Single Sign-On for Unified Communications

Ping Lin
System Manager Architecture
Avaya Inc.
Belleville, Canada

Sunil Menon
Identity Engine Portfolio
Avaya Inc.
Santa Clara, USA

Shailesh Patel
Identity Engine Portfolio
Avaya Inc.
Santa Clara, USA

Lin Lin
Messaging Architecture
Avaya Inc.Belleville,
Canada

Abstract: This paper presents several approaches for solving the Single Sign-On (SSO) problem for multiple applications (based on different protocols) that constitute a Unified Communications (UC) solution. It focuses on SSO for HTTP- and SIP-based applications first for concreteness, and then extends the discussion to how analogous approaches can apply to applications based on other protocols such as IMAP, SMTP and LDAP. Some considerations about the network and application environment that these design approaches need to work into are also covered. The paper concludes with a discussion of the security of SSO in general and some specific security concerns that arise in the UC context.

Keywords: Single Sign-on, SIP, IMAP, SMTP, LDAP, OpenAM, SAML, Digest Authentation, **IF-MAP, Kerberos**, ***SASL, Unified Communications.***

I. INTRODUCTION

Unified Communications (UC) solutions integrate real-time communications applications such as telephony, audio/video conferencing, instant messaging and presence with non-real-time applications such as web browsing, email, voice mail and directories to provide a unified user experience across multiple media and device types. Given the importance of maintaining a common context and providing a level of usability that users expect for "unified" communications, there is a strong requirement for Single Sign-On (SSO) to the multiple applications that comprise these solutions.

SSO technology is most often applied to web applications based on the HyperText Transfer Protocol (HTTP); however applications in UC solutions can also be based on various other protocols in addition to HTTP. For example, internet telephony may be based on the Session Initiation Protocol (SIP), email / voice mail may be received and sent via the Internet Message Access Protocol (IMAP) and Simple Mail Transfer Protocol (SMTP) respectively, and directories may be accessed via the Lightweight Directory Access Protocol (LDAP). What is needed in UC solutions is SSO that works across multiple protocols to enable the end user to sign on just once to all of the applications being used.

In this paper, we examine several design approaches to solve the SSO problem for multiple UC applications. For concreteness, we will take web access and internet telephony as specific examples of applications and focus on SSO for HTTP and SIP first, followed by a discussion on how analogous approaches can apply to other protocols such as IMAP, SMTP and LDAP. We also cover some considerations about the network and application environment that these design approaches need to work into, as well as thoughts about the security of SSO in the context of UC.

II. SSO FOR HTTP-BASED APPLICATIONS

To provide some background information for the rest of this paper, let us take a look at how SSO typically works for web applications, using OpenAM and Secure Assertion Markup Language (SAML) as specific examples of an SSO implementation and an SSO standard respectively. Other alternative implementations such as Shibboleth, JOSSO, etc., and standards such as OpenID and oAuth, differ in design but espouse similar principles at a high level.

A. OpenAM

OpenAM [1, 2] is an open source software package that enables users to authenticate to multiple applications as part of a single sign-on session without having to re-enter their authentication credentials, and to be authorized to these applications based on their authenticated identities and the policies that have been defined to govern access to each application.

When a user requests access to content or functionality provided by an application, a policy agent (installed on the same machine as the resource that requires protection) intercepts the request and directs it to OpenAM, which authenticates the user using one of a variety of mechanisms that it implements. Once OpenAM determines that the user is authentic, it provides an SSO token to the user's client.

Following authentication, access to the requested content or functionality is determined by the policy agent, which evaluates the policies associated with the authenticated identity indicated by the SSO token. OpenAM enables the creation of authorization policies that control which identities are allowed to access a particular resource and specify the conditions under which this authorization is valid. Based upon the results of the policy evaluation, the policy agent either grants or denies the user access.

OpenAM implements a number of different authentication mechanisms, including HTTP basic, LDAP authentication,

Active Directory, SAML, oAuth, etc. Figure 1 provides an example of OpenAM SSO with Active Directory / Kerberos authentication.

