 10.9억 달러 규모로 성장할 것으로 예상한다. 특히, SaaS 분야는 0.7억 달러에서 5.8억 달러 규모로 연평균 51% 성장할 것으로 예상하고 있다[1]. 또한, 향후 클라우드 컴퓨팅은 그리드에서 사설 클라우드 서비스와 공용 클라우드 서비스가 함께 발전할 것으로 예상이 된다[2].

이처럼 지속적으로 성장하는 클라우드 서비스를 지원하기 위한 가상화/프로비저닝/빅데이터 처리 기술 등의 다양한 기술들이 발전하고 있음에도 불구하고, 클라우드 서비스 관련 보안 사고는 지속적으로 발생하고 있으며, 대표적인 사고 사례는 [표 2]와 같다[5][6].

표 2. 클라우드 서비스 보안사고 사례

서비스	보안사고 사례
IaaS	- 아마존의 S3 서비스 인증 요청 문제로 서비스 속도 저하(2008.2) - 악성코드의 감염에 의한 아마존 EC2 서비스 침해 사고(2009.9) - 마이크로소프트 데인저 데이터 손실 및 접속장애로 사이드킥 서비스 중단(2009.10) - 국내 K 社의 관리자 실수로 고객 정보 및 가상머신 삭제 사고(2012.
PaaS	- 마이크로소프트 애저 다운(2009)
SaaS	- 트위터 서버의 좀비컴퓨터 공격에 의한 다운 (2010.6) - 구글 데이터센터네 소프트웨어 오류 지메일 서버다운 및 주소록 삭제(2009.2)

이러한 이유로 인해 주요 데이터를 외부에 노출을 꺼리거나, 안정적인 서비스를 원하는 기업은 사설 클라우드 서비스를 직접 구축 및 운영한다. 하지만, 해당 기업의 전산담당자는 예산 부족 및 서비스 구축 기간 부족 등을 이유로 신규로 서비스를 개발하지 않고, 기존의 시스템에서 제공하던 서비스를 변경 없이 기존과 동일한 서비스가 제공되도록 요구한다. 전산담당자들의 이러한 요구사항으로 인해 기존 서버 장비에 호스트 운영체제와 가상화 소프트웨어를 설치하고, 서비스 별로 가상 운영체제를 독립적으로 설치하여, 하나의 서버장비에서 여러 개의 응용 서비스를 제공하게 된다. 또한, 사용자들은 가상 운영체제에 원격으로 접속하기 위해 원격 데스크톱 소프트웨어를 설치하고, 사용자 권한에 따라 가상 운영체제에 원격 데스크톱 소프트웨어를 이용하여 서비스를 이용한다.

이러한 구조로 인해 다음과 같은 문제가 발생한다. 첫째, 하나의 서버 장비에 여러 개의 가상 운영체제와 서비스에 필요한 응용 소프트웨어의 반복하여 설치함에 따른 서버 장비의 리소스를 낭비한다. 둘째, 원격에서 서비스를 운영하는 서버에 접속하고자 하는 사용자는 반드시 전용 데스크톱 소프트웨어를 설치 후 해당 서버에서 제공하는 서비스를 사용할 수 있다. 셋째, 한명의 사용자가 각각 다른 가상 운영체제의 서비스를 변경하면서 접속하

고자 하는 경우 가상 운영체제 별로 반복적으로 사용자 로그인을 하거나 한 번의 인증 과정으로 여러 개의 가상 운영체제에 접근을 가능하게 하는 싱글사인온(Single Sign On)을 사용해야 한다. 넷째, 가상 운영체제의 보안 패치 및 소프트웨어 업데이트를 개별적으로 수행함에 따라 관리자가 관리해야 할 운영체제와 소프트웨어의 수량이 증가한다. 마지막으로, 관리 대상인 가상 운영체제의 수량 증가에 따른 운영체제에 대한 라이선스 비용과 소프트웨어 라이선스 비용이 증가한다.

따라서 본 논문에서는 개방형 인증 프로토콜과 가상화 기술을 통해 사용자 권한에 따른 응용 소프트웨어 실행 방안을 제안한다.

II. 관련 연구

2.1 싱글사인온(Single Sign On)

일반 기업의 전산실에는 다수의 서버가 구축되어 있고, 서버 관리자는 이러한 다수의 서버에 개별적으로 접속하여 관리자의 식별 및 인증과정을 거친다. 이처럼 관리자나 사용자로 하여금 여러 번의 인증 절차를 거치는 것을 한 번의 아이디와 비밀번호를 입력하여, 각종 시스템이나 서비스를 이용할 수 있는 보안 솔루션으로 싱글사인온 또는 단일 인증이라고도 한다[7][10].

