Review / Artículo de revisión / Artigo do Revisão - Tipo 3

Token-Based Authentication Techniques on Open Source Cloud Platforms

Amit Banerjee, Ph.D / amit@cs.sau.ac.in

Mahamudul Hasan, MSc / rana.nstu@gmail.com

South Asian University, New Delhi, India

ABSTRACT Cloud computing is a service-oriented computational platform that allows on-demand resource provisioning for low-cost application deployment. However, security and privacy of the users is a major concern for the cloud service provider, particularly for applications handling users personal information (health record, GPS location) or performing financial transactions. Authentication is an important security measure for establishing accountability and authorization of the users, is often a prerequisite for accessing cloud-based services. In this paper, we mainly focus on the token-based authentication techniques, supported by popular open source cloud platforms [OSCPs], like Cloudstack, OpenStack, Eucalyptus and OpenNebula. In general, most OSCPs support the basic text-based user authentication. Other techniques, such as biometrics, gesture and image, can also be implemented on OSCPs. However, in this paper, we choose to discuss the token-based authentication, as it allows users to gain access to multiple cloud services with a single sign-on (SSO). Moreover, token's can be shared among multiple users for accessing cloud-based services..

KEYWORDS Mobile cloud computing; open source cloud; authentication; accountability; confidentiality; integrity.

Técnicas de autenticación basadas en tokens en plataformas de código abierto en la nube

RESUMEN El concepto de computación en la nube hace referencia al uso de una plataforma computacional externa, orientada a servicios, que permite suministrar recursos bajo demanda, a bajo costo, para el desarrollo de aplicaciones. La seguridad y privacidad de los usuarios son preocupaciones centrales de los proveedores de este tipo de servicios, particularmente cuando las aplicaciones manejan información personal reservada (como historias clínicas o ubicación geográfica) o cuando realizan transacciones financieras. La autenticación es una importante medida de seguridad para establecer cuentas y autorizar usuarios, por ellos, es un prerrequisito para el acceso a servicios basados en la nube. Este artículo se ha enfocado en las técnicas de autenticación basadas en tokens, las cuales están soportadas por plataformas en nube de código abierto muy comunes, tales como CloudStack, OpenStack, Eucalyptus y OpenNebula. Aunque la mayoría de ellas plataformas soporta la autenticación básica de usuario basada en texto, también admiten otras técnicas, tales como el uso de características biométricas, gestos e imágenes. Se selección a las técnicas de autenticación basadas en tokens para la discusión, porque ellas le permiten a los usuarios el acceso a múltiples servicios en la nube con un único inicio de sesión, y porque los tokens pueden ser compartidos entre múltiples usuarios para el acceso a servicios basados en la nube.

PALABRAS CLAVE Computación móvil en la nube; nube de código abierto; autenticación; responsabilidad; confidencialidad; integridad.

Técnicas de autenticação baseadas em tokens em plataformas de nuvem de código aberto

RESUMO O conceito de computação em nuvem refere-se ao uso de uma plataforma de computação externa, focada a serviços, que permite fornecer recursos sob demanda, a baixo custo, para o desenvolvimento de aplicações. A segurança e a privacidade dos usuários são preocupações centrais dos provedores desse tipo de serviço, especialmente quando os aplicativos lidam com informações pessoais ou privadas (como registros médicos ou localização geográfica) ou quando realizam transações financeiras. A autenticação é uma medida de segurança importante para gerenciar contas e autorizar usuários, por isso é um pré-requisito para acessar serviços baseados em nuvem. Este artigo está focado em técnicas de autenticação baseadas em tokens, que são suportadas por plataformas de nuvem de código aberto muito comuns, como CloudStack, OpenStack, Eucalyptus e OpenNebula. Embora a maioria das plataformas suporta a autenticação básica de usuário baseada em texto, elas também suportam outras técnicas, como o uso de dados biométricos, expressões faciais e imagens. Selecionamos para discussão as técnicas de autenticação baseadas em tokens, porque elas permitem que os usuários acessem múltiplos serviços na nuvem com um login único, e porque os tokens podem ser compartilhados entre vários usuários para acesso a serviços baseados em nuvem.

PALAVRAS-CHAVE Computação móvel na nuvem; nuvem de código aberto; autenticação; responsabilidade; confidencialidade; integridade.

Received / Recepción: August 6, 2018 - Accepted / Aceptación: September 4, 2018 *doi:10.18046/syt.v16i47.3211*

I. Introduction

Deployment of cloud-based applications are increasing on a large scale (Figure 1) and applications like healthcare (Thota, Sundarasekar, Manogaran, Varatharajan, & Priyan, 2018; Tang, Hu, & Hsu, 2010; Doukas, Pliakas, & Maglogiannis, 2010; Hoang & Chen, 2010; Nkosi & Mekuria, 2010); online education (Chen, Liu, Han, & Xu, 2010; Hong-qing & Yan-jie, 2010; Li (2010); mobile commerce (Alizadeh & Hassan, 2013; Yesudas, Gupta, & Ramamurthy, 2014; Yang, Pan, & Shen, 2010; online gaming and video streaming (Alizadeh & Hassan, 2013); has already gained popularity. However, providing data security is one of the major concerns in these applications. Along with the date security, maintaining confidentiality, integrity, availability, and accountability of the data (Xiao & Xiao, 2013) are also challenging tasks for the cloud service providers. Applications transmitting personal information and performing financial transactions are particularly prone to security threats (Alizadeh & Hassan, 2013).

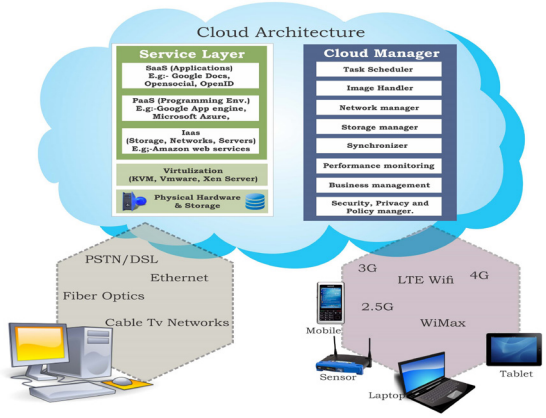

Figure 1. Architecture of cloud and mobile cloud computing / Arquitectura de computación en nube y en nube móviln (Guo, Liaw, Hsiao, Huang, & Yen, 2012)

Open-Source Cloud [OSC] computing solutions become popular to build a private cloud for user-specific demand. Due to cost-effectiveness, medium and small organizations are migrating their existing infrastructures to the OSC pen Source Cloud Platform. Open Source Cloud Platform can provide robust security features to the application. But handling different levels of user authentication can be a challenging task for OSC.

Authentication is a process of verifying the identity of an individual or an object such as a mobile device. The user has to provide its personal credentials which is then compared with the date, saved in the database (Sarvabhatla & Vorugunti, 2015; Ruj, Stojmenovic, & Nayak, 2014). In OSC, providing security and privacy to the user's authentication is essential (Grzonkowski, Corcoran, & Coughlin, 2011), as user furnish their sensitive information to the cloud. User

I. Introducción

El desarrollo de aplicaciones basadas en la nube se ha incrementado en una gran escala (ver Figura 1) y aplicaciones para el cuidado de pacientes (Thota, Sundarasekar, Manogaran, Varatharajan, & Priyan, 2018; Tang, Hu, & Hsu, 2010; Doukas, Pliakas, & Maglogiannis, 2010; Hoang & Chen, 2010; Nkosi & Mekuria, 2010); educación online (Chen, Liu, Han, & Xu, 2010; Hong-qing & Yan-jie, 2010; Li (2010);comercio en dispositivos móviles(Alizadeh & Hassan, 2013;Yesudas, Gupta, & Ramamurthy, 2014; Yang, Pan, & Shen, 2010); y juegos on line y streaming de video (Alizadeh & Hassan, 2013) han ganado una importante popularidad. Sin embargo, proveer seguridad a los datos procesados es una de las principales preocupaciones en estas aplicaciones; asimismo, mantener la confidencialidad, integridad, disponibilidad y rendición de cuentas de los datos son tareas desafiantes para los proveedores de servicios en la nube(Xiao & Xiao, 2013). Las aplicaciones que transmiten información personal y financiera son particularmente propensas a amenazas de seguridad (Alizadeh & Hassan, 2013).

Las denominadas soluciones de computación en la nube open source [OSC,OpenSource Cloud] se han convertido en procedimientos populares para la construcción de nubes privadas bajo petición de usuarios específicos. Debido a su relación costo/beneficio, las pequeñas y medianas empresas están migrando sus infraestructuras existentes hacia plataformas OSC, las cuales proveen características robustas de seguridad para las aplicaciones. Sin embargo, el manejo de diferentes niveles de autenticación de usuarios puede llegar a ser una tarea desafiante para las OSC.

La autenticación es el proceso de verificar la identidad de un individuo u objeto como un dispositivo móvil, en el cual usuario debe facilitar sus credenciales personales, las cuales son comparadas con la información almacenadas en las bases de datos (Sarvabhatla & Vorugunti, 2015; Ruj, Stojmenovic, & Nayak, 2014). En las OSC, proveer seguridad y privacidad en el proceso de autenticación de usuario es una tarea fundamental, puesto que el usuario brinda información sensible a la nube (Grzonkowski, Corcoran, & Coughlin, 2011). El acceso a los usuarios es controlado a través de autenticación adecuada, tanto como una organización tenga diferentes políticas de acceso para diferentes usuarios. Algunos de los retos de la autenticación incluyen la complejidad en proveer credenciales de usuario, el número de handshakes requeridos para la verificación y el retardo.

La mayoría de plataformas OSC soportan la autenticación tradicional basada en texto (esto es, usuario y contraseña). Este tipo de autenticación requiere de una contraseña lo suficientemente fuerte para acceder a información sensible, la cual puede ser difícil de recordar para el usuario. Además, para acceder a múltiples servicios, el usuario necesita autenticarse separadamente con diferentes credenciales, lo cual puede ser engorroso para él. La autenticación basada en token provee una solución similar. Por ejemplo, un proveedor de servicio [SP, Service Provider] puede emplear autenticación basada en token para proveer múltiples servicios a sus usuarios por medio

de una plataforma en la nube open source. El SP puede utilizar cualquier herramienta de autenticación de proveedores externos para la generación del token.

Algunas de las herramientas de autenticación popularesson SMAL 2.0 (Cantor, Kemp, Philpott, & Maler, 2005), OpenID (Recordon & Reed, 2006) y OAuth (Hardt, 2012). Para acceder a un servicio, el usuario envía una petición para un token al TPA; al recibir dicha petición, el TPA expide un token al usuario válido; dicho token permite al usuario acceder a múltiples servicios y, dependiendo de los TPA, el formato del token y el proceso de generación pueden variar. El token es almacenado en un lugar seguro (como la memoria caché) para uso futuro. Aplicaciones populares como Facebook, Twitter, Google+ y GitHub emplean tokens para sus procesos de autenticación.

En este documento se discuten las herramientas de autenticación utilizadas por los OSC enfocadas principalmente en la autenticación basada en tokens. El propósito principal es encontrar el número de handshakings requeridos para obtener un token y acceder a un servicio dentro del portafolio del proveedor; también se discute el rol del retardo en la autenticación y su dependencia en el retardo de transmisión y propagación del medio de comunicaciones. Por consiguiente, el número total de handshakes juega un rol importante en la autenticación basada en tokens. Además, en los ambientes de computación móvil en la nube [MCC, Mobile Cloud Computing], el proceso de comunicación se lleva a cabo a través de medios inalámbricos poco seguros, por ende, el objetivo de un sistema de autenticación es reducir el número de handshakes, lo que mantiene la seguridad en el sistema. También se discuten los detalles de las herramientas de autenticación basadas en tokens de proveedores externos como SAML, OpenID y OAuth y, finalmente, se presentan las vulnerabilidades de la autenticación basada en tokens.

El resto de este documento se organiza de la siguiente manera: la sección 2 presenta las características básicas de los tokens; en la sección 3 los métodos populares de autenticación de proveedores externos se discuten; en la sección 4 se consideran, en detalle, las técnicas de autenticación utilizadas en las plataformas OSC, mientras que las vulnerabilidades de seguridad de los mecanismos de autenticación basados en tokens se presentan en la sección 5; en la sección 6 se exponen las conclusiones de este trabajo.