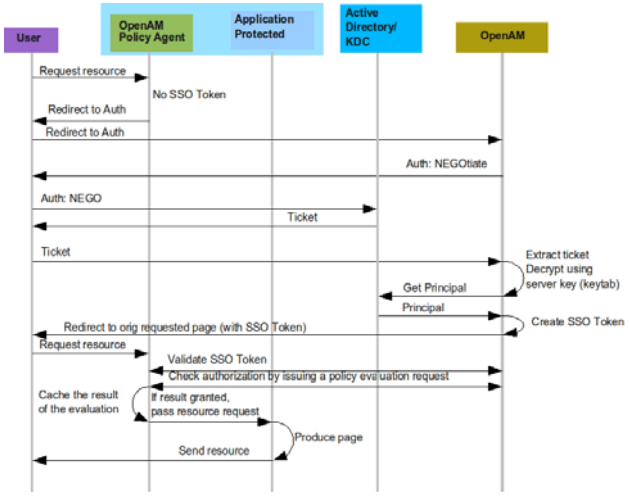

Figure 1. OpenAM SSO Example (from https://wikis.forgerock.org/)

B. *SAML*

SAML [3], standardized by the Organization for the Advancement of Structured Information Standards (OASIS), is an XML-based framework for exchanging authentication, authorization and attribute assertions about security principals among trusting entities. It aims to address use cases including single sign-on, federated identity, and secure web services.

Two distinct types of entities are recognized in a SAML interaction:

- Identity Provider (IdP): The authoritative source of the user's identity. It is the entity that has authenticated the user at some point in time and is responsible for distributing authorization data and/or attributes pertaining to the user.
- Service Provider (SP): The entity from which the user is making a request for a resource. The SP relies on the IdP to identify the user, and is therefore also termed the "relying party".

The SAML standard consists of multiple components and sub-components. The following outlines the main ones:

- Assertions
  - Authentication statement: A principal "x" was authenticated by identity provider "y" (optionally) using mechanism "z".
  - Authorization statement: A principal "x" is authorized to access resource "y".
  - Attribute statement: A principal "x" (or an anonymous principal) has these attributes.
- Protocols
  - Assertion Query Request Protocol: Query existing SAML assertions
  - Authentication Request Protocol: Request assertions tailored to a SAML profile (see the point re. profiles below)
  - Artifact Resolution Protocol: Request resolution of an artifact (a small piece of data referring to a SAML message) into the actual message
- Bindings
  - SAML SOAP Binding: Transport of SAML protocol messages over Simple Object Access Protocol (SOAP)
  - HTTP Redirect Binding: Transport of SAML protocol messages using HTTP redirect messages
  - HTTP POST Binding: Transport of SAML protocol messages in a HyperText Markup Language (HTML) form control
  - HTTP Artifact Binding: Transport of an artifact from a message sender to a message receiver via HTTP
- Profiles: Define how assertions, protocols and bindings can be combined to solve a use case. An example of a frequently-used profile is:
  - Web Browser SSO Profile: Defines SSO from a web browser to multiple services

Figure 2 shows a typical SAML-based SSO flow.

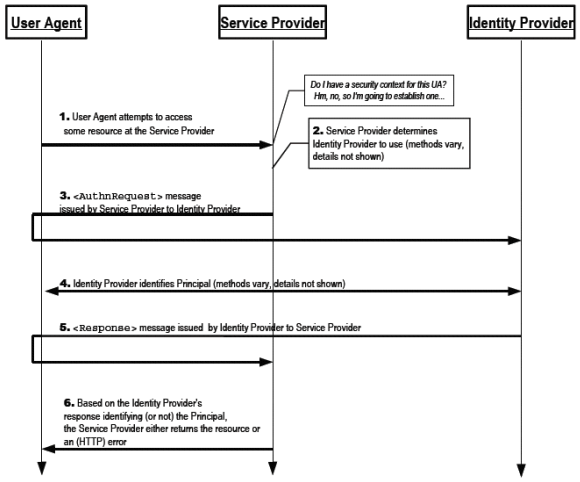

Figure 2. SAML-Based Web Browser SSO (from [4])

III. A SAMPLE UC SSO USAGE SCENARIO

As an example of the type of SSO user experience needed in UC solutions, let us look at the following scenario:

- A user signs on to his/her UC client, which may take the form of a hard or soft phone.
- The UC client authenticates to a SIP proxy server as part of its initial registration sequence. A SIP proxy server routes real-time communication flows among SIP user agents (UAs), i.e., endpoint devices that can issue or respond to SIP protocol messages, and is an intermediary that acts both as a server and as a client for the purpose of making requests on behalf of other clients.
- The user makes a number of SIP calls without further authentication.
- When the user decides to check messages later on, clicking the messaging icon transparently signs on to his/her mailbox hosted by the voice messaging server.
- The user retrieves a voice message that also contains a link to an attachment located on an email server, and obtains the attachment without having to sign on again to the email server.

IV. SSO FOR SIP AND HTTP

We start our discussion of design approaches for UC SSO by first examining SSO for SIP and HTTP.

C. *RFC 5090*

RFC 5090 [5] defines an extension to the Remote Authentication Dial-In User Service (RADIUS) protocol to enable support of digest authentication for use with HTTP and other HTTP-style protocols such as SIP.

The digest authentication mechanism, as defined in RFC 2617 [6], was originally designed for HTTP and later adapted for SIP. It is now considered the mandatory authentication mechanism for SIP UAs.

Digest authentication is based on a challenge-respone paradigm and provides additional security to the information being exchanged by the use of a nonce, i.e., an unpredictable value used to prevent replay attacks. The nonce generator may use cryptographic mechanisms to produce nonces that it can recognize without maintaining state.

In the case of SIP, a valid response to the challenge contains a checksum of the user ID, domain, password, the received nonce value, the SIP method, and the requested uniform resource identifier.

RFC5090 protocol extensions are used between the SIP server / SIP proxy server and an Authentication, Authorization and Accounting (AAA) server, as shown in Figure 3.

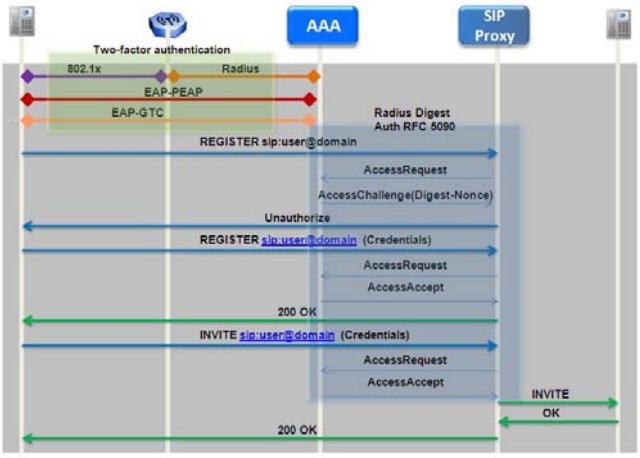

Figure 3. RFC 5090 Authentication

a. *IF-MAP*

Interface to Metadata Access Points (IF-MAP) is a Trusted Computing Group standard to facilitate the sharing of information among different types of systems, including information about users and their roles, network addresses, endpoint status, network activity, physical location, etc. It can also be easily extended to support virtually any kind of metadata.

IF-MAP defines a protocol and an associated database service that provides real-time aggregation, correlation and distribution of metadata among infrastructure systems, management systems, and applications. It is being used to support dynamic network access control, integration of physical and network security, and other functions. This technology helps to reduce integration complexity / cost and facilitate new application combinations.

Some of the areas in which IF-MAP is used include:

- Federation between remote access and network access control (NAC)
- Integration of NAC with endpoint profiling and behavior monitoring, data leak detection and enforcement for unmanaged devices
- Integration of application access control and NAC
- UC security: Providing the flexibility to use a large variety of parameters – including user location, time of day, device type, available networks, user preferences, etc. – to deliver optimal voice, data and video services to users at all times.

b. *SIP Authentication with ID-FF/SAML*

Reference [7] describes an approach where SSO capability based on the Liberty Alliance's Identity Federation Framework (ID-FF) is incorporated into the SIP authentication sequence. ID-FF was the main contribution to OASIS that triggered the work on SAML V2.0 back in 2003, and SAML V2.0 represents a convergence of ID-FF, SAML V1.x and a number of other extensions.