대부분의 기업에서 운영하는 다양한 업무 시스템 또는 서비스 증가에 따라 관리자가 관리해야 할 서버의 수는 기하급수적으로 증가하고 있으며, 이에 따른 싱글사인온에 대한 필요성은 더욱 증가하고 있다. 하지만, 싱글사인온을 적용할 수 있는 대상은 서버 장비에만 적용할 수 있으며, 응용 소프트웨어에 대한 관리자의 접근 권한을 관리하기는 어렵다는 제약사항이 존재한다.

2.2 개방형 인증 프로토콜

대부분의 웹 사이트에서는 사용자 아이디와 비밀번호를 통해 사용자를 식별하고 인증한다. 또한, 하나의 웹사이트에서 식별 및 인증을 거친 후, 다른 웹 사이트로 접속하는 경우 이전에 접속 시 사용한 데이터를 재사용할 수 없고, 다시 사용자의 아이디와 비밀번호를 매번 입력해야 하는 불편함이 존재한다. 이러한 이유로 인해, 사용자 데이터나 웹 리소스를 보호하면서 특정 리소스나 사용자 데이터에 접근이 가능하도록 임시로 접근 권한을 위임할 수 있는 방안이 제안 되었다. 대표적인 방법으로는 트위터의 오픈 아이디(OpenID), 구글 AuthSub, AOL의 OpenAuth, 아마존의 웹서비스 API 등이 있다[11-13].

개방형 인증 프로토콜은 사용자의 주요 정보를 노출시키지 않고 보호대상 웹 사이트 간에 주요데이터를 서로 공유할 수 있도록 표준화도니 인증 프로토콜로써, 웹 사이트나 응용 소프트웨어에서 사용자에게 인증 요청 없이 보호대상 웹 사이트나 보호된 리소스에 접근할 수 있도록 해준다. 이 방식은 2006년 트위터에서 최초 적용되었으며, 2007년 10월에 인터넷 커뮤니티에 의해 개방 인증 프로토콜 표준(Open Authentication: OAuth)초안으로 발표 하였고, 2010년 국제 인터넷 표준화 기구(IETF)의 RFC 5849로 표준화 되었으며, 트위터에서는 모든 제3자 응용 소프트웨어에 대한 사용자 식별 및 인증을 위해 개방형 인증 프로토콜 기술을 사용하고 있다[8][9][13].

개방형 인증 프로토콜은 여러 종류의 보안토큰을 URL 형식으로 통신 상대방과 주고받으면서 보안토큰에 포함된 식별키, 타임스탬프, URL, 전자서명값 등을 검증을 통해 웹 사이트의 보호된 자원에 접근을 허용 또는 거부를 한다. 개방형 인증 프로토콜을 통해 주고 받은 보안토큰의 예제는 [그림 1]과 같다.

```
http://203.250.143.52/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
```

그림 1. 보안토큰 예제

III. 제안 방안

서론에서 언급한 사설 클라우드 서비스 구축에 따른 문제점을 해결하기 위한 방안으로 본 논문에서는 개방형 인증 프로토콜을 이용하여 사용자 권한에 따라 응용 소프트웨어를 실행할 수 있는 방안을 제안한다.

3.1 개요

제안방안은 [그림 2]에서 보는 바와 같이 어플리케이션서버, 중계서버, 클라이언트로 구성된다.

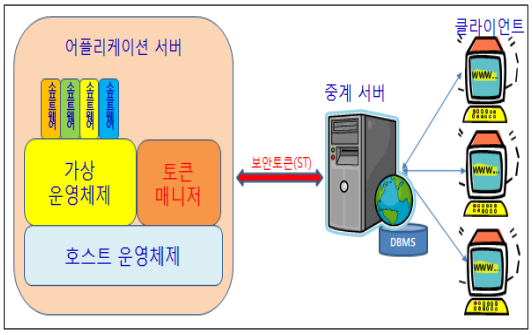

그림 2. 제안방안

구성요소 중 어플리케이션서버는 가상 운영체제에 설치된 응용소프트웨어를 클라이언트의 사용요청에 따라 서비스를 제공하며, 호스트 운영체제와 가상 운영체제, 토큰매니저, 소프트웨어로 구성된다. 이때, 호스트 운영체제는 어플리케이션서버에 설치된 운영체제이며, 가상 운영체제는 토큰매니저의 서비스 요청에 따라 소프트웨어를 실행할 수 있도록 운영환경을 지원한다. 토큰 매니저는 가상운영체제에 설치된 다양한 응용 소프트웨어의 접근 권한 검사 및 실행을 제어하며, 중계서버로부터 수신한 보안 토큰의 전자서명을 검증한다. 마지막으로, 가상 운영체제에 개별적으로 설치된 소프트웨어는 클라이언트의 요청에 따라 실행이 가능한 응용 소프트웨어로 한글 2010, MS 오피스 제품(워드, 파워포인트, 엑셀) 등이다.