II. Conceptos básicos de los tokens

La autenticación basada en tokens se fundamenta en la noción de "lo que tienes". Esto hace referencia a qué tipo de información posee un usuario dado para autenticarse. Básicamente, un token consiste en datos relacionados con la identidad de un usuario particular. Los usuarios pueden obtener dichos tokens a través de una combinación de nombre de usuario/contraseña, lo que les permite acceder a los recursos solicitados por un periodo específico de tiempo. Durante este periodo no se requieren métodos adicionales de autenticación. Una característica importante de los tokens es que permiten ser heredados a otros usuarios, por ello, la autenticación basada en tokens es adecuada cuando un token puede proveer acceso a múltiples servicios.

access is con- trolled through proper authentication as an organization has different access policies for different users. Some of the challenges of authentication include complexity in providing user credentials, number of handshakes required for verification and delay.

Most of OSC platforms support the traditional text-based (i.e user-name, password) user authentication. The text-based authentication required a strong password for accessing sensitive services, which may be difficult for a user to remember. Furthermore, for accessing multiple services the user needs to authenticate separately with different credentials, which may be a cumbersome task for the users. Token-based authentication provides a solution for the same. For example, a Service Provider [SP] can use token-based authentication to provide multiple services to its user via an open source cloud platform. The SP can use any third-party authentication tools for token generation.

Popular third-party authentication tools are SMAL 2.0 (Cantor, Kemp, Philpott, & Maler, 2005), OpenID (Recordon & Reed, 2006) and OAuth (Hardt, 2012). To access a service user sends a request for a token to the TPA. On receiving the request, TPA issues a token to the valid user. A token allows the user to access multiple services. Depending upon the TPAs, the format of the token and generation procedure can vary. The token is stored in the secure place (e.g cache memory) for future use. Popular applications like Facebook, Twitter, Google+ and GitHub use tokens for user authentication.

In this paper, we discuss the authentication tools used by the OSCs mainly focusing on token-based authentication. Our main aim is to find the numbers of handshaking required for obtaining a token and accessing a service form the service providers. Note that the authentication delay dependent on the transmission and propagation delay of the communication media. Thus, the total numbers of handshakes play an important role in the token-based authentication. Moreover, in Mobile Cloud Computing [MCC] environment, the data communication takes place over an unreliable wireless media. Hence, the goal of the authentication system is to reduce the numbers of handshaking, which maintains the security of the system. We also discuss the details of token-based third-party authentication tools, like SAML, OpenID, OAuth. In addition, we present the security vulnerabilities of token-based authentication.

Rest of the paper is organized as follows: In section-2 we present the basic characteristics of tokens; popular third-party authentications are discussed in section-3; in section-4 we discuss the authentications techniques used in OSC platform in details; and security vulnerabilities of token-based authentication are presented in section 5. Finally we conclude our discussion in section 6.

II. Basics of Tokens

Token-based authentication is based on the notion "what you have" that means what type of information a user has to authenticate itself. Basically token consists of data regarding the identity of a particular user. Users can get a token by providing its username/password, which allows them to access an intended resource for a specific time period. During this time period, further authentication is not needed. One more important feature is that tokens are passable. Token-based authentication is best suited, where a single token can be used for granting access to multiple services.

There are several popular token-based authentication protocols (Zwattendorfer & Tauber, 2012; Chiang, Yen, & Chen, 2013; Kang & Zhang, 2010; Hani & Dichter, 2016), as discussed below. Most popular among them are SMAL 2.0 (Cantor et al., 2005); OpenID (Recordon & Reed, 2006; Mainka, Mladenov, Schwenk, & Wich, 2017); and OAuth (Hardt, 2012; Richer & Sanso, 2017). In the following, we listed the basic overview of token generation and its properties:

Token Generation/Regeneration process

This is one of the most important aspects of token-based authentication mechanisms, generally, tokens are generated by using the following three methods:

- Using Pseudo-Random Number: In this method, the trusted server generates some random bits of specific length (Banerjee, Hasan, Rahman, & Chapagain, 2017), using a Random/Pseudo-Random Number Generator [PRNG] (Wichmann & Hill, 1982). Other popular PRNG algorithms are: Middle Square Method (Von-Neumann, 1946/1951); Linear Congruential Generator [LCG] (Lecuyer, 1999); and Cubic Congruential Generator [CCG]. The bits are then transferred to the user using the secure channel and is used as the one-time authentication key. The length of the key is an important parameter to provide security, against brute force attack. Long key implies sufficient randomness and therefore difficult to break, but on the other hand, requires more storage. Cryptographically Secure Pseudo-Random Number Generator [CSPRNG] (Blum & Micali, 1984) is best suited for this purposes. These random numbers are stored as a key for regeneration of the token.
- Using hashing: In this method, a part of the user's information like username, timestamp, client network address is provided as an input to a pre-defined hashing function to generate the token. Both, by the server and user, can use the same procedure for generating or regenerating the token. A hash function h

Existen diversos protocolos de autenticación basados en token populares, como se muestra en los trabajos de Zwattendorfer y Tauber (2012); Chiang, Yen, y Chen, (2013); Kang y Zhang,(2010); Hani y Dichter, (2016); los más populares entre ellos son: SMAL 2.0 (Cantor et al., 2005); OpenID (Recordon & Reed, 2006; Mainka, Mladenov, Schwenk, & Wich, 2017); y OAuth (Hardt, 2012; Richer & Sanso, 2017). Una sección con los fundamentos de la generación de tokens y sus propiedades se presenta a continuación.

Generación de tokens / Proceso de regeneración

Este proceso es uno de los aspectos más importantes de los mecanismos de autenticación basados en tokens. Generalmente, los tokens son generados utilizando alguno de los siguientes tres métodos:

- Utilizar números pseudoaleatorios: en este método, el servidor de confianza genera algunos bits aleatorios de longitud específica (Banerjee, Hasan, Rahman, & Chapagain, 2017) utilizando un generador de números aleatorio/pseudoaleatorio [PRNG, Pseudo Random Number Generator] (Wichmann & Hill, 1982). Otros algoritmos PRNG populares son el método de cuadrados medios (Von-Neumann,1946/1951); generador congruente lineal [LCG, Linear Congruential Generator] (Lecuyer, 1999); y generador congruente cúbico [CCG, Cubic Congruential Generator]. Los bits son entonces transferidos al usuario utilizando un canal seguro y se emplean como una llave de autenticación de único uso. El tamaño de dicha llave es un parámetro importante para garantizar la seguridad ante ataques de fuerza bruta. Llaves más largas conllevan a suficiente aleatoriedad y serán más difíciles de romper, pero por otra parte, esto conlleva a mayor almacenamiento. El generador de números pseudoaleatorios criptográficamente seguro [CSPRNG, Cryptographically Secure Pseudo-RandomNumber Generator] (Blum & Micali, 1984) es más adecuado para este propósito, puesto que los números aleatorios se almacenan como una llave para la regeneración del token.
- Utilizando hash: en este método, una parte de la información del usuario, como nombre de usuario, marca de tiempo y dirección de red, se utiliza como entrada para una función de hash (resumen criptográfico) predefinida, buscando generar el token. Ambos —el servidor y el usuario— pueden utilizar el mismo procedimiento para generar o regenerar el token. Se elige una función hash h tal que para una entrada x y su correspondiente valor de hash h(x), debería ser difícil encontrar una entrada y para la cual h(y)=h(x).
- Utilizando encriptación: en este método, algunas partes de la información del usuario son tomadas como entradas y encriptadas por llaves predefinidas, conocidas tanto por el servidor, como por el usuario. Las técnicas tradicionales de encriptación (AES, triple-DES) pueden ser utilizadas para este propósito. El procedimiento de

generación del token utilizando encriptación requiere de menos almacenamiento y menos indexación para revocar el token. Kerberos (Neuman, Yu, Hartman, & Raeburn, 2005) utiliza encriptación para generar el token.

Tamaño y formato del token

Cada herramienta de TPA posee su propio formato y tamaño de token, no hay un estándar para esto. Por ejemplo, en los tokens JSON de OAuth se almacena el ID de usuario, el alcance y alguna otra información requerida por la aplicación, mientras que el token de Kerberos contiene información como el ID del cliente, su dirección de red, el periodo de validez y la llave de sesión.

Validez del token/sesión

La validez del token depende nuevamente del TPA. Ésta es definida como el tiempo durante el cual el token puede ser usado para acceder al servicio específico. El token Kerberos usualmente presenta un periodo de validez de diez horas, pero esto puede cambiar de acuerdo con los requerimientos de seguridad de la aplicación.

Intercambio del token

En general, el token es intercambiado entre las redes existentes sin utilizar un canal seguro. Esta es la característica principal de la autenticación basada en token. El hecho de secuestrar o robar un token no garantiza una entrada no autorizada al sistema. La mayoría de las implementaciones realizan encriptación al token antes de su transmisión. Como ejemplo, en Kerberos, tanto el servidor, como el usuario, utilizan la llave privada para el proceso de encriptación/desencriptación.

III. Autenticación por fuentes externas

SAML hace referencia a lenguaje de marcado para confirmaciones de seguridad [SAML, Security Assertion Markup Language]. SAML 2.0, introducido en 2005, es la versión actual del lenguaje (Joshi & Lau, 2018; Cantor et al., 2005) y consiste en un protocolo basado en XML donde las tres principales entidades son el usuario, el proveedor de identidad [IdP, Identity Provider] y el proveedor de servicio. Un token de seguridad, conocido como aseveración, contiene la información de usuario que es intercambiada entre el IdP y el SP para validar la identidad del usuario. Permite el uso del inicio de sesión sencillo [SSO, Single Sign-On] para aplicaciones web y elimina problemas como el reúso o robo de contraseñas. SAML 2.0 utiliza los protocolos HTTP, SMTP, FTP y SOAP y requiere seis handshakes para autenticación, como se muestra en la Figura 2a. Primero, para acceder a un servicio, el usuario envía una petición al correspondiente SP y éste genera una petición SAML y redirige el navegador hacia el IdP; el IdP valida el usuario al buscar en la base de datos y envía una respuesta SAML al usuario; finalmente, el navegador envía la respuesta de regreso al SP y después de verificar dicha respuesta, el SP provee o niega el acceso al usuario.

OpenID 2.0 es un estándar abierto manejado por la Fundación OpenID, que permite al usuario identificarse en múltiples sitios web sin necesidad de procesos de registro. Diversas

is chosen such that for an input x and corresponding hash value h(x), it should be difficult to find an input y for which h(y) = h(x).

- Using encrypting: In this method, some parts of the user's information is taken as input and encrypted by using a pre-defined key, which is known to both the server and the user. Traditional encryption techniques (i.e., AES, triple-DES) can be used for this purposes. Token generation procedure using encryption needs less storage and less indexing for revoking the token. Kerberos (Neuman, Yu, Hartman, & Raeburn, 2005) uses encryption for generating the token.

Token Size and Format

Every TPA tools has its own token size and format, as there is no standard for the same. For example, in OAuth JSON token stores the user id, scope and some other information as required by the application. On the other hand, Kerberos token contains information like client ID, client network address, validity period and a session key.

Token Validity/Session

The validity of token again depends on the TPA. The validity is defined as how long the generated token can be used to access the specified service. The Kerberos token usually has a default validity of about 10 hours. But it can be changed according to the requirement of an application's security.

Token Exchange

In general, the token is exchanged through existing networks, without any secure channel. This is the main feature of token-based authentication. Merely hijacking a token does not guarantee an unauthorized entry into the system. Most implementations perform encryption on the token, before transmission. For example, in Kerberos both the server and the user uses the private key for the encryption/decryption process.

