Towards Kerberizing Web Identity and Services

22 December 2008

Abstract

Today authentication and authorization are addressed in an incoherent, and often site-specific, fashion on the Internet and the Web specifically. This situation stems from many factors including the evolution, design, implementation, and deployment history of HTTP and HTTP-based systems in particular, and Internet protocols in general.

Kerberos is a widely-implemented and widely-deployed authentication substrate with a long history in various communities and vendor products. Organizations that currently use Kerberos as a key element of their infrastructure wish to take advantage of its unique benefits while moving to Web-based systems, but have had limited success in doing so.

The authors of this paper have drawn upon their combined experience with supporting large Kerberos deployments, writing and developing web-based identity protocols, and integrating heterogeneous authentication services in order to produce this paper. Thus the views expressed herein are not necessarily those of the MIT Kerberos Consortium or its members.

In this paper we outline the evolution of Web Identity and Services and describe the issues surrounding this complex landscape. These issues are captured within a set of more specific requirements that are deemed necessary to satisfy the relevant stakeholders; these requirements are then framed within the context of some general use cases. We then propose and describe a number of activities that leverage Kerberos to realize these improvements, and present an overall strategy and architectural model for working towards a more cohesive and widely deployed Kerberos-based Web authentication infrastructure.

Authors (alphabetical):

Jeff Hodges
Josh Howlett
Leif Johansson
RL "Bob" Morgan

Table of Contents

1. Introduction

In this paper we attempt to provide a high level overview of how authentication, and to a lesser extent authorization, fits into today's Web landscape and explain Kerberos' place in that landscape. We follow with a brief presentation of "user stories" – simple statements of what end users, service providers and enterprises desire in terms of their security related experiences when using the Web. Next, we describe a number of use cases that are intended to place these requirements within the context of typical scenarios. A number of specific web-relevant security technologies are discussed, and opportunities for extensions and improvements highlighted. We touch briefly on implementation and deployment technologies (e.g. browsers and application development libraries) before closing the paper with a presentation of an architectural model along with recommended opportunities to pursue.

The authors of this paper have drawn upon their combined experience with supporting large Kerberos deployments, writing and developing web-based identity protocols, and integrating heterogeneous authentication services in order to produce this paper. Thus the views expressed herein are not necessarily those of the MIT Kerberos Consortium or its members.

This document surveys a broad spectrum of diverse, complex technologies and is intended to be suggestive of opportunities rather than rigorous. We hope that it will inspire enthusiasm for future efforts intended to make Kerberos work better for the Web.

1.1. Acknowledgments

The authors are grateful for the support of the MIT Kerberos Consortium. We also acknowledge and thank Russ Allbery, Love Hörnquist Åstrand, Steve Buckley, Michael Gettes, Sam Hartman, Ken Raeburn, Nicolas Williams, Tom Yu, and Larry Zhu for their input and comments.

2. Overview

2.1 A Short History of Web Identity and Services

For a system that was originally intended to facilitate collaboration between physicists, it is remarkable how ubiquitous Web technologies have become. It is no exaggeration to claim that the HTTP protocol defined in RFC 2616 is the *de facto* transport for applications operating between internetworked hosts. This is due, in part, to HTTP's success in meeting its authors' goal of defining a "generic, stateless, protocol which can be used for many tasks beyond its use for hypertext transfer", but perhaps mainly to the convenience of using a common transport in an Internet composed of networks with diverse capabilities and policies.

2.1.1 The Primordial Identity Soup

HTTP's standard authentication capabilities, Basic and Digest, are specified in RFC 2617. The most obvious deficiencies of these protocols, such as capture of credentials on-the-wire, were widely appreciated at the time, and the response was to remedy them using SSL (and subsequently TLS). However, the user experience provided by browsers implementing these methods is generally considered unsatisfactory (for example, the bland and "unbrandable" username and password dialog). Consequently, while these methods are often deployed within the Enterprise (typically to protect Intranets and programmatic web services where the user experience is a secondary concern), they were largely ignored by the broader Web development community, where so-called Form-based authentication has become dominant.

Unlike Basic and Digest authentication, which are typically implemented in the web server – at a system level – Form-based authentication is typically implemented at the web application level. The application uses XHTML elements to construct a Form, typically consisting of username and password fields (and increasingly also a *CAPTCHA* dialog). The user enters his credentials into the Form, which is submitted to the application using the HTTP POST method.

The paradigm of using TLS-protected Form submission for user authentication and X.509-based server authentication became extremely common, and it remains the most common approach to authentication and establishing trust. However, this paradigm is not without drawbacks. The most significant problems include:

- Phishing of credentials
- Excessive numbers of credentials
- Inconsistent and complex user experience
- Privacy violations
- Ad-hoc and unstandardized
- not a system-level approach

These drawbacks can be viewed as symptoms of this paradigm's failure to scale with the explosive growth of the Web. This confounded early attempts to improve HTTP authentication (for example, attempts to leverage SASL (RFC 4422)) which might have permitted the use of a Kerberos mechanism) because the Web had not yet attained the scale and significance where the symptoms had developed to the chronic level that we observe today, and so consequently there was no obvious case for significant alterations to protocols or implementations.

