Chapter 5

Federation, Presence, Identity, and Privacy in the Cloud

5.1 Chapter Overview

Building a seamless federated communications capability in a cloud environment, one that is capable of supporting people, devices, information feeds, documents, application interfaces, and other entities, is affected by the architecture that is implemented. The solution chosen must be able to find such entities, determine their purpose, and request presence data so that others can interact with them in real time. This process is known as discovery. Providing discovery information about the availability of various entities enables organizations to deploy real-time services and achieve significant revenue opportunities and productivity improvements.

The advent of on-demand cloud services is changing the landscape for identity management because most current identity management solutions are focused on the enterprise and/or create a very restrictive, controlled, and static environment. We are now moving into a new world, where cloud services are offered on demand and they continuously evolve to meet user needs. Previous models are being challenged by such innovations. For example, in terms of trust assumptions, privacy implications, and operational aspects of authentication and authorization, solutions that seemed to work before are now considered old, outdated, and clunky fixes to identity management. The fluid and omnipresent aspects of federation, presence, and identity in the cloud create new opportunities for meeting the challenges that businesses face in managing security and privacy in the cloud.

5.2 Federation in the Cloud

One challenge in creating and managing a globally decentralized cloud computing environment is maintaining consistent connectivity between untrusted components while remaining fault-tolerant. A key opportunity

for the emerging cloud industry will be in defining a federated cloud ecosystem by connecting multiple cloud computing providers using a common standard.

A notable research project being conducted by Microsoft, called the Geneva Framework, focuses on issues involved in cloud federation. Geneva has been described as a claims-based access platform and is said to help simplify access to applications and other systems. The concept allows for multiple providers to interact seamlessly with others, and it enables developers to incorporate various authentication models that will work with any corporate identity system, including Active Directory, LDAPv3-based directories, application-specific databases, and new user-centric identity models such as LiveID, OpenID, and InfoCard systems. It also supports Microsoft's CardSpace and Novell's Digital Me.

The remainder of this section focuses on federation in the cloud through use of the Internet Engineering Task Force (IETF) standard Extensible Messaging and Presence Protocol (XMPP) and interdomain federation using the Jabber Extensible Communications Platform (Jabber XCP),[1] because this protocol is currently used by a wide range of existing services offered by providers as diverse as Google Talk, Live Journal, Earthlink, Facebook, ooVoo, Meebo, Twitter, the U.S. Marines Corps, the Defense Information Systems Agency (DISA), the U.S. Joint Forces Command (USJFCOM), and the National Weather Service. We also look at federation with non-XMPP technologies such as the Session Initiation Protocol (SIP), which is the foundation of popular enterprise messaging systems such as IBM's Lotus Sametime and Microsoft's Live Communications Server (LCS) and Office Communications Server (OCS).

Jabber XCP is a highly scalable, extensible, available, and device-agnostic presence solution built on XMPP and supports multiple protocols such as Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE) and Instant Messaging and Presence Service (IMPS). Jabber XCP is a highly programmable platform, which makes it ideal for adding presence and messaging to existing applications or services and for building next-generation, presence-based solutions.

Over the last few years there has been a controversy brewing in web services architectures. Cloud services are being talked up as a fundamental shift in web architecture that promises to move us from interconnected silos to a

1. Jabberwas acquired by Cisco Systems in November 2008.

collaborative network of services whose sum is greater than its parts. The problem is that the protocols powering current cloud services, SOAP (Simple Object Access Protocol) and a few other assorted HTTP-based protocols, are all one-way information exchanges. Therefore cloud services aren't real-time, won't scale, and often can't clear the firewall. Many believe that those barriers can be overcome by XMPP (also called Jabber) as the protocol that will fuel the Software-as-a-Service (SaaS) models of tomorrow. Google, Apple, AOL, IBM, Livejournal, and Jive have all incorporated this protocol into their cloud-based solutions in the last few years.

Since the beginning of the Internet era, if you wanted to synchronize services between two servers, the most common solution was to have the client "ping" the host at regular intervals, which is known as polling. Polling is how most of us check our email. We ping our email server every few minutes to see if we have new mail. It's also how nearly all web services application programming interfaces (APIs) work.

XMPP's profile has been steadily gaining since its inception as the protocol behind the open source instant messenger (IM) server jabberd in 1998. XMPP's advantages include:

- It is decentralized, meaning anyone may set up an XMPP server.
- It is based on open standards.
- It is mature—multiple implementations of clients and servers exist.
- Robust security is supported via Simple Authentication and Security Layer (SASL) and Transport Layer Security (TLS).
- It is flexible and designed to be extended.