III. Third Party Authentication

SAML stands for Security Assertion Markup Language SAML 2.0 (Joshi & Lau, 2018), introduced in 2005, is the current version of SAML standard (Cantor et al., 2005). It is an XML based protocol. The three main parties of SAML 2.0 are the user, Identity Provider [IdP] and Service Provider [SP]. A security token, known as assertions, containing user information is passed between the IdP and SP for validating user's identity. It allows Single Sign-On [SSO] to web applications, thus eliminating problems like password reuse or theft. SAML 2.0 uses HTTP, SMTP, FTP and SOAP protocols and it requires six handshakes for authentication, as shown in Figure 2a. First, for accessing a

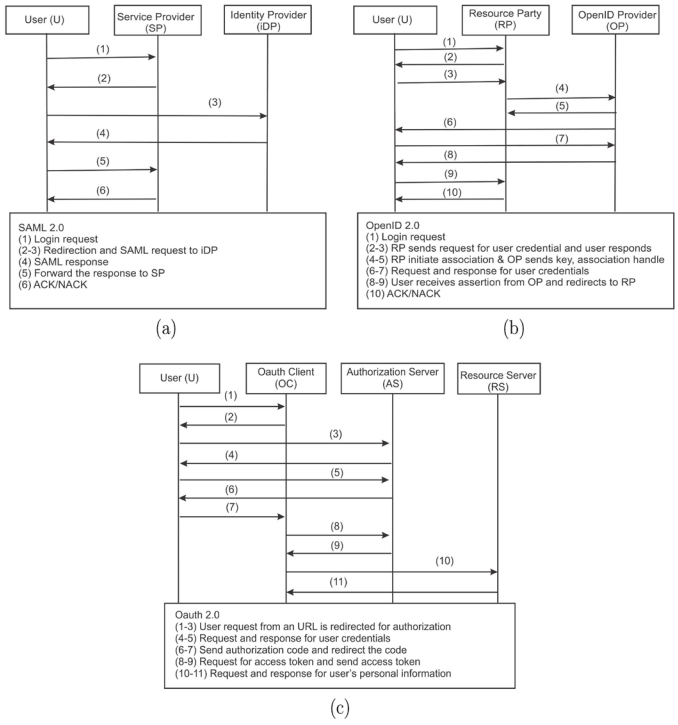

Figure 2. Token based authentication using SAML (a), OpenID 2.0 (b) and OAuth 2.0 (c)
Autenticación basada en tokens usando SAML (a), OpenID 2.0 (b) y OAuth 2.0 (c)

service, the user sends a request to the corresponding SP; then, the SP generates a SAML request and redirects the browser to IdP; the IdP validates the user by checking its database and sends a SAML response to the user; finally, the browser sends the response back to the SP and upon verifying the response, SP can provide or deny access to the user.

OpenID 2.0 is an open standard managed by OpenID foundation. It allows the user to log in multiple websites without going through their registration process. Several organization issues or accepts OpenID for user authentication, such as Google, Sun, Yahoo, BBC, IBM. Users can create an OpenID, which is an URL or eXtensible Resource Identifier [XRI] assigned by an OpenId provider [OP], later, the user can use it to access websites, known as Relying Party [RP], supporting OpenID authentication; the OP verifies user's identity and can share user's personal information (e.g. name, gender, email) with the RP after consulting with the user. The message exchanged in this protocol are shown in Figure 2b. First, to access a service, the user sends a request to the RP; the RP asks for the login credentials, to which the user can forward its OpenID (URL or XRI) for authentication; RP reads the OpenID and initiates an association with the OP; OP responds back to the RP with a key and an association handle message; Then, OP contacts the user for authentication; on receiving and verifying the login credentials, OP redirects the user to the RP with its signed assertion; finally, RP validates the assertion and creates a session for the user.

organizaciones como Google, Sun, Yahoo!, BCC, IBM, entre otras, utilizan o aceptan OpenID como estándar autenticación de usuario. El usuario puede crear una identificación denominada OpenID, la cual es una URL o identificador de recurso extensible [XRI, eXtensible Resource Identifier] asignada a un proveedor de OpenID [OP, Openid Provider]. Después, el usuario puede utilizar dicho OpenID para acceder a sitios web que soportan autenticación OpenID, un proceso conocido como entidad de confianza [RP, Relying Party],

El OP verifica la identidad del usuario y puede compartir información personal de él (e.g., nombre, género, correo electrónico) con el RP, después de consultar con el usuario. Los mensajes intercambiados en este protocolo se muestran en la Figura 2b. Primero, para acceder a un servicio, el usuario envía una petición al RP y éste pregunta por las credenciales de acceso para que el usuario pueda reenviar su OpenID (URL o XRI) para autenticación; el RP lee el OpenID e inicia la asociación con el OP, éste responde al RP con una llave y un mensaje de manejo de asociación; entonces, el OP contacta al usuario para su autenticación o recibe y verifica las credenciales de ingreso, al hacerlo, el OP redirige al usuario hacia el RP con la afirmación firmada; finalmente, el RP valida la afirmación y crea una sesión para el usuario.

Al contrario de SAML u OpenID, OAuth 2.0 es un protocolo de autorización de estándar abierto, aunque puede ser utilizado para realizar una pseudo-autenticación. Al utilizar OAuth, el dueño de un recurso o el usuario final pueden proveer acceso limitado a una página web externa o a una aplicación conocida como cliente OAuth [OC, Oauth Client] para

recuperar su información personal almacenada en un servidor de recursos [RS, Resource Server] al utilizar un token facilitado por un servidor de autorización OAuth [AS, OAuth Authorization Server]. El AS asume que el usuario confía en el OC con sus datos y permite al OC acceder a sus APIs. OAuth se utiliza comúnmente en conjunto con un protocolo de autenticación (e.g., OpenID Connect 1.0). La versión 2.0 de este protocolo fue liberada en 2012 por la IETF en el RFC6749 (Hardt, 2012). Como se muestra en la Figura 2c, el usuario que trata de acceder a una aplicación o página web externa recibe una petición de autorización desde el OC es redirigido hacia el AS; al recibir la petición, el AS pregunta por las credenciales de usuario y verifica la autenticidad de este; si este proceso es exitoso, el AS envía un código de autorización al usuario y éste es redirigido al OC; al proveer el código de autorización, el OC recibe un token de acceso desde el AS; y finalmente, el OC puede acceder a información personal del usuario desde el RS utilizando este token.

X.509 (Bauer, 2012; Kim & Timm, 2014; Arifeen, Siddiqui, Ashraf, & Waheed, 2015) es un estándar de la UIT para infraestructura de llaves públicas [PKI, Public Key Infrastructure] que describe el formato de un certificado digital que asocia un sujeto (usuario, máquina o servicio) con su llave pública. Los certificados digitales de X.509 son emitidos por un TTP denominado autoridad certificadora [CA, Certification Authority], la cual es además responsable de la renovación y la anulación de certificados. Un certificado digital contiene información como: el nombre del sujeto, su llave pública, el periodo de validez y la firma, como se muestra en la Figura 3. El nombre del sujeto contiene el nombre distinguido [DN, Distinghished Name] de una entidad o del dueño de la llave pública del sujeto; y el campo de la firma tiene tres secciones: a) todos los restantes campos del certificado; b) un compendio encriptado con la llave privada del CA; y c) los algoritmos y parámetros descritos en b).

Para la verificación del certificado, un sujeto necesita conocer la llave pública del CA, por medio de la cual puede desencriptar el campo de firma en el certificado y verificar la llave pública del sujeto al compararla con la certificada por el CA. Los certificados digitales de X.509 son también empleados para autenticación mutua y SSO.

Kerberos, descrito en el RFC4120 (Neumanet al., 2005) es un protocolo de autenticación basado en criptografía de llave secreta. Una entidad externa de confianza [TTP, Trusted Third Party] se emplea en Kerberos para realizar una autenticación mutua y SSO a través de una red insegura (Le, Truong, Van, & Le, 2015; Al-Janabi & Rasheed, 2011; Khandelwal & Kamboj, 2015). Fue originalmente desarrollado por el MIT en la década de 1980, cuando Kerberos v5 apareció como RFC1510, y fue reemplazado después, en 2005, por el RFC 4120. El KDC actúa como un TTP, responsable de mantener las llaves secretas de todos los clientes y servicios siguiendo el par de mensajes que son intercambiados en el protocolo Kerberos, donde un cliente trata de acceder servicios desde los SP.

- En (1-2) de la Figura 4, el cliente envía una petición

OAuth 2.0, unlike SAML or OpenID are open standard authorization protocol. Although, they can be used to perform pseudo-authentication. Using OAuth, a resource owner or an end-user can grant limited access to a third party website or an application, referred as OAuth Client [OC], to retrieve its personal data stored in a Resource Server [RS] by using token provided by an OAuth Authorization Server [AS]. The AS assumes that the user trusts OC with its data and allows the OC to access its APIs. OAuth is commonly used in conjunction with an authentication protocol (e.g. OpenID Connect 1.0). Version 2.0 of this protocol was released in 2012 by IETF as RFC 6749 (Hardt, 2012). As shown in Figure 2c, the user trying to access a third party website/application receives an authorization request from OC, which gets redirected to AS; on receiving the request, AS asks for user's credentials and verifies the authenticity of the user; on successful verification, AS sends an authorization code to the user and gets redirected to the OC; by providing the authorization code, OC gets an access token from the AS; finally, using this token, OC can access user's personal data from the RS.

X.509 (Bauer, 2012; Kim & Timm, 2014; Arifeen, Siddiqui, Ashraf, & Waheed, 2015) is an ITU-T standard for a Public Key Infrastructure [PKI], describing the format of a digital certificate that associates a subject (user, machine or service) with its public key. X.509 digital certificates are issued by a TTP, called the Certification Authority [CA]. The CA is also responsible for certificate renewal and revocation. A digital certificate contains information such as subject's name, subject's public key, a period of validity and a signature, as shown in Figure 3. The subject name field contains the Distinguished Name [DN] of an entity or the owner of subject's public key. The signature field has three sections: a). all other fields in the certificate; b). a digest encrypted with CA's private key; and c). the algorithms and parameters used in b. For verifying the certificate, a subject needs to know the public key of the CA, whereby it can decrypt the signature field in the certificate and verify the subject's public key by comparing with the one certified by the CA. X.509 digital certificates are also used for mutual authentication and SSO.

Kerberos, RFC 4120 (Neuman et al., 2005) is an authentication protocol based on secret-key cryptography. A Trusted Third Party [TTP] is used in Kerberos to perform mutual authentication and SSO over an unsecured network (Le, Truong, Van, & Le, 2015; Al-Janabi & Rasheed, 2011; Khandelwal & Kamboj, 2015). It was originally developed by MIT in the 1980s. Kerberos v5 appeared as RFC 1510 in 1993 and was later replaced by RFC 4120 in 2005. Figure 4 shows the entities of Kerberos with AS and TGS as two important components of KDC. The KDC acts as a TTP,

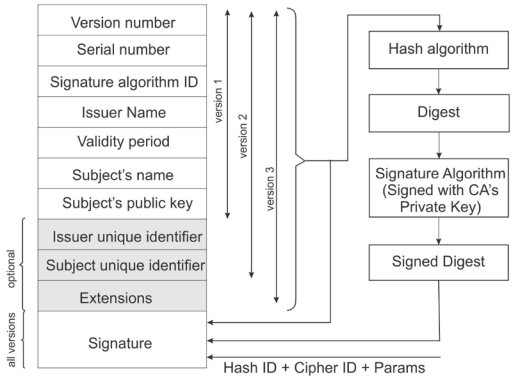

Figure 3. X.509 Certificate Format / Formato certificado X.509

responsible for maintaining the secret keys of all clients and services; following the pair of messages are exchanged in the Kerberos protocol, when a client tries to access service from SP.

