SaaS 플랫폼 사용자 인증을 위한 Multi-Authentication Server 설계*

김영만, 임승용, 강민철, 이진범, 임준현, 반은영, 한재일
국민대학교 컴퓨터공학부
{ymkim, chitos, mckang, durtkfmf, dathcain, rhal88, jhan}@kookmin.ac.kr

Design of Multi-Authentication Server for user authentication in SaaS platform

Young Man Kim, Seungyong Lim, Mincheol Kang, Jinbem Lee,
Eunyoung Ban, Junhyun Lim, Jaeil Han
Dept of Computer Science, Kookmin University

요 약

최근 새로운 소프트웨어 배포 방식인 SaaS가 주목을 받고 있다. SaaS는 원활하고 효율적인 서비스 제공을 위하여 소프트웨어 서비스를 제공하는 SaaS 어플리케이션과 과금, 사용자 인증과 같은 공통 서비스들을 제공하는 SaaS 플랫폼으로 역할이 분리되어 있다. 이런 배경을 바탕으로 본논문에서는 효율적인 사용자인증을 위한 SaaS 플랫폼의 사용자인증 서비스인 Multi-Authentication Server를 설계한다. 본논문에서 설계되는 Multi-Authentication Server는 OpenID, OAuth, CAS, X.509 프로토콜을 통합하여 웹기반의 사용자인증 서비스를 구현한다. 마지막으로 웹기반 사용자 인증의 보안성 한계 극복과, 최근 하드웨어 보안기능을 이용하여 RADIUS 프로토콜과 TPM칩을 통합한 하드웨어 기반 네트워크 인증서비스를 제공한다.

1. 서론

최근 새로운 소프트웨어 배포방식인 SaaS가 주목을 받고 있다. 기존의 소프트웨어 배포는 패키지 형태로 이루어졌다. 이런 방식의 경우, 사용자는 초기 소프트웨어의 구매를 위한 비용뿐만 아니라 구매 이후 유지보수를 위해 지속적으로 막대한 비용을 지불해야 한다. 이런 문제점을 해결하기위한 방안으로 패기지의 형식이 아닌 서비스의 형태로 소프트웨어를 배포하는 방식인 SaaS가 등장했다.

SaaS는 원활하고 효율적인 서비스 제공을 위하여 SaaS 어플리케이션과 SaaS 플랫폼의 역할을 분리한다. SaaS 어플리케이션은 사용자에게 서비스를 제공하는데 초점을 두고 운용되고, 소프트웨어 사용에 따른 과금이나 사용자의 인증, 사용자의 권한확인 등과 같은 서비스들은 SaaS 플랫폼에서 제공한다. [1][2][3]

본논문에서는 SaaS 플랫폼의 사용자 인증을 위한 인증 서비스를 설계한다. 만약, SaaS 플랫폼의 인증 서비스가 한가지로 고정되어 있다면 새롭게 SaaS 어플리케이션이 되길 희망하는 웹 어플리케이션은 자신의 사용자 인증 방식을 SaaS 플랫폼에서 제공하는 사용자 인증 서비스로 교체할 필요가 있다. 그러나 SaaS 플랫폼이 다양한 방식의 사용자 인증 서비스를 제공한다면 소프트웨어 변경없이 연동될 것이다. 하지만 SaaS 플랫폼의 인증 서비스들이 각각 독립적으로 동작이 이루어진다면, 사용자는 한 SaaS 어플리케이션에 대하여 플랫폼으로부터 인증을 받았더라도 다른 SaaS 어플리케이션을 추가 사용하고자 할 때 후자가 최초 소프트웨어가 채용하는 인증방식과 다른 방식을 적용하고 있다면 별도의 추가인증을 거쳐야 하는 불편함이 발생한다. 본 논문에서는 이런 불편들을 해소하기 위하여 다양한 사용자인증 프로토콜들을 통합하여 Single Sign On(SSO)를 제공하고 이에 따른 사용자 세션을 통합관리하는 Multi-Authentication Server를 설계한다.