중계서버는 사용자에게 웹 환경을 제공하고, 사용자를 식별 및 인증 후 보안토큰을 생성하여 어플리케이션과의 통신을 통해 어플리케이션 서버 내에서 동작하는 소프트웨어를 사용자가 이용할 수 있도록 중계한다.

클라이언트는 어플리케이션 서버에 설치된 소프트웨어를 사용하는 일반 사용자이다.

본 논문에서 제안한 보안토큰의 구조는 다음과 같다. 보안토큰은 가상 운영체제와 응용 소프트웨어에 대한 접근권한이 포함된 객체로 다음과 같이 구성된다.

보안토큰={ UserInfoKey & oauth_signature _method
& oauth_signature & oauth_timestamp
& oauth_nonce & userPermission }

UserInfoKey는 클라이언트 정보가 포함된 해시 값으로 128비트 문자열이다.

oauth_signature_method는 클라이언트 측에서 사용한 전자서명 알고리즘으로 HMAC-SHA1으로 고정하여 사용한다.

oauth_signature는 보안토큰에 대한 전자 서명 값이다.

oauth_timestamp는 보안토큰을 생성하는 시점의 타임스탬프 값이다.

oatuth_nonce는 재전송 공격을 막기 위해, 클라이언트가 인증시마다 random하게 생성하는 문자열이다.

UserPermission은 클라이언트 정보, 가상운영체제 접근 권한 정보, 응용 소프트웨어에 대한 접근권한 정보가 포함된 정보가 포함된 문자열이다.

3.2 동작 절차

[그림 3]은 제안방안의 동작절차이다.

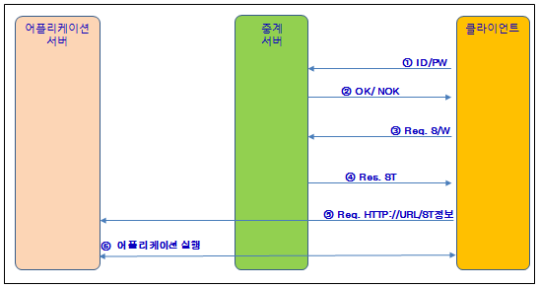

그림 3. 제안방안 동작 절차

(1) 클라이언트는 웹서버에 접속하여 자신의 아이디와 비밀번호를 입력하여, 사용자 로그인을 시도한다.

(2) 중계서버는 클라이언트로부터 수신한 사용자 아이디와 비밀번호를 검사하여 그 결과를 사용자에게 반환한다.

(3) 유효한 사용자인 경우에는 사용자가 사용할 소프트웨어 목록을 요청하고, 그렇지 않은 경우 연결이

종료된다.

(4) 클라이언트로부터 소프트웨어의 실행 요청을 수신한 중계서버는 사용자 아이디, 컴퓨터의 IP 주소 및 MAC 주소, 소프트웨어 실행권한, 가상 운영체제 실행권한, 타임스탬프 및 nonce값 등이 포함된 보안토큰을 생성하여 클라이언트에 전달한다.

(5) 중계서버로부터 보안토큰을 수신한 클라이언트는 소프트웨어가 설치된 서버의 IP 주소가 포함된 URL 형식(예. http://203.250.143.xx/보안토큰문자열)으로 소프트웨어 실행을 요청한다.

(6) 클라이언트로부터 소프트웨어 실행 요청을 수신한 어플리케이션서버는 보안토큰 서명검증, 타임스탬프값, nonce 등의 보안토큰의 유효성을 확인한다. 보안토큰의 유효성이 검증된 클라이언트에 대해 해당 소프트웨어에 대한 실행권한을 검사 후 소프트웨어를 실행시키고, 그 실행화면을 전송한다.

위의 (1)~(6)번 동작절차에 따라 사용자의 PC 웹 브라우저에 표시되는 화면으로 사용 가능한 소프트웨어 목록이 [그림 4]와 같이 표시된다.