- In (1-2) of Figure 4, the client sends a request in plain text to the AS for a Ticket-Granting Ticket [TGT]; on receiving the request, AS randomly generates a secret key, client/TGS session key (KC-TGS) shared between client and TGS and sends two messages to the client: Message (a) contains the key KC−TGS and is encrypted with the client's secret key, and message (b) is the TGT, containing client ID, TGS ID, timestamp, TGT lifetime and KC−TGS encrypted with the secret key of TGS.
- In (3-4) of Figure 4, the client retrieves KC−TGS by decrypting the message (a) using its secret key and generates a new message (c), called authenticator, containing client ID and a timestamp; then, it encrypts (c) with key KC−TGS and sends messages (b) and (c) to the TGS. The TGS decrypts the message (b) with its secret key, gets the session key KC −T GS and uses it to decrypt the message (c). TGS compares the client ID and timestamp presents in the messages (b) and (c) and, if they match, TGS randomly generates a client/SP session key (KC−SP) and sends messages (d) and (e) to the client. Message (d) is the client-to-server ticket containing client ID, timestamp, KC−SP and is encrypted with the service's secret key; message (e), containing KC−SP, is encrypted by key KC−T GS .
- Finally in Figure 4 (5-6), mutual authentication takes place between the client and the SP. Client decrypt (e) using key KC−TGS and creates a new message (f) containing client ID, timestamp. Then (f) is encrypted using key KC−SP and messages (d) and (f) are sent to the SP; the SP decrypts message (d) using its service's secret key and uses KC−SP for decrypting

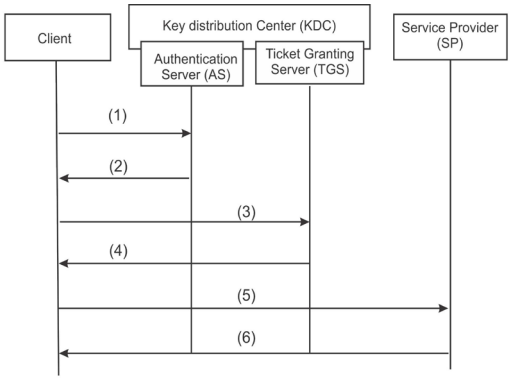

Figure 4. Basic Kerberos 5 Authentication Protocol / Protocolo básico de autenticación de Kerberos 5

en texto plano al AS para realizar el proceso de Ticket-Granting Ticket [TGT]; al recibir la petición, el AS genera aleatoriamente una llave secreta denominada llave de sesión de cliente/TGS (KC-TGS), la cual es compartida entre el cliente y el TGS; además, el AS envía dos mensajes al cliente: el mensaje (a) contiene la llave KC−TGS y es encriptada con la llave secreta del cliente; mientras que el mensaje (b) es el TGT que contiene el ID del cliente, el ID del TGS, la marca de tiempo, el tiempo de vida del TGT y el KC−TGS encriptado con la llave secreta del TGS.

- En (3-4) de la Figura 4, el cliente recupera el KC−TGS al desencriptar el mensaje (a) utilizando su llave secreta y genera un nuevo mensaje (c) llamado autenticador, que contiene el ID del cliente y la marca de tiempo; entonces, procede a encriptar el mensaje (c) con la llave KC−TGS y envía los mensajes (b) y (c) al TGS. Este último desencripta el mensaje (b) con su llave secreta y obtiene la llave de sesión KC−TGS, la cual utiliza para desencriptar el mensaje (c). El TGS compara el ID del cliente y la marca de tiempo presentes en los mensajes (b) y (c) y, si son iguales, genera aleatoriamente una llave de sesión de cliente/SP (KC−SP) y envía los mensajes (d) y (e) al cliente. El mensaje (d) es el ticket de cliente a servidor que contiene el ID del cliente, marca de tiempo, KC−SPy es encriptado con la llave secreta del servicio. El mensaje (e) que contiene el KC−SP es encriptado por la llave KC−T GS.
- Finalmente, en la Figura 4 (5-6), la autenticación mutua se ejecuta entre el cliente y el SP. El cliente desencripta el mensaje (e) utilizando la llave KC−SP y crea un nuevo mensaje (f) que contiene el ID del cliente y la marca de tiempo. Entonces, dicho mensaje (f) es encriptado utilizando la llave KC−SP y los mensajes (d) y (f) son enviados al SP; este último desencripta el mensaje (d) utilizando su llave secreta de servicio y utiliza KC−SP para desencriptar (f); el SP verifica la integridad del mensaje, al comparar el ID del cliente y la marca de tiempo en los dos mensajes, y manda un nuevo mensaje (g) al cliente. Este mensaje (g) va encriptado utilizando KC−SP y

contiene la marca de tiempo del mensaje (f). El cliente desencripta (g) utilizando KC−SP y verifica la validez de la marca de tiempo.

IV. Autenticación para la nube *open source*

En esta sección se discuten los procedimientos para autenticación de algunas plataformas open source en la nube: CloudStack(2018), OpenStack (2018), OpenNebula(2018) y Eucalyptus (2018). Las OSC ofrecen la infraestructura como servicio [IaaS, Infrastructure-as-a-Service] en un ambiente privado en la nube y son soluciones escalables, flexibles y efectivas respecto de plataformas comerciales o públicas en la nube, tales como Amazon AWS, Sales Cloud de Salesforce, App Engine de Google o Microsoft Azure. Las preocupaciones de seguridad acerca de plataformas públicas en la nube son relativamente mayores a las de las OSC por varios motivos: una nube pública es una caja negra para los clientes y éstos deben confiar en la seguridad de estas antes de utilizar el servicio. Al contrario, las partes internas de las OSC pueden ser estudiadas, cambiadas y actualizadas de acuerdo con los requerimientos de los clientes. Segundo, los OSC al ser una plataforma privada en la nube, reduce los riesgos internos y externos de seguridad, al contrario de las plataformas públicas.

El esquema de autenticación tradicional de usuario y contraseña se utiliza comúnmente en todas las plataformas OSC. Para esto, generalmente se instalan componentes de autenticación en el front-end y en el back-end en máquinas separadas. Los usuarios introducen sus credenciales a través del front-end (una interfaz de usuario online como Dashboard en OpenStack) y ella es verificada por el componente de back-end (Keystone en OpenStack). Adicionalmente, la mayoría de OSC ofrecen otros mecanismos para autenticar sus usuarios. La autenticación de terceros como SAML, OpenID 2.0 y OAuth 2.0 pueden ser utilizadas en los OSC. Los usuarios también tienen la posibilidad de utilizar API o plugins de OSC para diseñar sus propios sistemas de autenticación. La mayoría de OSC —como CloudStack y OpenStack— , soporta el uso del protocolo LDAP para autenticación (discutido a continuación).

LDAP [Lightweight Directory Access Protocol] es un estándar de la IETF, se trata de un protocolo de aplicación basado en el paradigma cliente-servidor para acceder y modificar directorios de servicios a través de redes TCP/IP. LDAP es un derivado del protocolo de acceso a directorios X.500, conocido como X.500 lite. Un servidor LDAP mantiene una estructura jerárquica para entradas (objetos) denominada árbol de información de directorios [DIT, Directory Information Tree]. Cada entrada se identifica por un DN, que es una ruta completa desde la raíz (root) a la entrada en el DIT; un cliente LDAP utiliza DN para enviar una petición (buscar, añadir, borrar, modificar) a un servidor LDAP, y este último consulta el DIT para DN y responde de acuerdo con la petición. Sin embargo, un cliente LDAP debe autenticarse y contar con privilegios para el acceso a los servicios desde el servidor LDAP. La autenticación LDAP puede llevarse a cabo internamente utilizando la operación de amarre (bind) o externamente vía Kerberos o certificados X.509. Los servidores LDAP pueden

(f). The SP ascertain the message integrity by comparing client ID and timestamp values in the two messages; then, SP sends a new message (g) back to the client, encrypted using KC−SP, containing the timestamp from message (f). The client decrypts (g) using KC−SP and checks for the validity of the timestamp.

IV. Authentication for Open Source Cloud

In the following section, we discuss the authentication procedure of some popular Open Source Cloud [OSC] platforms: CloudStack (2018), OpenStack (2018), OpenNebula (2018) and Eucalyptus (2018). The OSCs offer Infrastructure-as-a-Service [IaaS] in a private cloud environment and is a scalable, flexible and cost-effective alternative to the commercial or public clouds, such as Amazon's AWS, Salesforce's Sales Cloud, Google's App Engine or Microsoft Azure. The security concerns about a public cloud platform are relatively bigger than OSCs (Ristov & Gusev, 2013), because: first, a public cloud is a black-box for the customers and they need to trust the security of the CSP before using their service, instead, the internals of an OSC can be studied, changed and updated according to the requirements of the customers; and secondly, OSC being a private cloud platform reduces the internal and external security risks, which is in contrary to the public clouds.

The traditional username password-based authentication is commonly used in all OSC platforms. For this, generally a frontend and a backend authentication components are installed on separate machines. Users enter their login credentials through the frontend (an online user interface, such as Dashboard in OpenStack), and it is verified by the backend component (Keystone in OpenStack). In addition, most OSC offers other mechanisms for authenticating its users. Third party authentication, like SAML, OpenID 2.0 and OAuth 2.0 can be used in OSC. Users can also use APIs and plugins of OSC to design their own authentication system. Most OSC, like CloudStack and OpenStack, supports the use of Lightweight Directory Access Protocol [LDAP] for authentication, which is discussed below.

The LDAP is an IETF standard, a client-server based application protocol for accessing and modifying directory services over a TCP/IP network. LDAP is a derivative of X.500 Directory Access Protocol [DAP] and is also known as X.500 lite. An LDAP server maintains a hierarchical structure for entries (objects), called Directory Information Tree [DIT]. Each entry is identified by a Distinguished Name [DN], which is a complete path from the root to the entry in DIT. An LDAP client uses DN to send a request (search, add, delete, modify) to an LDAP server and the LDAP server consults the DIT for DN and responds ac-

cordingly. However, an LDAP client must authenticate and should have privileges for accessing services from the LDAP server. LDAP authentication can be performed internally using LDAP bind operation or externally via Kerberos or X.509 certificates. LDAP server can also perform authentication on behalf of a web server or email server by using a Pluggable Authentication Module [PAM].

RFC 4513 describes the modes in which an LDAP client can authenticate and access service from an LDAP server. For authentication, the bind operation (RFC 4511) is used for exchanging information between LDAP client and server. Four types of authentication in LDAP are anonymous, unauthenticated, name/password and SASL authentication. The first three are collectively referred to as simple authentication methods; for anonymous authentication, LDAP client sends a bind request to the LDAP server with no DN or password information, it provides limited access (read-only) to the server. Unauthenticated authentication is similar to anonymous authentication, in this, the LDAP client establishes an anonymous state by presenting DN in the bind operation, which is used for logging; the name/password authentication can establish an authenticated authorization state (Figure 5), however, the method is not secure as the password is sent in clear text. LDAP uses SASL framework [Simple Authentication and Security Layer] (RFC 4422) for secure authentication and data transfer. There are several SASL authentication mechanisms, like DIGEST-MD5, CRAM-MD5, GSS- API, Kerberos v4. A series of server challenge (BindResponse) and client response (BindRequest) messages are exchanged for completing a SASL authentication process. In the following section, we will discuss the different open source cloud's authentication methods in details.

Cloudstack

CloudStack is an open source cloud computing platform, developed by the Apache Software Foundation [ASF], for delivering IaaS in a private, public or hybrid cloud environment. The development of CloudStack was originally initiated by Cloud.com. In 2011, Citrix purchased Cloud.com and released the software completely under GNU General Public License (GPLv3); in 2012, the software was relicensed by Citrix under Apache Software License 2.0 (ASLv2) and it was accepted into the Apache Incubator. The first major release of the CloudStack software (version 4.0.0- incubating) was released towards the end of 2012; the current stable version of CloudStack (2018) is 4.11.0.0, released in January 15, 2018. In this section, we mainly focus on the authentication techniques supported by the CloudStack.

CloudStack provides two native methods for authentication, which is based on username/password and

también llevar a cabo autenticación en nombre de un servidor o servicio de email utilizando un módulo de autenticación conectable [PAM, Pluggable Authentication Module].

El RFC 4513 describe los modos en los cuales un cliente LDAP puede autenticarse y acceder a los servicios desde un servidor LDAP. Para autenticación, la operación de binding (RFC 4511) se utiliza para intercambiar información entre el cliente y servidor LDAP. Los cuatro tipos de autenticación en LDAP son: anónimo, sin autenticar, nombre/contraseña y autenticación SASL. Los primeros tres se refieren colectivamente a simples métodos de autenticación; para autenticación anónima, el cliente LDAP envía una petición bindal servidor LDAP sin información de DN o contraseña, brindando acceso limitado (de sólo lectura) al servidor. La autenticación sin autenticar es similar a la anónima, en ella, el cliente LDAP establece un estado anónimo al presentar el DN en la operación de bind, el cual es usado para el acceso. La autenticación por usuario/contraseña puede establecer un estado de autorización autenticado (ver Figura 5), sin embargo, el método no es seguro puesto que la contraseña se envía en texto plano. LDAP utiliza el framework SASL (de Simple Authentication and Security Layer) (RFC 4422) para la autenticación segura y transferencia de datos. Existen diversos mecanismos de autenticación vía SASL como DIGEST-MD5, CRAM-MD5, GSS-API y Kerberos v4. Una serie de mensajes del servidor (bindresponse) y respuestas del cliente (bindrequest) se intercambian para completar el proceso de autenticación SASL. En la siguiente sección se discuten los distintos métodos de autenticación de OSC en detalle.