SSO capability is incorporated into SIP in one of two ways:

- Pipeline SSO on SIP REGISTER: Assuming that the SIP UA can be enhanced to support HTTP in addition to SIP, it can "pipeline" the use of the two protocols during authentication as shown in Fig. 4.

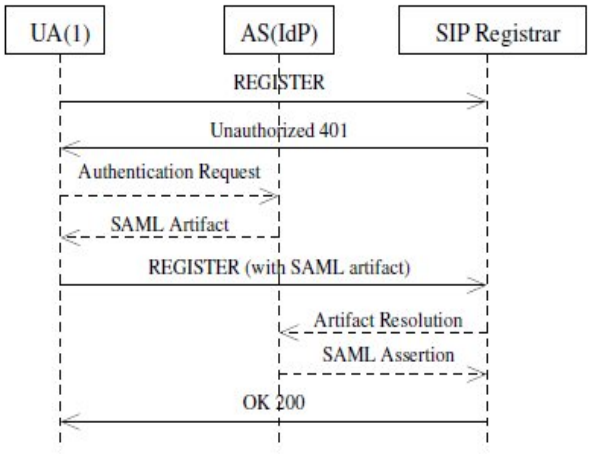

Figure 4. SSO on SIP REGISTER (from [7])

In this figure:

  - The UA sends an unauthenticated REGISTER to a SIP registrar/proxy.
  - The SIP registrar/proxy generates an UNAUTHORIZED response (401) with an enclosed authentication request.
  - The UA extracts the authentication request and forwards it to the IdP.
  - The IdP issues a SAML assertion for the authentication request and returns an artifact pointing to the assertion.
  - The UA resends REGISTER to the SIP registrar/proxy with the artifact.
  - The SIP registrar/proxy verifies the artifact with the IdP.
  - The SIP registrar/proxy sets up shared credentials with the UA which enable subsequent calls from the UA to bypass further authentication.

- SSO over native SIP: The SIP UA performs the entire SSO exchange described above, including SAML assertion request and artifact resolution, over SIP only. It leverages SAML SIP-URI-based Binding (SSUB) and SAML SOAP Binding over SIP (SSSB) for message transport. Note that this approach requires that the IdP be conversant in SIP, and essentially become a special SIP proxy server that performs authentication, while the pipeline approach described above works with existing IdPs which support HTTP.

V. SSO FOR OTHER UC PROTOCOLS

Let us now look at how SSO might work for other protocols used in UC.

As an example, a messaging application might use three protocols, IMAP, SMTP and LDAP, for retrieving messages, sending messages, and accessing directories / address books respectively. The authentication portions of all of these protocols can be based on the Simple Authentication and Security Layer (SASL) defined in RFC 4422 [8].

SASL defines an abstract authentication message exchange between a client and a server:

- The server advertises a list of supported authentication mechanisms.
- The client initiates authentication using one of these mechanisms.
- The server sends a challenge; the client sends a response. This can take place one or more times. The details of the message exchange is a function of the chosen mechanism.
- The server sends the outcome.

A protocol that adopts SASL defines concrete implementations of these steps. For example, in IMAP:

- Supported mechanisms are listed in the CAPABILITY response.
- The client uses the AUTHENTICATE command to request authentication.
- Challenges are conveyed using the command continuation request. This is a line starting with a "+" to which the client must respond.
- The outcome is sent in a tagged status response.

SSO under SASL is typically based on Kerberos [9]. Kerberos enables a client to prove its identity to a server with the help of a third party (the Key Distribution Center, or KDC) trusted by both the client and the server.

When a client wants to communicate with a server, it obtains a session key from the KDC. The KDC encrypts the session key in two ways, once using a secret key that it shares with the client, and again using a secret key that it shares with the server (this is known as a ticket). If the client is not an impostor, it should be able decrypt the session key with its secret key. The client gives the ticket to the server, which should also be able decrypt the session key with its own secret key.