2장에서는 Multi-Authentication Server에 통합되는 사용자인증 프로토콜들에 대한 간략한 소개를 하고, 3장에서는 본논문의 주제인 Multi-Authentication Server 설계에 대해 기술한다. 마지막으로 4장에서는 본논문에 대한 결론을 맺는다.

* 본 연구는 중소기업청의 "PIR/LPR/PDR 센서 기반의 물체 추적 시스템 소프트웨어 개발"과제의 지원을 받아 수행된 결과임

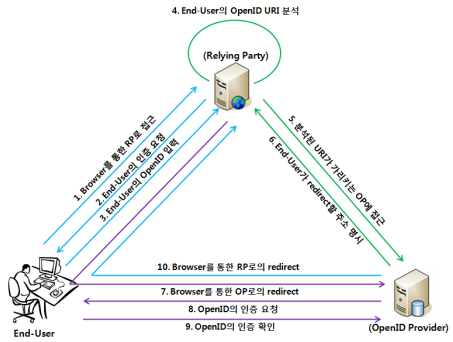
2. 관련 연구

2.1 OpenID [4][5][6][7]

OpenId 서비스는 웹사이트처럼 하나의 URI(URL)의 형태를 가진 식별자를 사용자에게 제공한다. 이 URI 형태의 OpenID로 사용자는 OpenID 인증서비스를 제공하는 웹사이트나 웹어플리케이션을 이용하여 복잡한 가입절차없이 인증을 받을 수 있다.

OpenID 서비스를 이용하기 위해서 사용자는 OpenID

Provider에 자신을 등록하여 URI 형태의 OpenID를 발급받아야 한다. 이후 사용자는 자신이 이용하고자 하는 웹 사이트(Consumer)의 요청에 따라 자신의 OpenID를 제시하여 인증절차를 진행한다. Consumer는 사용자가 입력한 URI를 분석하여 해당 OpenID를 발급 해준 Provider를 파악하고, 사용자 인증이 수행될 주소를 요청한다. Consumer는 사용자를 Provider로 리다이렉트시켜 사용자와 Provider간의 인증이 이루어지도록 한다. 사용자의 인증이 완료되면 Provider는 보안토큰 등의 정보들과 함께 사용자를 원래 있던 Consumer로 리다이렉트시킨다. 다음 [그림1]은 OpenID 프로토콜 인증절차를 나타낸다.

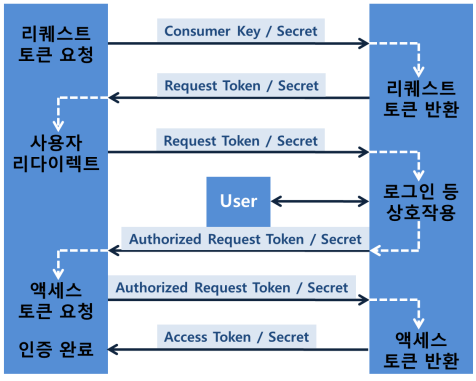


[그림1] OpenID 프로토콜 사용자 인증절차

2.2 OAuth [8]

데이터엑세스 프로토콜인 OAuth 프로토콜은 Consumer 인증이 완료된 Provider의 사용자 개인정보 또는 개인 서비스에 접근할 수 있도록 Consumer와 Provider 사이의 인증을 유도한다. 이때, OAuth는 Consumer Key와 Consumer Secret, Request Token과 Access Token을 이용하여 인증방식의 보안수준을 강화한다.

OAuth 프로토콜을 이용하기 위해서 사전에 Consumer는 Provider로 부터 발급받은 Consumer Key와 Consumer Secret을 보유하고 있어야 한다. 이 두가지 값들은 Provider에게 Consumer가 인증을 받은 대상자라는 사실을 입증한다. Consumer는 Provider에게 사용자 인증을 위한 Token을 요청할 때, 이 두가지의 Key값을 제출하여 Service Provider에게 자신이 이미 인증된 웹 어플리케이션이라는 사실을 보여준다. [그림2]는 OAuth 프로토콜 인증절차를 나타낸다.