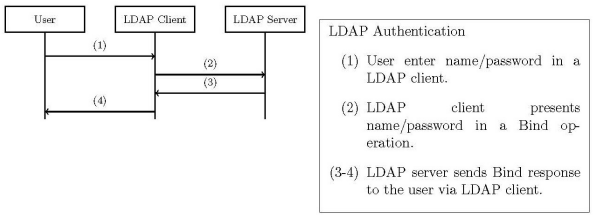

Figure 5. Simple Authentication in LDAP
Autenticación simple en LDAP

Cloudstack

CloudStack es una plataforma de computación en la nube open source desarrollada por la Fundación Apache [ASF, Apache Software Foundation], para proveer IaaS en un ambiente de nube privada, pública o híbrida. El desarrollo de CloudStack fue originalmente iniciado por Cloud.com. En 2011, Citrix adquirió Cloud.com y lanzó al mercado el software completamente bajo la licencia GNU; en 2012, el software fue re-licenciado por Citrix bajo la licencia de software de Apache [ASL, Apache Software License] 2.0 y aceptado en el incubador de Apache. Al final de 2012 se lineró la primera gran versión CloudStack (versión 4.0.0 – incubada). La versión actual de CloudStack (2018) es la 4.11.0.0, liberada en enero de 2018. Este documento se enfocó en las técnicas de autenticación soportadas por CloudStack.

CloudStack provee dos métodos nativos de autenticación, ambos basados en usuario/contraseña y firmas HMAC. La

autenticación y autorización en CloudStack se gestiona a través del componente de control de acceso [ACC, Access ControlComponent], el cual es parte del servidor de mantenimiento (Sabharwal& Shankar, 2013). Los usuarios pueden enviar su solicitud de acceso a través del portal web de CloudStack usando la línea de comandos o mediante llamadas a la API (como se muestra en la Tabla 1). Al recibir la petición, el ACC autentica al usuario y determina los privilegios de acceso con base en el dominio, proyecto u otra información grupal del usuario; finalmente, el ACC emite un token de autenticación indicando los privilegios de acceso dados al usuario y almacena la información en un archivo log.

Los servidores LDAP, como el directorio activo de Microsoft o ApacheDS, pueden ser utilizados para autenticación externa. CloudStack provee API para conectarse a un servidor LDAP, buscar en el DIP del servidor y buscar información del usuario, como: nombre completo, email, nombre de usuario, etc. Para autenticación, el usuario introduce su nombre de usuario y contraseña a través de la interfaz web de CloudStack o a través de comandos; CloudStack busca dicho usuario en el DIT del servidor LDAP, y si tiene éxito autentica al usuario al iniciar una operación de bind con el DN y la contraseña del mismo. Las APIs de CloudStack para LDAP son listLdap-Configurations, addLdap-Configuration, deleteLdapConfiguration, linkDomainToLdap, listLdapUsers, importLdapUsers y ldapCreateAccount.

OpenStack

OpenStack (2018) es una plataforma open source en la nube manejada por la Fundación OpenStack disponible libremente bajo la licencia Apache 2.0. Provee IaaS a sus usuarios en ambientes de nubes privadas y públicas. La última versión de

HMAC signature. The authentication and authorization in CloudStack is handled by the Access Control Component [ACC], which is a part of the management server (Sabharwal & Shankar, 2013). Users can submit their login request through CloudStack's web portal, Command Line Interface [CLI] or through API calls (as shown in Table 1); on receiving the request, ACC authenticates the user and determines the access privileges based on the domain, project and other group information of the user; finally, the ACC issues an authentication token indicating the access privileges granted to the user. ACC also saves this information in a log file.

The LDAP servers, such as Microsoft Active Directory or ApacheDS, can be used for external authentication. CloudStack provides APIs to connect to an LDAP server, search the LDAP Directory Information Tree (DIT) and fetch user information like first/last name, email, username. For authentication, the user enters the username and password through the CloudStack web interface or CLI. CloudStack searches for the username in the DIT of an LDAP server; if successful, it authenticates the user by initiating a bind operation with the Distinguished Name (DN) and password of the user. CloudStack APIs for LDAP are: listLdapConfigurations, addLdap- Configuration, deleteLdapConfiguration, linkDomainToLdap, listLdapUsers, importLdapUsers and ldapCreateAccount.

OpenStack

OpenStack (2018) is an open source cloud platform managed by the OpenStack Foundation and is freely available

Table 1. Authentication APIs of CloudStack
API de autenticación en CloudStack (2018)

API	Functionalities / Función
login	Used for logs a user into the CloudStack. After successful log in JSESSIONID cookie is generated and passed the value for the consecutive queries until the logout or the expiration of the session.
	Sirve para iniciar la sesión de un usuario en CloudStack. Después de un inicio de sesión exitoso, se genera la cookie JSESSIONID y se pasa el valor de las consultas consecutivas hasta el cierre o la expiración de la sesión.
logout	Logs out the user.
	Cierra la sesión del usuario
samlSso	Service Provider initiated SAML Single Sign ON API.
	Inicio de sesión único en la API del proveedor de servicios
samlSlo	SAML Global Log Out API
	API de cierre de sesión global de SAML
getSPMetadata	Returns SAML2 CloudStack Service Provider Metadata
	Presenta los metadatos del proveedor de servicios de SAML2 CloudStack
listIdps	Returns list of discovered SAML Identity Providers
	Presenta la lista de proveedores de identidad SAML descubiertos
authorizeSamlSso	Allow or disallow a user to use SAML SSO
	Permite o no a un usuario usar SMAL SSO
listSamlAuthorization	Lists authorized users who can used SAML SSO
	Lista los usuarios autorizados para usar SAML SSO

under Apache License 2.0. It provides IaaS to its users in a public and private cloud environment. The latest version of OpenStack is Queens, released on 30 August 2017. OpenStack has a modular architecture, integrating services from various components through their APIs. The core services of OpenStack includes compute, dashboard, block storage, object storage, identity, network and image services (Ismaeel, Miri, Chourishi, & Dibaj, 2015). The compute service (Nova) is responsible for managing the lifecycle of the virtual machines. The storage requirement for saving unstructured data objects and for running instances are handled by the object (Swift) and block (Cinder) storage, respectively. The identity service (Keystone) manages authentication and authorization. All components of OpenStack interact with each other via the network service (Neutron) and the dashboard (Horizon) is the web interface through which administrators or cloud tenants can interact with OpenStack services.

As discussed above, the authentication and authorization in OpenStack is handled by a central identity service named Keystone, who maintains the list of valid users and services that they can access. OpenStack can integrate various authentication techniques though keystone, like user- name/ password based, token-based or can authenticate users through an existing directory service like LDAP. Keystone plugins can be used to perform third-party authentication like OAuth, SAML, OpenID and X.509.

OpenStack APIs can be used to create projects (tenants), users, and roles. A project is the logical grouping of users and a role is the set of rights and privileges. After registration, users can submit their authentication request through dashboard (GUI) or using CLI/API calls. The user credentials for authentication can be passed using the environment variables or via the CLI options, for example, URL of an Identity endpoint can be specified with CLI variable --os-auth-url or by using OS-AUTH-URL environment variable. Keystone can be configured to use authentication methods in various domains.

Keystone supports various token providers for authentication, such as Universally Unique IDentifier [UUID] and Public Key Infrastructure [PKI]. Each token differs in technology, scalability and architectural requirements. For UUID token, users send their credential to the Keystone server; the keystone server validates the user; on successful verification, the server generates and sends a UUID token back to the user; the token is passed along with each subsequent API requests from the user; the keystone server verifies the token and sends an appropriate response to the user. A limitation of UUID token is that it generates heavy traffic load on the Keystone server. The PKI token, introduced in OpenStack since 2013, whereby the Keystone server acts

OpenStacks se denomina Queens y fue lanzada en agosto de 2017, ella presenta una arquitectura modular que integra servicios de varios componentes a través de sus API. Los servicios básicos de OpenStacks incluyen: cómputo, tableros de mando (dashboards), almacenamiento de bloques, almacenamiento de objetos, identidad, servicios de imagen y red (Ismaeel, Miri, Chourishi, & Dibaj, 2015). El servicio de cómputo (Nova) es responsable de mantener el ciclo de vida de las máquinas virtuales; el requerimiento de almacenamiento para guardar objetos de datos sin estructurar y para ejecutar instancias se maneja por el almacenamiento de objetos (Swift) y de bloques (Cinder), respectivamente; el servicio de identidad (Keystone) gestiona la autenticación y autorización. Todos los componentes de OpenStack interactúan entre sí a través del servicio de red (Neuron) y el dashboard (Horizon) es la interfaz web a través de la cual los administradores o arrendatarios de la nube pueden interactuar con los servicios de OpenStack.

Tal como se mencionó, la autenticación y autorización en OpenStacks es manejada por un servicio de identidad central llamado Keystone, el cual mantiene la lista de usuarios válidos y de los servicios a los que pueden acceder. OpenStacks puede integrar varias técnicas de autenticación a través de Keystone, como las basadas en usuario/contraseña, las basadas en tokens e incluso puede autenticar usuarios a través de un servicio de directorio existente como LDAP. Los plugins de Keystone pueden ser utilizados para realizar la autenticación de fuentes externas como OAuth, SAML, OpenID y X.509.

Las API de OpenStack pueden ser utilizadas para crear proyectos, usuarios y roles. Un proyecto es el agrupamiento lógico de usuarios y un rol es el conjunto de derechos y privilegios. Después del registro, los usuarios pueden enviar su petición de autenticación a través del dashboard (GUI) o utilizando llamadas a la línea de comandos o API. Las credenciales de usuario para la autenticación pueden transferirse utilizando variables de entorno o vía opciones en la línea de comandos. Por ejemplo, la URL del extremo de una identidad puede ser especificada a través de la variable en línea de comandos --os-auth-url o utilizando la variable de ambiente OS-AUTH-URL. Keystone se puede configurar para utilizar métodos de autenticación en varios dominios.

Keystone soporta varios proveedores de tokens para autenticación, tales como identificador único universal [UUID, Universally Unique IDentifier] e infraestructura de llave pública [PKI, Public Key Infrastructure]; cada token varía en tecnología, escalabilidad y requerimientos de arquitectura. Para el token UUID, los usuarios envían sus credenciales al servidor Keystone; éste valida al usuario y si la verificación es exitosa, el servidor genera un token UUID al usuario; dicho token es enviado junto con cada llamado subsiguiente a la API desde el usuario; el servidor Keystone verifica el token y envía la respuesta apropiada al usuario. Una limitación de este tipo de tokens es que genera una elevada carga de tráfico en el servidor Keystone. El token PKI —introducido en OpenStack desde 2013—, actúa como la autoridad certificadora [CA, Certificate Authority] a través del servidor Keystone y emite un certifi-

cado y firma el token de usuario usando su llave secreta. Cada extremo de la API, junto con el certificado Keystone, la lista de revocación y la llave firmada, pueden validar el token PKI enviado al usuario independientemente.

Khan, Ylitaloy Ahmed (2011) integran OpenStack y OpenID para proveer "autenticación como un servicio" y el servicio single-sign-on a sus usuarios; en el algoritmo propuesto, las API de autenticación se comunican con el usuario a través de la GUI y en el back-end, el proceso de autenticación ocurre a través de un punto de decisión de políticas [PDP, Policy Decision Point].

OpenNebula

OpenNebula (2018; Ristov & Gusev, 2013; Endo, Goncalves, Kelner, & Sadok, 2010) es una popular plataforma de IaaS en la nube, desarrollada por la comunidad OpenNebula; integra un hipervisor (Xen, KVM, VMware), una base de datos (SQLite, MySQL), un sistema de archivos distribuidos (GlusterFS, NFS), almacenamiento de imágenes (ceph, LVM,vmfs), redes virtuales (Open vSwitch) y servidores externos (DHCP, LDAP), para construir y gestionar plataformas flexibles en la nube. OpenNebula fue originalmente lanzado en 2008 y su versión estable, la 5.4.0, en julio de 2017. La arquitectura de OpenNebula presenta tres capas: controlador, núcleo y herramientas. El controlador en el nivel bajo es responsable de interactuar con los sistemas operativos para ubicar recursos para las máquinas virtuales, redes y almacenamiento. También realiza gestiones AAA (Accounting, Authorization, and Authentication) para los usuarios. Provee una interfaz de línea de comandos [CLI, Command Line Interface], APIs y una GUI basada en web para la interacción con OpenNebula para el manejo de las VM.