Under Kerberos, a client logs in by using the above mechanism to establish a session with a Ticket Granting Server (TGS). The user's password is typically required to retrieve the session key for the TGS. To access any other server, the client

asks the TGS for a session key and ticket for that server. Since the connection with the TGS is secure, the user does not have to authenticate again.

SASL is sufficiently abstract that a SAML mechanism can be defined with ease. A straightforward implementation might work as follows:

- The client issues a command such as AUTHENTICATE SAML20.
- The server sends a SAML authentication request in a challenge.
- The client forwards the request to an IdP.
- The SAML response comes back from the IdP after the user is asked to prove his/her identity.
- The client forwards the response to the server.
- A web browser is not required, so this approach is a good fit for SAML's Enhanced Client or Proxy (ECP) profile.

Of course, a web browser might very well be present on the client, so a proposal [10] exists to use SAML's web browser SSO profile with SASL:

- The client issues a command such as AUTHENTICATE SAML20.
- The client sends the domain name of the IdP to the server.
- The server uses this information to construct a SAML authentication request as a uniform resource locator (URL) and sends it to the client in a challenge. (The actual SAML message resides in a parameter called SAMLRequest.)
- The client sends an empty response.
- The client passes the URL to a browser, which contacts the IdP.
- The IdP sends the SAML response as an HTTP redirect, which causes the browser to pass it on to the server.

An SSO implementation must have mechanisms to conduct an authentication dialog and to store authentication state. In web browser SSO, the former is often accomplished by redirecting the user to a login page when a request for a protected resource is made. If authentication is successful, a session cookie is deposited to allow subsequent requests to go straight through. Note that merely having a server treat a client's IP address as safe after authentication is not sufficient; the client may have several IP addresses, and there might be network address translation. The client must keep a non-trivial piece of data to prove its identity.

Non-HTTP protocols are used outside of browsers, so the mechanisms for authentication dialog and state storage tend to be different. Many applications have authentication dialogs built in (e.g., logging into voice mail by entering mailbox number and password using a telephone keypad). When using Kerberos-style SSO, the application no longer needs to know the user's password; it is used once to obtain a TGS ticket and discarded.

SAML is not designed to be used directly in checking a user's credentials; its intent is to report that someone has performed this check. Maintaining centralized password management and making the result of application authentication dialogs known to IdPs are the real work items in bringing SSO to non-HTTP protocols.

With non-HTTP protocols, authentication state needs to be shared among several applications running on an endpoint. For example, an e-mail client acquires authentication state after login. When it invokes a web browser to display content, it is expected that the user would not be prompted to login again -- the browser should have access to the e-mail client's authentication state.

VI. Environmental Considerations

In addition to the design approaches discussed in the previous section which focus on the protocol aspects of UC SSO, there are also considerations about the larger network and application environment that any chosen design approach needs to work into.

a. User Provisioning and Identity Mapping

Some considerations from a user provisioning and identity mapping perspective include:

- In UC solutions, where applications provided by a UC vendor are integrated with existing enterprise IT applications (e.g., email, directory, etc.) deployed in the customer environment, a given user usually has different IDs in the enterprise vs. the UC domain. A mapping function is needed as part of the SSO process to translate between these two IDs.
- Within a UC solution, a given user may also have multiple IDs which require ID mapping similar to what was described in the previous point. Examples include alphanumeric ID vs. numeric ID to be entered from the telephone, or just different IDs for different UC products that form part of the solution, such as telephony call server or multimedia conferencing service, etc.
- To reduce repetitive administration effort for the different IDs and associated user profiles, a facility known as flow-through provisioning offered by system management products from various UC vendors can be used. A typical usage scenario for flow-through provisioning enables the administrator to add a user to the customer's directory (e.g., Microsoft Active Directory, Oracle/Sun Directory Server, etc.) and automatically trigger the addition of a telephone call server account, a voice mailbox, a multimedia conferencing user ID, etc. for the user. Since the user profiles for each of these UC products typically include

more attributes (that are specific to the UC services provided) than are available in the customer's directory, a templating approach is often applied. As an example, a number of templates may be defined for groups of telephony call server users (e.g., general employees, call center agents, executives, etc.) to reflect the different features provided to each group, so that it is easy to define a new user profile by selecting one of the templates instead of individually entering the list of attributes that the template captures. Templates can be defined for multi-product combinations as well (e.g., general employees have these telphony features and these voice messaging feeatures, etc.).