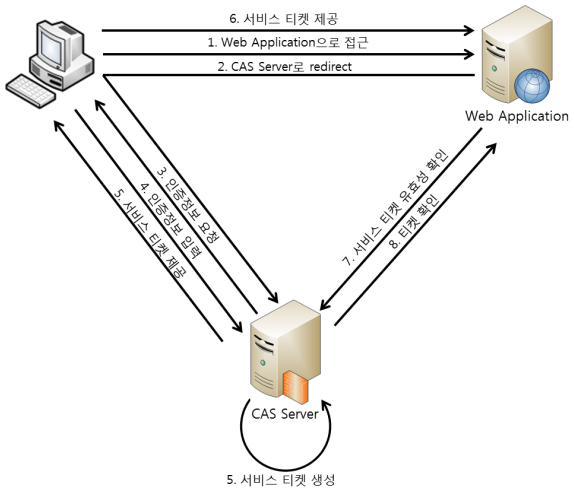


[그림2] OAuth 프로토콜 사용자 인증절차

2.3 CAS(Central Authentication Service) [9][10]

CAS(Central Authentication Service)는 웹 기반의 다수의 어플리케이션을 이용하는데 있어 SSO(Single Sign On)를 제공하기 위해 구현된 중앙인증서비스이다. CAS는 웹 어플리케이션에 접근하는 사용자에게 서비스 티켓을 발급함으로써 해당 티켓을 발급받은 사용자를 인증함과 동시에 사용자의 티켓별로 권한을 부여하는 서비스를 제공한다.

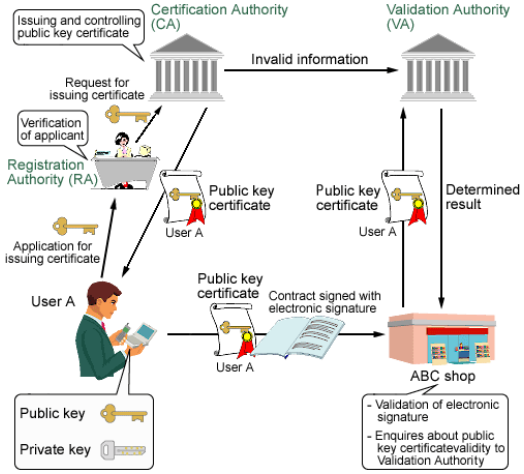
CAS 인증에는 티켓이라는 세션정보를 이용한다. 티켓 생성시 사용자 인증 결과와 해당 사용자의 권한을 첨가하여 사용자에게 제공함으로써 사용자는 어플리케이션 이용시 발급받은 티켓을 제시하여 자신을 증명한다. [그림3]은 CAS 서버를 이용한 사용자 인증 절차를 간략하게 표현한다.



[그림3] CAS 서버 사용자 인증절차

2.4 X.509 [11][12]

X.509는 ITU-T의 공개키 기반(PKI) 인증서와 인증알고리즘 표준으로 현재까지 널리 사용되고 있다. X.509 인증서에는 버전, 일련번호, 알고리즘 ID, 발행자(Issuer), 유효성, 주체, 주체 공개키 정보 등이 기록되어 있으며 이를 이용하여 PKI 인증절차를 수행하게 된다. [그림4]는 X.509 인증서를 활용한 PKI 기반의 인증절차를 나타낸다.