Por defecto, OpenNebula presenta un usuario y contraseñas internas basadas en sistemas de autorización y autenticación, donde las credenciales ingresadas por el usuario se verifican contra las que están presentes en la base de datos OpenNebula. Para gestionar privilegios de acceso, los usuarios son clasificados en cuatro tipos: administrador dedicado (administrador de grupo), usuario regular (acceso más funcional), usuario público (funcionalidad básica e interfaces públicas) y usuario de servicio (cuenta de usuario de servicio). En general, el dueño de un recurso tiene el privilegio de acceder y gestionar por sí

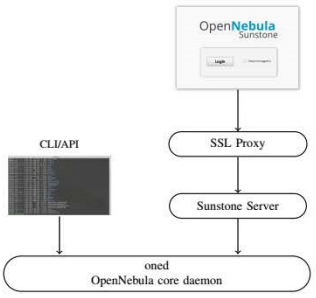

Figure 6. Overview of OpenNebula
Visión general de OpenNebula (2018)

as the Certificate Authority [CA]. It issues a certificate and signs the user token by using its secret key. Each API endpoint, with Keystone's certificate, revocation list, and signature key, can independently validate the PKI token sent by a user.

Khan, Ylitalo, and Ahmed (2011) integrate OpenStack and OpenId to provide "Authentication as a service" and single-sign-on service to its users; in the proposed algorithm, authentication APIs communicate with the user through the GUI and in the back, authentication procedure happens through a Policy Decision Point [PDP].

OpenNebula

OpenNebula (2018; Ristov & Gusev, 2013; Endo, Goncalves, Kelner, & Sadok, 2010) is a popular open source IaaS cloud platform developed by the OpenNebula Community. It integrates hypervisor (Xen, KVM, VMware), database (SQLite, MySQL), distributed file system (GlusterFS, NFS), image datastore (ceph, LVM, vmfs), virtual networks (Open vSwitch) and external servers (DHCP, LDAP) to build and manage flexible cloud platforms. OpenNebula was originally released in 2008. The current stable version is v5.4.0, released on July 20, 2017. The architecture of OpenNebula has three layers: driver, core and tools. The driver at the bottom, it is responsible for interacting with the underlying operating system to allocate resources for spawning Virtual Machines [VMs], handling storage space and networking; in the middle, the core is the centralized management layer that controls and monitors resources for VMs, virtual networks and storage, and also performs Accounting, Authorization and Authentication [AAA] for the users; the tool layer provides the Command Line Interface [CLI], libvirt APIs and web-based GUI for interaction with OpenNebula and for handling the VMs.

By default, OpenNebula has an internal name/password based authentication and authorization system, in which the credentials entered by the user is verified against those already present in the OpenNebulas database; for managing access privileges, users are classified into four types: namely administrator (admin group), regular user (access most functionality), public user (basic functionality and public interfaces) and service user (service user account). In general, an owner of a resource has the privilege of accessing and managing the same and is able to share its resources by use and/or manage privileges with other users. OpenNebula also supports external authentication, via X.509 or LDAP, for its users. Figure 6 corresponding to an architectural overview of OpenNebula authentication. As shown in the Figure 6, authentication can be performed through CLI, API or Sunstone GUI. OpenNebula's Sunstone server and corresponding web-based GUI simplify the task of in-

teraction and management of the cloud infrastructure, for both, administrator and user. OpenNebula management operations are performed through the Sunstone server. Following are the authentication mechanisms supported in OpenNebula v4.14.

- Built-in name/password and token authentication: OpenNebula commands oneuser create and oneuser delete are used by the administrator (oneadmin) for adding and deleting user accounts. The user information is stored in a backend database for future user authentication. OpenNebula maintains an internal variable, AUTH DRIVER, to select authentication driver for the user. The default driver, core, is used for handling the name/password or token based internal authentication. Users enter their name and a valid password through the Sunstone portal or CLI/API, and it is validated by consulting the backend database. An authentication token, valid for a certain period of time, can also be used for authentication.
- SSH Authentication: In OpenNebula v4.12, the SSH authentication mechanism is only supported through the CLI interface. It uses the standard SSH RSA key pair to perform authentication. The user can generate the key pair and send the public key to the administrator for creating a new user account in OpenNebula. The login phase requires an OpenNebula username and a token encrypted with user's private key. The user can generate the login token by using oneuser login command, with a default expiration time of 10 hours.
- X.509 Authentication: OpenNebula also supports X.509 certificates, shown in Figure 3, for user authentication and for creating, updating user accounts. An OpenNebula administrator can create the user account using X.509 certificate, with password same the subject's distinguished name (DN) present in the certificate. For authentication, the X.509 certificates can either be used in two ways. Firstly, the certificates can be passed through CLI. In this mode, the user executes the login command with username, X.509 certificate and private key as parameters. The command uses the private key of the user to encrypt the text document and generates a login token with a default expiration time of 10 hours. Subsequently, the user sends the token to OpenNebula for authentication. The server uses the public key of the user to retrieve the X.509 certificate from the token and uses it to authenticate the user. Secondly, the X.509 certificates can also be used through Sunstone UI. For this configuration, the administrator needs to deploy an SSL

mismo y es capaz de compartir sus recursos a usar, y gestionar privilegios para otros usuarios. OpenNebula también soporta la autenticación externa vía X.509 o LDAP para sus usuarios. La Figura 6 corresponde a una visión de la arquitectura de la autenticación de OpenNebula. Como se puede ver en ella, la autenticación se puede llevar a cabo a través de CLI, API o GUI Sunstone. El servidor Sunstone de OpenNebula, y su correspondiente interfaz basada en web, simplifican las tareas de interacción y gestión de la infraestructura en la nube, tanto para el administrador, como para el usuario. Las operaciones de gestión de OpenNebula se ejecutan a través del servidor Sunstone. Los siguientes son los mecanismos de autenticacion soportados por OpenNebula v4.14.

- Usuario y contraseña incorporados y autenticación por token: el administrador (oneadmin) utiliza los comandos oneuser create y oneuser delete para añadir y borrar cuentas de usuario. La información de usuario se almacena en una base de datos back-end para futuras autenticaciones del usuario. OpenNebula mantiene una variable interna AUTH DRIVER para seleccionar el controlador de autenticación para el usuario; el controlador por defecto —el núcleo— se emplea para manejar el nombre/contraseña o token basado en autenticación interna. Los usuarios ingresan su nombre y una contraseña válida a través del portal Sunstone o de CLI/API; esta información es validada al consultar la base de datos de back-end. Un token de autenticación, válido por cierto periodo de tiempo, también se puede utilizar para autenticación.
- Autenticación SSH: en OpenNebula v4.12, el mecanismo de autenticación SSH es soportado únicamente a través de la interfaz de línea de comandos; utiliza un par de llaves estándar SSH RSA para que el administrador cree una nueva cuenta de usuario en OpenNebula. La fase de autenticación requiere un usuario de OpenNebula y un token encriptado con la llave privada. El usuario puede generar el token con un tiempo de expiración por defecto de diez horas utilizando el comando oneuser login.
- Autenticación X.509: OpenNebula también soporta certificados X.509 —mostrados en la Figura 3— para la autenticación de usuarios y para crear o actualizar cuentas de usuario. Un administrador de OpenNebula puede crear una cuenta de usuario utilizando un certificado X.509 con contraseña igual al DN presente en el certificado. Para autenticación, los certificados X.509 puede ser utilizado de dos maneras: primera, los certificados se transmiten a través de la CLI, en este modo, el usuario ejecuta el comando de ingreso con su usuario, el certificado X.509 y la llave privada como parámetros; el comando utiliza la llave privada del usuario para encriptar el documento de texto y genera un token de ingreso con el tiempo de expiración por defecto de diez horas. Consecuentemente, el usuario envía el token a OpenNebula para autenticación; el servidor utiliza la llave

pública del usuario para recuperar el certificado X.509 desde el token y lo utiliza para autenticar al usuario; después, los certificados X.509 se pueden utilizar a través de la interfaz Sunstone, para lo cual el administrador necesita desplegar un SSL capaz de procesar peticiones HTTP de proxies encima del servidor Sunstone —como lo muestra la Figura 6—. El servidor maneja la validez del certificado, encripta las credenciales utilizando el certificado del servidor y envía el token a OpenNebula.

- Autenticación LDAP: OpenNebula se puede configurar para realizar una autenticación externa a través de LDAP. Para ello, se usa un add-on de autenticación LDAP para conectar con un servidor LDAP y autenticar al usuario mediante una operación de bind, como se explicó El add-on no puede crear o modificar la información de usuario en el servidor LDAP, pero si realizar operaciones de bind y búsqueda para usuarios del servidor.
- Autenticación de servidores: este método soporta el uso de servidores externos para gestionar la autenticación en OpenNebula, para lo que se basa en servidores Sunstone y EC2. Los servidores autentican el usuario antes de reenviar su petición al demonio (daemon) de OpenNebula. La encriptación de llave simétrica o certificados X.509 pueden utilizarse para asegurar la comunicación entre el servidor de autenticación y OpenNebula.

Eucalyptus

Eucalyptus (2018) es otra plataforma open source popular para construir nubes privadas y públicas compatibles con Amazon Web Services [AWS]. Fue liberado bajo la licencia GPLv3 y desarrollado por Eucalyptus Systems Inc. La versión 4.4.2 es estable y fue lanzada en agosto de 2017. Ella presenta cinco componentes principales: controlador de nodo [NC, Node Controller], controlador de almacenamiento morsa [WS3, Walrus Storage controller], controlador de clúster [CC, Cluster Controller], controlador de nube [CLC, CLoud Controller] y controlador de almacenamiento [SC, Storage Controller]. El NC se ejecuta en cada nodo albergando instancias de máquinas virtuales y controla el ciclo de vida de las instancias en el nodo al interactuar con el OS, con el hipervisor y con el CC simultáneamente. El CC es la parte delantera del clúster y es responsable de concertar la ejecución de las VM en un particular NC, reúne información de los NC y envía reportes al CLC. Este último provee la interfaz web para que los usuarios interactúen con Eucalyptus. Adicionalmente, el CLC es responsable de programar los recursos y ejecutar la contabilidad en el sistema. El WS3 ofrece almacenamiento persistente para las imágenes de las VM, además de instantáneas de volumen para los datos de usuario. Finalmente, el SC provee un almacenamiento en bloques para instancias en su clúster.

El sistema de gestión de acceso e identidad [IAM, Identity and Access Management] se utiliza para crear y gestionar cuentas de usuario, asignar roles, configurar cuotas y definir políticas de acceso. Eucalyptus gestiona el control de acceso

capable http proxy on top of the Sunstone server, as shown in Figure 6. It handles certificate validation, encrypts the credentials using server certificate and sends the token to OpenNebula.

- LDAP Authentication: OpenNebula can be configured to perform external authentication through LDAP. For this, an LDAP authentication add-on is used to connect to an existing LDAP server and authenticate the user using Bind operation, as explained earlier. The addon cannot create or modify user information in the LDAP server but can perform Bind and search operations for users on the server.
- Servers Authentication: This method supports the use of external servers for handling authentication in OpenNebula. OpenNebula ships with Sunstone and EC2 servers for this purpose. The servers authenticate the user before forwarding their request to the OpenNebula daemon. Symmetric-key encryption or x509 certificates can be used to secure the communication between the authentication server and OpenNebula.

Eucalyptus

Eucalyptus (2018) is another popular open-source software platform for building Amazon Web Services [AWS]-compatible private and hybrid clouds. It is released under the GPLv3 license and developed by the Eucalyptus Systems Inc. Its current stable version is v4.4.2, released on August 30, 2017. Its five main components are: Node Controller [NC], Walrus Storage controller [WS3], Cluster Controller [CC], Cloud Controller [CLC] and Storage Controller [SC]. The NC is executed on every node hosting VM instances and controls the life cycle of instances on the node by interacting with the OS, hypervisor and CC simultaneously. The CC is the frontend of a cluster, it is responsible for scheduling the execution of VMs on particular NC, collects information of NCs and sends a report to CLC. The CLC provides the web interface for the users to interact with Eucalyptus. In addition, CLC is also responsible for scheduling resources and performs system accounting. WS3 offers persistent storage for VM images, volume snapshots and user data. Finally, the SC provides the block storage for instances within its cluster.