- Some UC solutions involve twinning, where a user is provisioned with a telephone and a computing device to be used together in integrated workflows. It would be desirable to provide some form of SSO for the twinned combination of devices, but since each device has its own credentials cache, this is not a straightforward problem to solve.

b. Federating SSO

Many companies attempt to consolidate user account information in a single authoritative data source, typically a directory or meta-directory. Applications can then leverage a centralized authentication system that utilizes this information.

The centralized identity provider model is focused on the internal consolidation of multiple identity management systems into a single system used by multiple service provider applications. A SAML IdP proxy in this model is a bridge or gateway between a federation of SAML IdPs and a federation of SAML SPs, as shown in Figure 5.

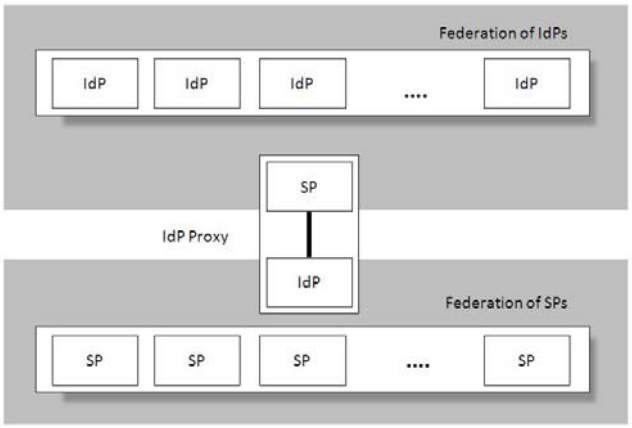

Figure 5. IdP Proxy

To an SP, an IdP proxy looks like an ordinary IdP. Likewise, to an IdP, an IdP proxy looks like an SP. Thus an IdP proxy has the combined capability of both an IdP and SP.

The IdP proxy provides the following functionality:

- Caching of attributes
- Controlled access to the federation of IdPs
- SAML request/response filtering

Figure 6 shows a sample federated SSO flow.

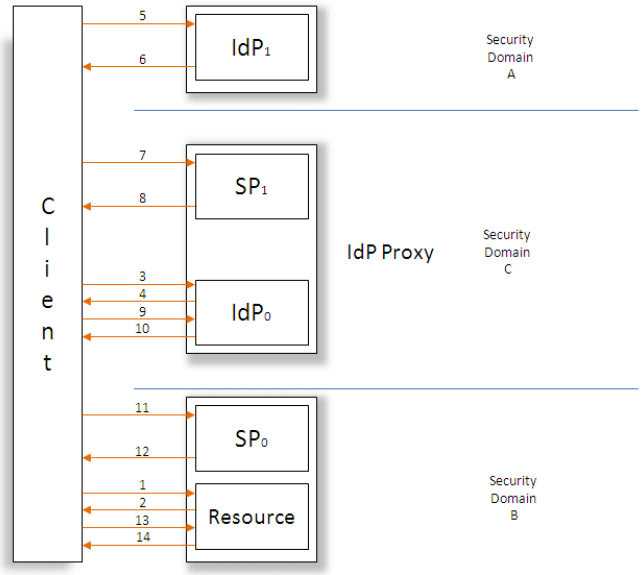

Figure 6. A Federated SSO Flow

In this figure:

1. A browser client requests a web resource protected by a SAML SP (SP0).
2. The client makes a SAML request to the SSO service at IdP0.
3. The client is redirected to the IdP component of the IdP proxy (IdP0), which is protected by the SP component of the IdP proxy (SP1).
4. The request is cached and the client is redirected to the terminal IdP (IdP1).
5. The client makes a SAML request to the SSO service at IdP1.
6. IdP1 updates its security context for this principal, and issues one or more assertions.
7. The client submits the response to the assertion consumer service at SP1.
8. SP1 updates its security context for this principal and redirects the client to IdP0.
9. The client makes a SAML request to IdP0.
10. IdP0 updates its security context for this principal, issues a single assertion, and returns a response to the client. The response may also contain the assertions issued by IdP1 at step 6.
11. The client submits the response to the assertion consumer service at SP0.