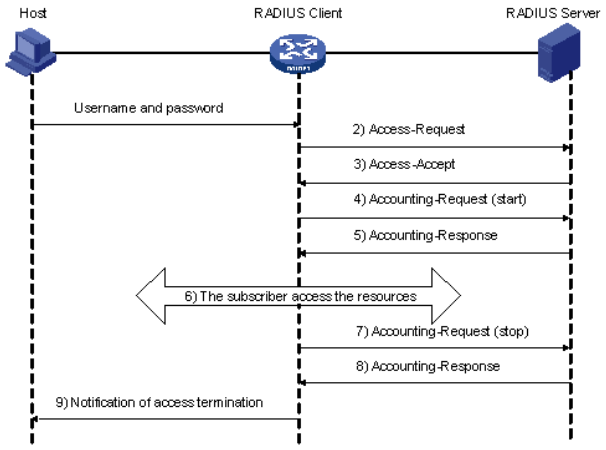
[그림4] X.509 인증서와 PKI를 통한 인증절차

2.5 RADIUS [13][14]

RADIUS는 특정 AP, Gateway 등을 통해 접속해 온 사용자를 인증하고 사용자가 요청한 시스템이나 서비스 사용에 대한 접근 권한을 부여하는 목적으로 사용되는 프로토콜이다.

최초 사용자는 인증정보로서 username과 password를 RADIUS Client에 입력한다. RADIUS Client는 입력된 사용자 정보를 이용하여 RADIUS Server로 인증요청 메시지(Access-Request)를 보낸다. RADIUS Server는 전송된 사용자의 정보와 데이터베이스에 저장된 사용자의 정보를 비교하여 일치여부에 따라 연결허가 메시지(Access-Accept)나 불허 메시지(Access-Reject)를 반환한다. Server와의 연결이 허가된 경우, 허용된 연결의 계정 정보를 RADIUS Server에 요청하여 정보를 획득한다.

[그림5]는 RADIUS프로토콜 인증절차를 나타낸다.



[그림5] Radius 프로토콜 사용자 인증절차

3. Multi-Authentication Server 설계

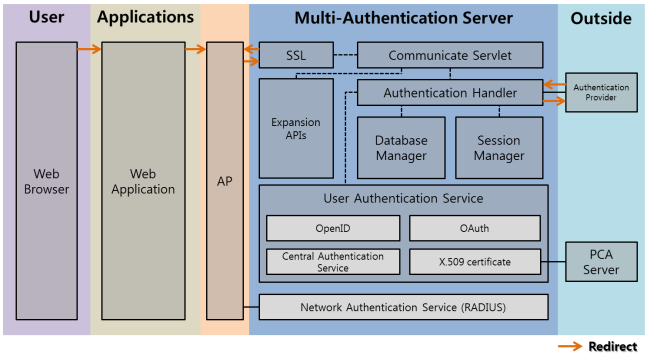
3.1 개요

본논문에서는 여러 사용자 인증방식을 통합하여 SaaS 플랫폼 사용자 인증을 위한 Multi-Authentication Server를 설계한다. Multi-Authentication Server는 OpenID, OAuth, CAS, X.509 프로토콜을 통합하여 웹기반의 사용자 인증서비스를 구현한다. 또한 웹기반의 사용자 인증의 보안성 한계 극복과, 최근 하드웨어 보안 수준의 강화 경향에 맞추어 RADIUS 프로토콜과 TPM칩을 통합한 네트워크 인증 및 하드웨어 인증서비스를 통합한 보다 강력한 인증 서비스를 제공한다.

3.2 설계

Multi-Authentication Server는 표준화된 사용자 인증 프로토콜들을 통합하여 하나의 인증 서비스를 제공한다. 따라서 어플리케이션이 해당 프로토콜 표준에 적합한 인증 서비스를 제공하고 있다면 별도의 인증모듈 수정없이 Multi-Authentication Server에 사용자 인증을 요청할 수 있다. Multi-Authentication Server 역시 표준 프로토콜에 의한 사용자인증 요청에 대해 해당 프로토콜 표준에 따라 사용자를 인증하고 그 결과를 프로토콜 표준에 따라 어플리케이션으로 반환한다. 또한 표준에 명시된 절차 이외에 사용자 인증에 있어 추가적인 정보가 필요할 경우를 대비해 추가적인 API를 제공한다.