The Identity and Access Management [IAM] system is used for creating and managing user accounts, assigning roles, setting quotas and defining access policies. Eucalyptus manages access control by associating each user with a single account where the users are associated with Groups with certain access privileges. Accounts are identified by its Unique ID or a unique name. Eucalyptus uses two special identities, namely eucalyptus account and admin account,

for the administration purposes. An administrator can create or delete accounts, groups and users, and assign roles through CLI by using commands, like euare-accountcreate, euare-groupcreate, euare-rolecreate and euare-usercreate, available in Euca2ools. Eucalyptus supports different types of user credential for authentication, like X.509 certificate and secret access key. Users can enter their credential through the web interface or CLI, what is verified against the information stored in the local CLC database.

Eucalyptus also supports the integration of LDAP or Active Directory (AD) to import user and group information from these servers, however, Eucalyptus maintains the user-specific information, such as secret access keys, X.509 certificates and policies, in its internal database. The mapping of the user/groups information from an LDAP/AD server to the identity structure of Eucalyptus can be either performed manually or based on object classes in LDAP (RFC 4512). The internal database of Eucalyptus can be enabled for automatic synchronization with the LDAP/AD server or the synchronization can be performed manually by uploading the LDAP/AD Integration Configuration [LIC] file.

IV. Security Vulnerabilities of Token - Based Authentication

Token-based authentication is popularly used in lots of applications. Many service providers like Facebook, Twitter, Google+ and GitHub use tokens for user authentication. Although the system has advantages as it eases up the process of authentication, it has the following limitations:

- Firstly, and the most important one, the authentication information of every users is stored in a centralized database (Aull, Kerr, Freeman, & Bellmore, 2003). This database can be a bottleneck in various use-cases. Denial-of-service attack, hacking or any other catastrophic event can bring the whole system to a halt.
- The placement of user data is another important issue, the application developer uses third-party token based authentication tools for this purposes. The sensitive user credentials are stored on the third-party server, which may use the data for their own benefit. For example, one of the largest third-party authentication provider OAuth was found guilty of selling user data (Ding, Dey, Kaafar, & Ross, 2014). Moreover, users are unaware of the information residing inside the token.
- Putting many eggs in one basket (Wright, 1996) is a vital issue for token-based authentication. In general, a single token is used for a number of services on the

al asociar cada usuario con una cuenta sencilla, en la que dichos usuarios son asignados a grupos con ciertos privilegios de acceso. Las cuentas se identifican por su ID o nombre único. Eucalyptus utiliza dos identidades especiales para propósitos de administración, denominadas cuenta eucaliptus y cuenta de administrador. El administrador puede crear o borrar cuentas, grupos o usuarios, además de asignar roles a través de la CLI, utilizando comandos como euare-accountcreate, euare-groupcreate, euare-rolecreate y euare-usercreate, disponibles en Euca2ools. Eucalyptus soporta diferentes tipos de credenciales de usuario para autenticación, como certificados X.509 y llaves de acceso secretas. Los usuarios pueden ingresar sus credenciales a través de la interfaz web o a través de línea de comandos, donde son verificadas con la información almacenada en la base de datos CLC local.

Eucalyptus soporta la integración de LDAP o directorio activo [AD, Active Directory] para importar información del usuario o grupo para estos servidores, pero mantiene la información específica de usuario —como llaves de acceso secretas, certificados X.509 y políticas— en su base de datos interna. El mapeo de la información de usuarios/grupos desde un servidor LDAP/AD, para la estructura de identidad de Eucalyptus, puede ser llevado a cabo manualmente o estar basado en clases de objetos en LDAP (RFC 4512). La base de datos interna de Eucalyptus se puede activar para sincronización automática con el servidor LDAP/AD o la sincronización se puede llevar a cabo manualmente, al subir el archivo de configuración de integración de LDAP/AD.

IV. Vulnerabilidades de seguridad de la autenticación basada en tokens

La autenticación basada en tokens es popular en diversas aplicaciones. Distintos proveedores de servicio como Facebook, Twitter, Google+ y GitHub la utilizan con sus usuarios. Aunque este sistema tiene ventajas y el proceso de autenticación es relativamente fácil, tiene las siguientes limitaciones:

- La primero y la más importante, la información de autenticación de cada usuario se almacena en una base de datos centralizada (Aull, Kerr, Freeman, & Bellmore, 2003), la que puede convertirse en un cuello de botella en varios casos. Ataques de denegación del servicio, hackeos o cualquier otro evento catastrófico pueden afectar totalmente al sistema.
- La ubicación de la información del usuario es otro problema importante, el desarrollador de la aplicación utiliza herramientas de autenticación basadas en tokens y en proveedores externos para este propósito. Las credenciales de usuario (información sensible) se almacenan en servidores externos, lo que puede conllevar al uso no autorizado de dicha información. Por ejemplo, uno de los más grandes proveedores de autenticación OAuth externos fue hallado culpable de vender la información que guardaba (Ding, Dey, Kaafar, & Ross, 2014). Además, los usuarios no son conscientes de la información presente dentro del token.

- El "colocar todos los huevos en una sola canasta" (Wright, 1996) es un dilema vital para la autenticación basada en tokens. En general, un solo token se utiliza para un cierto número de servicios en la red, por lo que, si alguien no autorizado obtiene acceso a uno de los servicios utilizando un token falso, podría tener acceso a todos los servicios, sin restricciones.
- Los mecanismos externos están sujetos a ataques de phishing (Garera, Provos, Chew, & Rubin, 2007). Aunque estos problemas se abordan activamente en diferentes plataformas, una solución completa no está disponible aún. Cuando se presenta un problema, la entidad generalmente lo soluciona emitiendo actualizaciones. Un incidente ocurrió con OAuth, cuando un script del lado de usuario pudo acceder a los sistemas sin utilizar contraseña alguna.

V. Discusión y conclusiones

En este artículo se han investigado las técnicas de autenticación basadas en tokens empleadas por plataformas en la nube open source como Cloudstack, OpenStack, Eucalyptus y OpenNebula. Se exploraron las propiedades de los tokens y se han discutido las herramientas de autenticación externas que son comúnmente utilizadas en plataformas SC como: SAML, OpenID 2.0 y OAuth 2.0. También se ha estudiado extensivamente el número de handshakes requeridos para obtener tokens y para acceder a servicios desde los proveedores, además de discutir las vulnerabilidades de seguridad de la autenticación basada en tokens. Remarcamos brevemente a continuación algunas características importantes, observaciones y limitaciones de esta técnica de autenticación.

La autenticación basada en tokens reduce el problema de recordar múltiples nombres de usuario y contraseñas para acceder a distintos servicios desde un CSP, como resultado de ellos, se incrementa la seguridad de las contraseñas y se acelera el proceso de inicio de sesión (Cirani, Picone, Gonizzi, Veltri, & Ferrari, 2015); sin embargo, existen algunas vulnerabilidades de seguridad en estos mecanismos, como que son propensos a ataques de phishing (Hammer-Lahav, 2010). Adicionalmente, la privacidad de los usuarios puede ser utilizada incorrectamente en la autenticación basada en tokens. Por otra parte, carece de anonimidad para el usuario, puesto que la información puede ser compartida por el proveedor de servicios (Tan, Hsu, & Pinn, 2001).

La técnica de autenticación basada en token más popular es OAuth, la utilizan Google y Facebook para proveer sus servicios al usuario. Provee una API basada en un framework de autorización de usuario con medidas de seguridad bien definidas y puede ser utilizada para soportar servicios web duraderos al usuario; sin embargo, como el framework de OAuth 2.0 es altamente extensible, la interoperabilidad de los componentes puede ser un reto (Hardt, 2012).

OpenID provee una plataforma para los usuarios en la que pueden elegir un proveedor de servicio externo para propósitos de autenticación y utiliza un protocolo de descubrimiento para

network, if somehow, an unauthorized person gets access to one service using a forged token, then it can gain access to all the services that entitled to use.

- Furthermore, the third-party mechanisms are subjected to phishing attacks (Garera, Provos, Chew, & Rubin, 2007). These concerns are being actively addressed at different platforms, but a complete solution is not yet available. Whenever an issue arises, the concerned party resolves that issue by issuing updates. One such incident happened with OAuth when a user side script could signing on the system without using any password.

V. Discussion and Conclusions

We have surveyed the token-based authentication techniques used by the popular open source cloud platforms like: Cloudstack, Open-Stack, Eucalyptus, and OpenNebula. We explore the properties of a token and discuss the third party authentication tools that are commonly used in the OSC platform like SAML, OpenID 2.0 and OAuth 2.0. Also, we extensively studied the number of handshakes required for obtaining tokens and for accessing services form the service providers, and we discuss the security vulnerabilities of token-based authentication. Now, we briefly remark important features, observations and limitations of token-based authentication technique.

Token-based authentication reduces the problem of remembering multiple username and password for accessing different services from a CSP. As a result, it increases the password security and accelerates the signup process (Cirani, Picone, Gonizzi, Veltri, & Ferrari, 2015). However, there are some security vulnerabilities in token-based authentication, such as it is prone to phishing attacks (Hammer-Lahav, 2010). In addition, the user privacy can be misused in token-based authentication. Another disadvantage is that it lacks the anonymity for the user, as the user's information can be miss-handled or shared by the service provider (Tan, Hsu, & Pinn, 2001).

The most popular token based authentication technique is OAuth. It is used by both, Google and Facebook, for providing their services to the user. It provides API based user authorization framework with well-defined security measures and can be used for supporting long lasting web-based services to the user. However, as the framework of OAuth 2.0 is highly extensible, interoperability of the components can be challenge (Hardt, 2012).

OpenID provides a platform to the users where a user can choose a third-party service provider for its authentication purposes. OpenID uses a discovery protocol to find the provider for a corresponding openID. This is a main difference between OpenID and SAML, with SAML, users

are coupled with the SAML Identity Providers, which is not true for OpenID. Nowadays, OpenID based authentication is mostly used in designing the mobile applications.

Kerberos, on the other hand, is designed by MIT for secure mutual authentication between a user and a service provider. It can resist attacks, like phishing, intrusion and replay; however, the specification requires all components (i.e Kerberos server, user and servers) to co-exist in the same network for accessing services from Kerberos, which can be a limitation for accessing to cloud services.

encontrar el proveedor para una correspondiente openID. Esta es la principal diferencia entre OpenID y SAML; con el último, los usuarios se acoplan con los proveedores de identidad SAML, lo cual no ocurre con OpenID. Actualmente, la autenticación basada en OpenID es más usada en el diseño de aplicaciones móviles.

Por su parte, Kerberos, que fue diseñado por el MIT para autenticación mutua segura entre un usuario y proveedor de servicio, puede resistir ataques como phishing, intrusión y repetición; sin embargo, la especificación requiere que todos los componentes (servidor Kerberos, usuario y servidores) coexistan en la misma red para acceder a los servicios desde Kerberos, lo que puede representar una limitación para acceder a servicios en la nube.

References / Referencias

Alizadeh, M. & Hassan, W. (2013). Challenges and opportunities of mobile cloud computing. In: *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, (pp. 660-666). IEEE.

Al-Janabi, S. T. F. and s. Rasheed, M. A. (2011). Public-key cryptography enabled Kerberos authentication. In: *Developments in E-systems Engineering (DeSE), 2011*, (pp. 209-214). IEEE.

Arifeen, F. U., Siddiqui, R. A., Ashraf, S., & Waheed, S. (2015). Inter-cloud authentication through x.509 for defense organization. In: *Applied Sciences and Technology (IBCAST), 2015 12th International Bhurban Conference on*, (pp. 299-306). IEEE.

Aull, K., Kerr, T., Freeman, W. & Bellmore, M. (2003). *US Patent App. 10/027,607* [Public key infrastructure token issuance and binding]. Washington, DC: US Patent and Trademark office.

Banerjee, A., Hasan, M., Rahman, M. A. & Chapagain, R. (2017). Cloak: A stream cipher based encryption protocol for mobile cloud computing. *IEEE Access, 5*, 17678-17691.

Bauer, C. (2012). X.509 identity certificates with local verification. In: *Communications (ICC), 2012 IEEE International Conference on*, (pp. 6727-6732). IEEE.