12. SP0 updates its security context for this principal and redirects the client to the resource.
13. The client requests the resource.
14. The resource makes an access control decision based on the security context for this principal.

VII. Unified Identity Provider Across Network and Application Layers

In a typical enterprise with today's security architecture, user authentication is performed at the network level and at the application level. Moreover, different authentication mechanisms are used to authenticate the user at each layer. Repeated authentication at multiple levels wastes resources and leads to a poor user experience.

As an example scenario, a user may authenticate at the network level in order to access the network from an endpoint such as a tablet. Next, the user may want to use a voice application such as a UC client, and will then be prompted for authentication again at the application level. The fact that the user has logged in to the network with valid credentials, as well as information about how he/she is accessing the network, are all but ignored when the application level authentication occurs. As a result, network and application security are seen as two different ecosystems of access even though the user is the same.

The network and application layers should form a federation to make secure user access as easy as possible. In this federation, Network Access Control (NAC) acts as a Unified Identity Provider (UIdP), and the application layer reuses the security capabilities of the underlying network to simplify access to applications.

NAC is an approach to network security that attempts to unify endpoint security technology (e.g., antivirus, host intrusion prevention, vulnerability assessment, etc.), user/system authentication, and network security enforcement. It uses a set of protocols to define and implement a policy that describes how to secure access to network nodes by users and devices when they initially attempt to access the network.

NAC may also integrate the remediation process (fixing non-compliant devices before allowing access) to enable network infrastructure elements such as routers, switches and firewalls to work together with back office servers and end user devices to ensure secure operation before interoperability is allowed.

Once the user authenticates through NAC, the NAC system can build a context of the user to be utilized when he/she subsequently tries to access an application. The SP responsible for access to the application can request authorization from the UIdP so that the user does not have to go through another authentication sequence at the application level.

VIII. Security of SSO in the UC Context

So far in this paper we have focused on the design and implementation aspects of SSO as an enabler for unified user experience in UC. Taking a step back, however, it should be noted that in some sense, the use of SSO represents a tradeoff between usability and security, since the single point of sign-on may become a single point of attack.

a. Security of SSO in General

Interest in research into the security of SSO has been rising recently. We give two examples here to provide a flavor of some of the findings.

- In [11], the authors performed automated traffic testing on a number of commercially-deployed SSO systems, such as OpenID (including Google ID and Paypal Access), Facebook Connect, etc. and found 8 serious logic flaws in the web sites' integration of SSO APIs. The approach used involved attempting to deduce semantic information from the traffic flow between browser, service provider and identity provider using an analyzer tool for browser-relayed messages, then trying to modify or replay the flow so as to forge or steal the SSO token and allow the attacker to sign on as the victim.
- In [12], the authors present an analysis of 14 major SAML frameworks and demonstrate that in 11 of them, the integrity protection implemented by applying the XML signature standard can be circumvented by certain XML signature wrapping attacks. What is alarming with this result is that the vulnerabilities identified can be exploited by an attacker who does not control the network or have real-time eavesdropping capabilities, but just works with SAML assertions whose lifetimes have expired. A single signed assertion is sufficient to completely compromise a SAML identity provider. Fortunately, efficient and practical countermeasures to these vulnerabilities are also identified in the paper.

b. Additional Security Considerations in the UC Context

The more applications that a user can access via SSO, the more serious the effect of forged or stolen credentials can be. In the specific context of UC SSO, these are some additional considerations that we can think of as motivation for further work:

- When more protocols other than HTTP are involved in SSO as part of a UC solution, more potential attacks can occur.
- A large variety of communication devices can be used to access UC, including fixed phone sets, mobile phones, tablets, laptops, PCs, and possibly other consumer devices (e.g., cameras based on the same operating systems used in smartphones, etc.) going forward. Even though the variety of devices is not unique to UC, UC does increase the variety and modes of access from these devices. This implies more potential weak points from which SSO can be attacked.
- UC often involves interfacing with older telecommunications products such as telephony call

servers, voice mail systems, etc. in which the code may be more difficult to change due to its heritage or due to feature interactions. This can potentially lead to "make-do" implementations of SSO that are more vulnerable to attack. As an example, it was mentioned earlier in the identity mapping discussion that within a UC solution, a given user may have multiple IDs including a numeric ID to be entered from the telephone. The numeric ID is typically associated with a numeric password or personal identification number for ease of entry on the telephone as well. When soft clients that run on PCs or tablets are developed, they may continue to use a numeric ID and password for ease of interfacing with the telephony call server. If the soft client is integrated into the UC SSO scheme and becomes the point from which the user initiates SSO, there is a question of how much additional access to other applications should be allowed.

IX. SUMMARY AND CONCLUSIONS

In this paper, we have discussed several approaches to providing SSO for multiple UC applications based on different protocols. SSO to SIP and HTTP-based applications and SSO for IMAP, SMTP and LDAP were examined as specific examples. A number of environmental considerations were also explored, such as user provisioning / identity mapping, federation with other identity providers deployed or used by enterprises, and unifying network and application level access.

It is observed that providing centralized password management, making the result of application authentication dialogs known to identity providers, and sharing authentication state among several applications running on an endpoint are the key work items in bringing SSO to non-HTTP protocols.

It was also highlighted that SSO can become a single point of attack if not properly secured, and UC SSO is potentially even more vulnerable due to the involvement of multiple protocols, communication devices and sometimes legacy telecommunications products. Focus on maintaining strong security is essential for credible UC SSO implementations.

Given the importance of SSO in providing a unified user experience (the “U” in “UC”), it is perhaps surprising that this topic has not received much research attention so far. The authors think that SSO for UC is an area that deserves more work, particularly in blending protocol-level design approaches with practical usage considerations and constraints in UC deployments to provide highly usable and secure SSO support for a variety of UC scenarios.

ACKNOWLEDGMENT

Ping Lin, Sunil Menon and Shailesh Patel would like to thank Ravi Palaparthi, Director of System Management and Monitoring, and our colleagues David Ahrens, Steve Warren, and Witold Kaczmarek at Avaya Inc. for discussing and working together on topics related to this paper.

REFERENCES

[1] OpenAM 10 Administration Guide, Lysaker, Norway: Forgerock AS, http://openam.forgerock.org/doc/admin-guide/index.html, 2011-2012

[2] Sun OpenSSO Enterprise 8.0 Technical Overview, Santa Clara, CA: Sun Microsystems Inc., 2008

[3] Security Assertion Markup Language (SAML) V2.0 Technical Overview, OASIS Community Draft 02, Mar. 2008

[4] Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, Mar. 2005

[5] RADIUS Extension for Digest Authentication, IETF RFC 5090, Feb. 2008

[6] HTTP Authentication: Basic and Digest Access Authentication, IETF RFC 2617, June 1999

[7] P. Nie, J. Tapi, S. Tarkoma, J. Heikkinen, “Flexible Single Sign-On for SIP: Bridging the Identity Chasm”, Proc. IEEE International Conference on Communications, 2009

[8] Simple Authentication and Security Layer (SASL), IETF RFC 4422, June 2006

[9] “The Kerberos Network Authentication Service (V5)”, IETF RFC 1510, Sept. 2003

[10] “A SASL and GSS-API Mechanism for SAML”, IETF draft, http://tools.ietf.org/html/draft-ietf-kitten-sasl-saml-09.txt, Feb. 2012

[11] Wang, R., Chen, S., Wang, X.: Signing Me onto Your Accounts through Facebook and Google: a Traffic-Guided SecurityStudy of Commercially Deployed Single-Sign-On Web Services, Proc. IEEE Symposium on Security and Privacy, May 2012

[12] Juraj Somorovsky, Andreas Mayer, Jörg Schwenk, Marco Kampmann, Meiko Jensen, "On Breaking SAML: Be Whoever You Want to Be", Proc. 21st USENIX Security Symposium, Aug. 2012.