[그림6]은 Multi-Authentication Server의 시스템 구조도 이다.



[그림6] Multi-Authentication Server 시스템 구조도

사용자 인증을 위한 프로토콜들은 하나의 모듈로 통합 관리되어 Multi-Authentication Server내에서의 사용자 인증 프로토콜 추가와 삭제의 독립성을 보장한다.

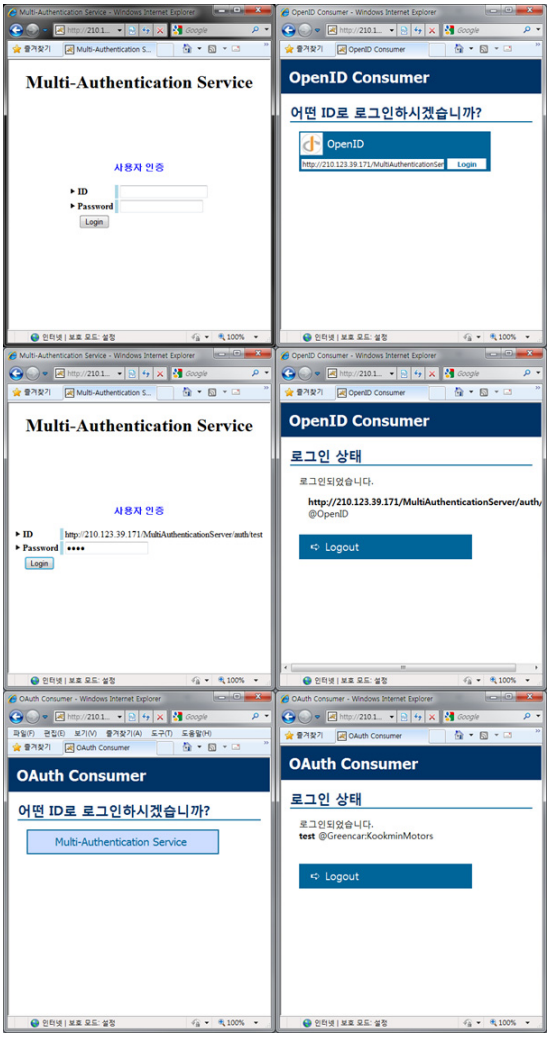
통신모듈에서 어플리케이션의 인증요청이 들어오게 되면 Authentication Handler에서 어플리케이션 요청에 해당하는 사용자인증 프로토콜을 파악하여 Authentication Service에서 해당 프로토콜의 인증 서비스를 호출한다. 호출된 인증 서비스에 따라 사용자 인증절차가 이루어지고, 사용자인증이 완료되면 Session Manager에서 Multi-Authentication Server의 인증세션을 발급한다. 모든 종류의 인증세션은 Session Manager에서 관리하여 사용자가 서로 다른 인증 프로토콜로 접근하더라도 세션 확인은 통합적으로 이루어져 반복적인 사용자인증이 일어나는 것을 방지한다.

Authentication Handler는 Multi-Authentication Server 내부에서의 사용자인증도 제어 하는데 내부 데이터베이스에 등록된 사용자의 확인과 같은 간단한 인증 방법부터 외부 Provider와의 특정 프로토콜 인증방식까지 다양한 인증방식을 제공한다. 이때, 사용되는 인증방식 역시 Authentication Service에서 통합관리된다.

마지막으로 사용자인증 액터간에 이루어지는 메시지교환은 SSL을 통해 암호화하여 보안 수준을 강화한다.

3.3 프로토타입 구현

OpenID와 OAuth 프로토콜을 하나의 인증 시스템으로 통합 구현한 통합인증시스템 1차 프로토타입을 구현하였다. 이를 기반으로 하여 여러 인증 프로토콜을 추가·통합하여 본논문에서 목표로 하는 Multi-Authentication Server를 구현한다. Java로 구현한 프로토타입의 동작예는 [그림7]과 같다.