XMPP is a good fit for cloud computing because it allows for easy two-way communication; it eliminates the need for polling; it has rich publish-subscribe (pub-sub) functionality built in; it is XML-based and easily extensible, perfect for both new IM features and custom cloud services; it is efficient and has been proven to scale to millions of concurrent users on a single service (such as Google's GTalk); and it also has a built-in worldwide federation model.

Of course, XMPP is not the only pub-sub enabler getting a lot of interest from web application developers. An Amazon EC2-backed server can run Jetty and Cometd from Dojo. Unlike XMPP, Comet is based on HTTP,

and in conjunction with the Bayeux Protocol, uses JSON to exchange data. Given the current market penetration and extensive use of XMPP and XCP for federation in the cloud and that it is the dominant open protocol in that space, we will focus on its use in our discussion of federation.

The ability to exchange data used for presence, messages, voice, video, files, notifications, etc., with people, devices, and applications gain more power when they can be shared across organizations and with other service providers. Federation differs from peering, which requires a prior agreement between parties before a server-to-server (S2S) link can be established. In the past, peering was more common among traditional telecommunications providers (because of the high cost of transferring voice traffic). In the brave new Internet world, federation has become a *de facto* standard for most email systems because they are federated dynamically through Domain Name System (DNS) settings and server configurations.

5.2.1 Four Levels of Federation

Technically speaking, federation is the ability for two XMPP servers in different domains to exchange XML stanzas. According to the XEP-0238: XMPP Protocol Flows for Inter-Domain Federation, there are at least four basic types of federation[2]:

1. **Permissive federation**. Permissive federation occurs when a server accepts a connection from a peer network server without verifying its identity using DNS lookups or certificate checking. The lack of verification or authentication may lead to domain spoofing (the unauthorized use of a third-party domain name in an email message in order to pretend to be someone else), which opens the door to widespread spam and other abuses. With the release of the open source jabberd 1.2 server in October 2000, which included support for the Server Dialback protocol (fully supported in Jabber XCP), permissive federation met its demise on the XMPP network.
2. **Verified federation.** This type of federation occurs when a server accepts a connection from a peer after the identity of the peer has been verified. It uses information obtained via DNS and by

2. Peter Saint-Andre, "XEP-0238: XMPP Protocol Flows for Inter-Domain Federation," http://xmpp.org/extensions/xep-0238.html, retrieved 1 Mar 2009.

means of domain-specific keys exchanged beforehand. The connection is not encrypted, and the use of identity verification effectively prevents domain spoofing. To make this work, federation requires proper DNS setup, and that is still subject to DNS poisoning attacks. Verified federation has been the default service policy on the open XMPP since the release of the open-source jabberd 1.2 server.

3. **Encrypted federation.** In this mode, a server accepts a connection from a peer if and only if the peer supports Transport Layer Security (TLS) as defined for XMPP in Request for Comments (RFC) 3920. The peer must present a digital certificate. The certificate may be self-signed, but this prevents using mutual authentication. If this is the case, both parties proceed to weakly verify identity using Server Dialback. XEP-0220 defines the Server Dialback protocol,[3] which is used between XMPP servers to provide identity verification. Server Dialback uses the DNS as the basis for verifying identity; the basic approach is that when a receiving server receives a server-to-server connection request from an originating server, it does not accept the request until it has verified a key with an authoritative server for the domain asserted by the originating server. Although Server Dialback does not provide strong authentication or trusted federation, and although it is subject to DNS poisoning attacks, it has effectively prevented most instances of address spoofing on the XMPP network since its release in 2000.[4] This results in an encrypted connection with weak identity verification.

4. **Trusted federation.** Here, a server accepts a connection from a peer only under the stipulation that the peer supports TLS and the peer can present a digital certificate issued by a root certification authority (CA) that is trusted by the authenticating server. The list of trusted root CAs may be determined by one or more factors, such as the operating system, XMPP server software, or local service policy. In trusted federation, the use of digital certificates results not only in a channel encryption but also in strong authentication. The use of trusted domain certificates effectively prevents DNS poisoning attacks but makes federation

3. http://xmpp.org/extensions/xep-0220.html, retrieved 28 Feb 2009.
4. http://xmpp.org/extensions/xep-0220.html, retrieved 28 Feb 2009.

more difficult, since such certificates have traditionally not been easy to obtain.

5.2.2 How Encrypted Federation Differs from Trusted Federation

Verified federation serves as a foundation for encrypted federation, which builds on it concepts by requiring use of TLS for channel encryption. The Secure Sockets Layer (SSL) technology, originally developed for secure communications over HTTP, has evolved into TLS. XMPP uses a TLS profile that enables two entities to upgrade a connection from unencrypted to encrypted. This is different from SSL in that it does not require that a separate port be used to establish secure communications. Since XMPP S2S communication uses two connections (bi-directionally connected), encrypted federation requires each entity to present a digital certificate to the reciprocating party.