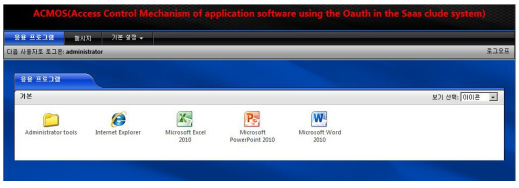

그림 4. 사용자 로그인 후 SW 목록 표시 화면

Ⅳ. 검증 및 고찰

이장에서는 제안 방안과 기존시스템과의 성능을 다음과 같이 비교하였다. 시스템의 원활한 성능 비교를 위해 일반 사용자들이 문서작업을 위해 사용빈도가 매우 높은 소프트웨어로 한글, MS-워드, 엑셀, 웹 브라우저 등을 임의로 선택하였다. 제안방안을 적용하지 않는 방안으로 하나의 가상운영체제에서 하나의 소프트웨어를 사용하는 상황으로 가정하고, 4개의 가상운영체제를 설치하고, 각각의 가상운영체제에 한글, MS-워드, 엑셀, 웹브라우저만 개별적으로 설치하였다. 제안 방안에서는 하나의 가상운영체제에 한글, MS-워드, 엑셀, 웹브라우저를 통합 설치하였다. 성능 측정 방안으로는 허가된 사용자가 응용 소프트웨어 1개를 임의로 선택하여 실행 완료되기까지의 응답시간과 어플리케이션 서버의 리소스(CPU, Memory, HDD) 변화율을 측정하였다.

표 2. 시뮬레이션 결과

구분	적용 전	적용 후
응답시간	2.1초	7.3초
CPU	13.3%	3.7%
MEM	7.0GB	2.9GB
HDD	50.4GB	38.9GB
가상 운영체제수	4개	1개

위의 [표 2]에서 보는바와 같이 응답시간은 제안방안 적용 이전보다 제안방안을 적용했을 때 7.3초로 약 3.5배 더 걸렸다. 또한, 제안방안이 적용했을 때 어플리케이션 서버의 리소스는 72%(CPU), 58%(MEM), 22%(HDD)로 각각 감소한다. 이때, 어플리케이션 서버의 리소스 사용량이 크게 감소하지만, 응답시간은 웹을 통해 어플리케이션의 실행모듈을 다운로드하는데 걸리는 소요시간과 보안토큰 생성/검증에 따른 소요시간이 합산된 시간이다. 라드웨어가 500여개의 웹사이트에서 응답시간을 시험한 결과에서도 대부분이 8초 이내가 74%를 차지하며, 8초 이상도 26%이상임을 알 수 있다[14]. 또한, 국내 웹 응용 소프트웨어의 품질 시험 시 응답시간은 내려 받는 콘텐츠의 크기에 따라 크게 달라질 수 있음을 인정하고 있다.

제안방안에 대해서 보안성 측면을 검토하기 위해 공통평가기준 및 공통평가방법론에 근거하여 공격자의 공격수준을 기본 공격 수준으로 정하고, 공격성공가능성을 계산하였다[15][16]. 공격성공가능성을 계산하기 위해 제안방안에 대한 우회/침해/직접공격에 의한 위협을 도출하고, 도출된 위협에 대한 대응방법을 조사하여 각각에 대한 공격 성공 가능성을 계산하였다. 계산된 결과는 [표 3]과 같다.

표 3. 공격 성공 가능성

위협	대응방법	공격 성공 가능성
클라이언트 컴퓨터의 리소스 접근 가능에 따른 데이터 노출 취약성	사용자 컴퓨터에 백신 및 방화벽 설치를 통한 대응	경과시간-0 전문지식-3 시스템 지식-3 기회-4 도구-4
사용자 비밀정보(아이디/비밀번호)정보 재전송에 따른 취약성	oauth_timestamp와 oauth_nonce값을 통해 재전송 공격에 대응	경과시간-4 전문지식-3 시스템 지식-3 기회-4 도구-4
네트워크 전송 패킷 도청(Sniffing)에 따른 데이터 노출 취약성	oauth_consumer_key와 oauth_consumer_secret를 통해 데이터 암호화를 통한 대응	경과시간-10 전문지식-3 시스템 지식-3 기회-4 도구-4
중간자(Man-In-The-Middle) 공격에 따른 데이터 노출 취약성	oauth_signature 필드를 통해 중간자 공격 탐지를 통한 대응	경과시간-10 전문지식-3 시스템 지식-3 기회-4 도구-4