[그림7] Multi-Authentication Server prototype snapshot

위 그림에서 사용자가 OpenID 인증방식에 의해 Multi-Authentication Server 프로토타입으로부터 인증을 받은 후, OAuth로 다시 인증을 요청할 경우 인증서버내에서의 사용자 추가인증절차를 생략하는 모습을 볼 수 있다. 이는 Multi-Authentication Server 프로토타입 내부에서 서로 다른 인증프로토콜에 대한 사용자 인증세션을 통합 관리한 결과이다.

4. 결론

웹 2.0시대가 도래하고 SaaS에 대한 개념이 확산되면서 세계적으로 SaaS에 대한 관심이 높아지고 있으며, 현재 계속적으로 서비스가 확대되고 있는 추세이다. 그 중 선두에 위치한 것이 세일즈포스닷컴이다. 세일즈포스닷컴은 다양하고 강력한 서비스를 기반으로 세계적으로 많은 사용자를 보유하고 있고, 앞으로도 그 영향력을 확대해 나갈 것으로 추정된다. 세일즈포스닷컴의 영향력이 확대됨에 따라 SaaS의 영향력 또한 증가될 것으로 보인다. 이에 따라 SaaS 플랫폼은 보다 효율적인 서비스를 제공하고, 새로운 SaaS 어플리케이션 통합에 편리함을 제공해야 할 것이다. 따라서 본논문에서 설계한 Multi-Authentication Server는 SaaS 어플리케이션과 플랫폼간의 수월한 통합과 보다 편리한 인증 절차를 거칠 수 있게 될 것이다. 본 논문에서 통합한 사용자 인증 프로토콜 통합대상은 5가지이지만, 그외의 프로토콜을 추가적으로 통합하게 된다면, 보다 다양한 프로토콜에 대응하는 인증서버로 발전시킬 수 있을 것이다. 또한 Multi-Authentication Server 내부의 사용자 인증방식 역시 다양한 프로토콜을 접합할수록 사용자 입장에서는 선택의 폭을 넓힐 수 있는 계기를 제공할 것이다. 궁극적으로 Multi-Authentication Server 사이에 사용자 인증 정보를 공유할 수 있는 프로토콜이 표준화된다면 사용자는 하나의 인증정보를 가지고 여러 SaaS 플랫폼에 연동된 다양한 SaaS 어플리케이션 서비스를 이용할 수 있을 것이다.

참고문헌

[1] 한국정보산업연합회, 웹 2.0 시대의 새로운 비즈니스SaaS (FKII 이슈 보고서)
[2] Microsoft, http://msdn.microsoft.com/architecture/saas/
[3] salesforce, http://www.salesforce.com
[4] 한재일, Nguyen Hoan An, 임승용, 정기용, 김영만, Mashed Login Access Control System 설계 및 구현, 2010 한국컴퓨터종합학술대회 논문집 Vol.37, 2010
[5] 신동휘, 전인경, 정현철, 계층적 ID를 통한 OpenID 보안 기능 강화 연구, 2009 한국컴퓨터종합학술대회 논문집, pp 10-14, 2009
[6] OpenID Authentication 2.0, http://openid.net/specs/openid-authentication-2_0.html
[7] OpenID: an actually distributed identity system, http://openid.co.kr
[8] OAuth Core 1.0 Specification, http://oauth.net/core/1.0/
[9] CAS Protocol, http://www.jasig.org/cas/protocol
[10] CAS manaual , https://wiki.jasig.org/display/CASUM/Home
[11] RFC 3280 Internet X.509 Public Key Infrastructure, IETF, 2002.4
[12] http://en.wikipedia.org/wiki/X.509
[13] RFC 5607 Remote Authentication Dial-In User Service (RADIUS) Authorization for Network Access
[14] Server (NAS) Management, IETF, 2009.7
http://en.wikipedia.org/wiki/RADIUS