Not all certificates are created equal, and trust is in the eye of the beholder. For example, I might not trust your digital certificates if your certificate is "self-signed" (i.e., issued by you rather than a recognized CA), or your certificate is issued by a CA but I don't know or trust the CA. In either case, if Joe's server connects to Ann's server, Ann's server will accept the untrusted certificate from Joe's server solely for the purpose of bootstrapping channel encryption, not for domain verification. This is due to the fact that Ann's server has no way of following the certificate chain back to a trusted root. Therefore both servers complete the TLS negotiation, but Ann's server then require's Joe's server to complete server Dialback.

In the trusted federation scenario, Dialback can be avoided if, after using TLS for channel encryption, the server verifying identity proceeds to use the SASL protocol for authentication based on the credentials presented in the certificates. In this case, the servers dispense with server Dialback, because SASL (in particular the EXTERNAL mechanism) provides strong authentication.

5.2.3 Federated Services and Applications

S2S federation is a good start toward building a real-time communications cloud. Clouds typically consist of all the users, devices, services, and applications connected to the network. In order to fully leverage the capabilities of this cloud structure, a participant needs the ability to find other entities of interest. Such entities might be end users, multiuser chat rooms, real-time

content feeds, user directories, data relays, messaging gateways, etc. Finding these entities is a process called discovery.

XMPP uses service discovery (as defined in XEP-0030) to find the aforementioned entities. The discovery protocol enables any network participant to query another entity regarding its identity, capabilities, and associated entities. When a participant connects to the network, it queries the authoritative server for its particular domain about the entities associated with that authoritative server.

In response to a service discovery query, the authoritative server informs the inquirer about services hosted there and may also detail services that are available but hosted elsewhere. XMPP includes a method for maintaining personal lists of other entities, known as roster technology, which enables end users to keep track of various types of entities. Usually, these lists are comprised of other entities the users are interested in or interact with regularly. Most XMPP deployments include custom directories so that internal users of those services can easily find what they are looking for.

5.2.4 Protecting and Controlling Federated Communication

Some organizations are wary of federation because they fear that real-time communication networks will introduce the same types of problems that are endemic to email networks, such as spam and viruses. While these concerns are not unfounded, they tend to be exaggerated for several reasons:

- Designers of technologies like XMPP learned from past problems with email systems and incorporated these lessons to prevent address spoofing, unlimited binary attachments, inline scripts, and other attack tactics in XMPP.
- The use of point-to-point federation will avoid problem that occur with multihop federation. This includes injection attacks, data loss, and unencrypted intermediate links.
- Using certificates issued by trusted root CAs ensures encrypted connections and strong authentication, both of which are currently feasible with an email network.
- Employing intelligent servers that have the ability to blacklist (explicitly block) and whitelist (explicitly permit) foreign services, either at the host level or the IP address level, is a significant mitigating factor.

5.2.5 The Future of Federation

The implementation of federated communications is a precursor to building a seamless cloud that can interact with people, devices, information feeds, documents, application interfaces, and other entities. The power of a federated, presence-enabled communications infrastructure is that it enables software developers and service providers to build and deploy such applications without asking permission from a large, centralized communications operator. The process of server-to-server federation for the purpose of interdomain communication has played a large role in the success of XMPP, which relies on a small set of simple but powerful mechanisms for domain checking and security to generate verified, encrypted, and trusted connections between any two deployed servers. These mechanisms have provided a stable, secure foundation for growth of the XMPP network and similar real-time technologies.

5.3 Presence in the Cloud

Understanding the power of presence is crucial to unlocking the real potential of the Internet. Presence data enables organizations to deploy innovative real-time services and achieve significant revenue opportunities and productivity improvements. At the most fundamental level, understanding presence is simple**:** It provides true-or-false answers to queries about the network availability of a person, device, or application. Presence is a core component of an entity's *real-time* identity. Presence serves as a catalyst for communication. Its purpose is to signal availability for interaction over a network. It is being used to determine availability for phones, conference rooms, applications, web-based services, routers, firewalls, servers, appliances, buildings, devices, and other applications. The management of presence is being extended to capture even more information about availability, *or even the attributes associated with such availability*, such as a person's current activity, mood, location (e.g., GPS coordinates), or preferred communication method (phone, email, IM, etc.). While these presence extensions are innovative and important, they serve mainly to supplement the basic information about an entity's network connectivity, which remains the core purpose of presence.

Presence is an enabling technology for peer-to-peer interaction. It first emerged as an aspect of communication systems, especially IM systems such as ICQ, which allowed users to see the availability of their friends. The huge role that IM has had in establishing presence is evident with the protocols