위의 표에 따라 공격 성공 가능성의 범위는 최소 14~최대 24로 나타났다. 따라서, 정보보호 시스템 공통평가기준 및 공통평가방법론의 취약성 평가방법론에 근거하여 제안방안은 기본 공격 수준을 갖는 공격자에 의해 평가보증등급 EAL4에서는 안전하다고 결정할 수 있다

V. 결 론

본 논문에서는 가상화 소프트웨어에서 사용자 권한에 따른 응용 소프트웨어 실행을 위한 가상화 환경을 지원하는 보안토큰을 활용한 방안을 제안하였다. 즉, 하나의 가상 운영체제에 필요한 모든 소프트웨어를 설치하고, 웹을 통해 사용자 권한에 따라 소프트웨어를 사용할 수 있도록 제안하였다.

제안방안을 통한 장점으로는 첫째, 반복적인 사용자 로그인을 줄일 수 있고, 둘째, 하나의 가상 운영체제에 소프트웨어를 통합 설치해도 동일한 서비스가 가능하였다. 셋째, 원격 데스크톱 소프트웨어를 사용하지 않고 웹 접속을 통해 응용 소프트웨어를 사용할 수 있다. 하지만, 시뮬레이션 수행 결과, 제안 방안의 지속적인 성능 개선과 개방형 인증 메커니즘 사용에 따른 보안 취약성에 대한 연구가 지속적으로 이뤄질 필요가 있다.

참 고 문 헌

[1] 주재욱, 김창완, 정용찬, 염수현, 손상영, 유선실, 김민식, 정부연, 이경남, 오정숙, 이은민, 정현준, 이기훈, 나상우, 이주영, *2013 ICT 시장 전망*, 정보통신정책연구원, 2012(11).

[2] 이병엽, 박준호, 유재수, “고가용성 클라우드 컴퓨팅 구축을 위한 그리드 소프트웨어 아키텍처”, 한국콘텐츠학회논문지, 제12권, 제2호, pp30-39, 2012.

[3] MSDN, http://social.msdn.microsoft.com/

[4] 이병엽, 박준호, 유재수, “클라우드 서비스를 위한 고가용성 대용량 데이터 처리 아키텍처”, 한국콘텐츠학회논문지, 제13권, 제2호, pp.32-43, 2013.

[5] 차영태, “클라우드 컴퓨팅 보안기술동향과 산업전망”, 한국산업기술평가원, 제12권, 제6호, 2012(7).

[6] 이향진, *안전한 클라우드서비스 제공·이용을 위한 보안 고려사항*, CLOUD SEC, 2012.

[7] Wiki, http://ko.wikipedia.org/wiki

[8] http://oauth.net/documentation/getting-started/

[9] OAuth Core 1.0, http://oauth.net/core/1.0/

[10] http://word.tta.or.kr

[11] https://developers.google.com/maps/documentation/mapsdata/developers_guide_protocol?hl=ko#AuthSub

[12] 정병권, 김학영, 최완, *미래사회와 빅 데이터 기술*, IT기획시리즈, pp.23-25, 2012.

[13] Getting Started OpenAuth, http://dev.aol.com

[14] http://www.webperformancetoday.com/

[15] *정보보호시스템 공통평가기준 1부, 2부, 3부*, CCMB, 2006.

[16] *정보보호시스템 공통평가방법론*, CCMB, 2007.

저 자 소 개

김 선 주(Sun-Joo Kim) 정회원

- 1998년 2월 : 배재대학교 컴퓨터공학과 졸업
- 2001년 2월 : 배재대학교 컴퓨터공학과 석사
- 2013년 2월 : 배재대학교 컴퓨터공학과 박사
- 2001년 1월 ~ 2003년 9월 : ㈜케이사인 선임연구원
- 2013년 9월 ~ 현재 : 한국정보통신기술협회 책임연구원

<관심분야> : 클라우드 컴퓨팅, SW 테스팅, 정보보호 제품 평가

조 인 준(In-June Jo) 정회원

- 1982년 2월 : 전남대학교 계산통계학과 졸업
- 1985년 2월 : 전남대학교 전자계산학과 석사
- 1999년 2월 : 아주대학교 컴퓨터공학과 박사
- 1983년 ~ 1994년 : 한국전자통신연구원 선임연구원
- 1994년 1월 ~ 현재 : 배재대학교 컴퓨터공학과 교수

<관심분야> : 정보보호, 컴퓨터네트워크보안, 전산조직응용