Blum, M. & Micali, S. (1984). How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing, 13*(4), 850-864.

Cantor, S., Kemp, I. J., Philpott, N. R. & Maler, E. (2005, March). *Assertions and protocols for the oasis security assertion markup language* [OASIS standard]. Retrieved from: https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

Chaabane, A., Ding, Y., Dey, R., Kaafar, M. A., & Ross, K. W. (2014). A closer look at third-party OSN applications: Are they leaking your personal information? In: *International Conference on Passive and Active Network Measurement*, (pp. 235-246). Basel, Swizerland: Springer.

Chen, X., Liu, J., Han, J. & Xu, H. (2010). Primary exploration of mobile learning mode under a cloud computing environment. In: *E-Health Net- working, Digital Ecosystems and Technologies (EDT), 2010 International Conference on*, (Vol. 2, pp. 484-487. IEEE.

Chiang, J., Yen, E.-W., & Chen, Y.-H. (2013). Authentication, authorization and file synchronization in hybrid cloud: On case of Google Docs, Hadoop and Linux local hosts. In: *Biometrics and Security Technologies (IS- BAST), 2013 International Symposium on*, (pp. 116-123). IEEE.

Cirani, S., Picone, M., Gonizzi, P., Veltri, L. & Ferrari, G. (2015). IoT-OAS: An OAuth-based authorization service architecture for secure services in IoT scenarios. *IEEE Sensors Journal, 15*(2), 1224-1234.

Apache CloudStack: *Open source cloud computing* [Website]. (2017). Retrieved from: http://cloudstack.apache.org/.

Doukas, C., Pliakas, T. & Maglogiannis, I. (2010). Mobile healthcare information management utilizing cloud computing and Android OS. In: *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, (pp. 1037-1040). IEEE.

Endo, P. T., Goncalves, G. E., Kelner, J. & Sadok, D. (2010). A survey on open-source cloud computing solutions. In: *Brazilian Symposium on Computer Networks and Distributed Systems*. SBC-LARC.

Eucalyptus cloud platform. (2018). Retrieved from: https://github.com/eucalyptus/eucalyptus

Garera, S., Provos, N., Chew, M., & Rubin, A. D. (2007). A framework for detection and measurement of phishing attacks. In: *Proceedings of the 2007 ACM Workshop on Recurring Malcode*, (pp. 1-8). New York, NY: ACM. doi:10.1145/1314389.1314391

Grzonkowski, S., Corcoran, P., & Coughlin, T. (2011). Security analysis of authentication protocols for next-generation mobile and CE cloud services. In: *Consumer Electronics - Berlin (ICCE-Berlin), 2011 IEEE International Conference on*, (pp. 83-87). IEEE.

Guo, M.-H., Liaw, H.-T., Hsiao, L.-L., Huang, C.-Y., & Yen, C.-T. (2012). Authentication using graphical password in cloud. In: *Wireless Personal Multimedia Communications (WPMC), 2012, 15th International Symposium on*, (pp. 177-181). IEEE.

Hammer-Lahav, E. (2010). *The OAuth 1.0 protocol* [IETF technical report]. Retrieved from: https://tools.ietf.org/html/rfc5849

Hani, Q. B. & Dichter, J. P. (2016). Secure and strong mobile cloud authentication. In: *2016 SAI Computing Conference*, (pp. 562–565). IEEE.

Hardt, D. (2012). *The OAuth 2.0 authorization framework* [IETF proposed standard]. Retrieved from: https://tools.ietf.org/html/rfc6749?

Hoang, D. & Chen, L. (2010). Mobile cloud for assistive healthcare (mocash). In: *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*, (pp. 325-332). IEEE.

Ismaeel, S., Miri, A., Chourishi, D. & Dibaj, S. M. R. (2015). Open source cloud management platforms: A review. In: *Cyber Security and Cloud Computing (CSCloud), 2015 IEEE 2nd International Conference on*, (pp. 470-475). IEEE.

Joshi, R. & Lau, W. (2018). *US Patent 9,973,491* [Determining an identity of a third-party user in an SAML implementation of a web-service]. Washington, DC: US Patent and Trademark office.

Kang, L. & Zhang, X. (2010). Identity-based authentication in cloud storage sharing. In: *Multimedia Information Networking and Security (MINES),2010 International Conference on*, (pp. 851-855. IEEE.

Khan, R., Ylitalo, J. & Ahmed, A. (2011). OpenID authentication as a service in OpenStack. In: *Information Assurance and Security (IAS), 2011, 7th International Conference on*, (pp. 372-377). IEEE.

Khandelwal, N. S. & Kamboj, P. (2015). Two factor authentication using visual cryptography and digital envelope in *Kerberos. In: Electrical, Electronics, Signals, Communication and Optimization (EESCO), 2015 International Conference on*. IEEE. doi:10.1109/EESCO.2015.7253638

Kim, H. & Timm, S. C. (2014). X.509 authentication and authorization in Fermi Cloud. In: *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, (pp. 732-737). IEEE.

Le, H. Q., Truong, H. P., Van, H. T. & Le, T. H. (2015). A new pre-authentication protocol in Kerberos 5: Biometric authentication. In: *Computing Communication Technologies - Research, Innovation, and Vision for the Future (RIVF), 2015 IEEE RIVF International Conference on*, (pp. 157-162). IEEE.

Lecuyer, P. (1999). Tables of linear congruential generators of different sizes and good lattice structure. *Mathematics of Computation of the American Mathematical Society, 68*(225), 249-260.

Li, J. (2010). Study on the development of mobile learning promoted by cloud computing. In: *Information Engineering and Computer Science (ICIECS), 2010 2nd International Conference on*, (pp. 1-4). IEEE. doi: 10.1109/ICIECS.2010.5678245

Mainka, C., Mladenov, V., Schwenk, J. & Wich, T. (2017). SoK: Single sign-on security: an evaluation of openID connect. In: *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on*, (pp. 251-266. IEEE.

Neuman, C., Yu, T., Hartman, S. & Raeburn, K. (2005). *RFC 4120: The Kerberos network authentication service (v5)* [IETF proposed standard]. Retrieved from: https://tools.ietf.org/html/rfc4120.html

von Neumann, J. (1951). Various techniques used in connection with random digits "Monte Carlo Method". In: A. S. Householder; G. E. Forsythe; and H. H. Germond (Eds.). *National Bureau of Standards Applied Mathematics Series, 12*, (pp. 36-38). Washington, D.C.: U.S. Government Printing Office.

Nkosi, M. & Mekuria, F. (2010). Cloud computing for enhanced mobile health applications. In: *Cloud Computing Technology and Science (Cloud-Com), 2010 IEEE Second International Conference on*, (pp. 629-633). IEEE.

OpenNebula.org [Website]. (2018). Retrieved from: http://opennebula.org/

OpenStack [Website]. (2018). Retrieved from: http://www.openstack.org/

Hong-qing, G. & Yan-jie, Z. (2010). System design of cloud computing based on mobile learning. In: *Knowledge Acquisition and Modeling (KAM), 2010 3rd International Symposium on*, (pp. 239-242). IEEE. doi:10.1109/KAM.2010.5646248

Recordon, D. & Reed, D. (2006). Openid 2.0: A platform for user-centric identity management. In: *Proceedings of the Second ACM workshop on Digital Identity Management*, (pp. 11-16). New York, NY: ACM.

Richer, J. & Sanso, A. (2017). *OAuth 2 in Action*. Shelter Island, NY: Manning Publications.

Ristov, S. & Gusev, M. (2013). Security evaluation of open source clouds. In: *EUROCON, 2013* (pp. 73-80). IEEE.

Ruj, S., Stojmenovic, M., & Nayak, A. (2014). Decentralized access control with anonymous authentication of data stored in clouds. *Parallel and Distributed Systems, IEEE Transactions on, 25*(2), 384-394.

Sabharwal, N. & Shankar, R. (2013). *Apache CloudStack cloud computing*. Packt Publishing.

Sarvabhatla, M. & Vorugunti, C. (2015). A robust mutual authentication scheme for data security in cloud architecture. In: *Communication Systems and Networks (COMSNETS), 2015 7th International Conference on*, (pp. 1-6). IEEE.

Tan, W., Hsu, J., & Pinn, F. (2001). *US Patent App. 09/792,785* [Method and system for token-based authentication]. Washington, DC: U.S. Patent and Trademark Office.

Tang, W.-T., Hu, C.-M. & Hsu, C.-Y. (2010). A mobile phone based home-care management system on the cloud. In: *Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on*, (Vol. 6, pp. 2442-2445). IEEE.

Thota, C., Sundarasekar, R., Manogaran, G., Varatharajan, R. & Priyan, M. (2018). Centralized fog computing security platform for IoT and cloud in healthcare system. In: *Exploring the convergence of big data and the Internet of things*, (pp. 141-154). IGI Global.

Wichmann, B. A. & Hill, I. D. (1982). Algorithm as 183: An efficient and portable pseudo-random number generator. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 31*(2): 188-190.

Wright, B. (1996). Eggs in baskets: Distributing the risks of electronic signatures, J. Marshall J. *Computer & Info. 15*(2), 189. Retrieved from: https://repository.jmls.edu/jitpl/vol15/iss2/1/

Xiao, Z. & Xiao, Y. (2013). Security and privacy in cloud computing. *IEEE Communications Surveys & Tutorials, 15*(2): 843-859.

Yang, X., Pan, T. & Shen, J. (2010). On 3g mobile e-commerce platform based on cloud computing. In: *Ubimedia Computing (U-Media), 2010 3rd IEEE International Conference on*, (pp. 198-201). IEEE.

Yesudas, M., Gupta, S., & Ramamurthy, H. (2014). Cloud-based mobile commerce for grocery purchasing in developing countries. *IBM Journal of Research and Development, 58*(5/6): 16:1-16:7.

Zwattendorfer, B. & Tauber, A. (2012). Secure cloud authentication using eIDs. In: *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*, (Vol.1, pp. 397-401). IEEE.

.

CURRICULUM VITAE

Amit Benerjee *Associated with South Asian University (New Delhi, India) as an assistant professor in the Department of Computer Science, since 2011. He received Ph.D. degree from the Department of Computer Science at National Tsing Hua University (Taiwan) in 2009. He served as an engineer in the Industrial Technology Research Institute (ITRI) in Taiwan between 2009 to 2011. His research interest include: cloud computing, IoT, network security, mobile ad-hoc and sensor networks. He is a member of IEEE / Profesor asistente del departamento de Ciencias de la Computación en la South Asian University (New Delhi, India) desde 2011. Recibió su doctorado del departamento de Ciencias de la Computación de la National Tsing Hua University (Taiwan) en 2009. Se desempeñó como ingeniero en el Industrial Technology Research Institute [ITRI] (Taiwan) entre 2009 y 2011. Sus áreas de interés incluyen: computación en la nube, Internet de las Cosas, seguridad en redes, mobile ad-hoc y redes de sensores. Es miembro de IEEE.*

Mahamudul Hasan *Received his master's degree in computer science from the Department of Computer Science of the South Asian University [SAU] (New Delhi, India) in 2013. Currently, he is a PhD student in the same department in SAU. Earlier, he received B.Sc. (Engineering) in Computer Science and Telecommunication Engineering from Noakhali Science and Technology University [NSTU] (Bangladesh). His research interest includes mobile computing, cloud computing, IoT, network security. He is also serving as a Lecturer in the Department of Computer Science and Telecommunication Engineering of NSTU. He received prestigious ICT and Bangabandhu fellowship from the Bangladesh Government for PhD and master's studies, respectively / Recibió su título como Máster en Ciencias de la Computación del departamento de Ciencias de la Computación de la South Asian University [SAU] en New Delhi (India) en 2013, entidad donde actualmente cursa sus estudios de doctorado. Es Ingeniero en Ciencias de la Computación y Telecomunicaciones de la Noakhali Science and Technology University [NSTU] de Bangladesh. Sus áreas de interés en investigación incluyen computación móvil, computación en la nube, Internet de las Cosas [IoT] y seguridad en redes. Ha sido lector del departamento de ciencias de comunicación y telecomunicaciones de la NSTU. Para sus estudios de doctorado y maestría, ganó las prestigiosas becas ICT and Bangabandhu, respectivamente, otorgadas por el Gobierno de Bangladesh.*