2.1.2 Birth of Web Single Sign-On and Identity

The need for some notion of "Web single sign-on" – termed "Web SSO" – was perceived by various disparate parties from the mid-1990s onwards. Often this occurred in a commercial, higher educational, or governmental enterprise setting; many similar solutions were concocted roughly in parallel, and the practice continues to the present day. Some examples include the WebAuth system concocted and deployed at Stanford University (mid-to-late 1990s), which was inspired by SideCar from Cornell University, both of which are Kerberos based. There also is PubCookie from University of Washington; CoSign from the University of Michigan, and Yale's Central Authentication System (CAS). Today there are a number of commercial vendors present in this field.

The first OS-vendor-driven effort to improve Web authentication, providing users with a cohesive experience of web-based "identity" in addition to Web SSO, was Microsoft's Passport service. However, Passport failed to gain any lasting traction, beyond Microsoft's own services, for reasons of trust. Technically, Microsoft's initial Passport SSO protocol was flawed, reducing confidence in this aspect of the system. More critically, Passport was perceived by potential consumers of the system – and others – as a vehicle that could hand Microsoft a monopoly in Internet identity. Passport's importance began to diminish in 2004 when Microsoft canceled the contracts with the most significant consumers of Passport, including eBay and Monster.com.

In parallel with Passport, a Technical Committee (TC) within OASIS – the Security Services TC (SSTC) – was established to define SAML, an open XML-based identity system. This was derived from two earlier proposals, S2ML and AuthXML. SAML was the first technology for federated identity, and is the most successful. The SAML specifications were quickly leveraged by the Liberty Alliance, a broad vendor and "deploying organizations" consortium reacting to the perceived threat from Passport, for their own ID-FF (Identity Federation Framework) specifications.

In contrast to Passport, which was conceived as a wholly centralized identity system, federated identity is an approach in which a website (normally called the Service Provider, or SP) delegates responsibility for user authentication (and sometimes user authorization) to a trusted third party (normally called an Identity Provider, or IdP) with which the user is affiliated. The relationship between one or more IdP(s) and one or more SP(s) that have federated is often called a "federation" (the Liberty Alliance refers to this as a Circle of Trust).

Federated identity does not replace Form-based authentication, nor was it ever intended to. Federated identity is independent of the authentication technique used by the IdP, although the authentication method may sometimes be a property of the authentication event about which the IdP informs the SP. Federated identity is used to centralize authentication (Form-based, Kerberos, X.509 certificate, etc) at the user's IdP, alleviating the SP of the burden of authenticating user credentials and implementing the processes require to issue and manage those. It also means that, in general, a user only needs to maintain a single identity per federation.

2.1.3 Towards Internet Identity

SAML and Liberty Alliance technology deployments have primarily been used for enterprise, e-government and e-commerce scenarios. In the case of the Liberty Alliance framework, the federated identity technology is complemented by non-technological materials, such as federation governance models, security best practices, and deployment guidelines. Some within the Web development community viewed this approach to federated identity as too complex for many non-enterprise use cases – and not scalable to the Internet community at large. A movement developed based on the interests and capabilities of a new generation of Internet users who run their own weblogs and personal web sites, and participate in many social networking services. The design principles of this movement were reliance on Web technology, simple deployments suitable for end-user-managed scripting environments, informal trust and security, and viral growth. The

vision has been identity services driven by the interests of individual Internet users, hence the terms "Internet identity" and "user-centric identity". OpenID technology has been the primary result; it is a web SSO system with several design features intended to support easy adoption and promiscuous interaction between sites. More recently other specifications such as OAuth and OpenSocial have emerged to create a loosely-organized "user-centric" suite.

OpenID has had significant adoption by some major industry players such as AOL, Yahoo!, and Google as well as many smaller sites, though thus far there are many more potential users than real ones. Due to the OpenID protocol design, the features of its implementations, and the interests of its community, OpenID has been limited to low-value, consumer-oriented applications. Thus in practice there has been a split between OpenID and SAML[1], despite the outward similarities of the technologies, leaving many potential adopters wishing for harmony.

The recognition of the diversity of identity technologies, and the diverse requirements of the applications consuming the identities, is a key feature of Kim Cameron's widely referenced *Laws of Identity*. Cameron was a vocal critic of Microsoft Passport and now, as Microsoft's Chief Identity Architect, is motivated to create a better system. He proposes an *Identity Metasystem* that unifies existing identity technologies, including OpenID, SAML, and Kerberos, and is adoptable and sustainable because it recognizes the interests of many stakeholders, especially its end users. The concrete design for the Metasystem is a technology called *Information Card*. It is based on the notion of user visibility and control of identity information via a smart client (*Identity Agent*, or *Identity Selector*), and the unification of interactions involving third-party (often enterprise) Identity Providers with those that are purely end-user-asserted. Information Card in turn is based on underlying *WS-** protocols, which comprise a portion of the overall "Web Services" specification space – to which we turn to in our next section. There are many Information Card implementations available today, but it has yet to see significant real-world use.

2.1.4 Emergence of Web Services

The notion of *Web Services* emerged in the late 1990s, and was intended to provide methods to facilitate "machine-to-machine" interactions that leveraged the same technologies underlying the browser-orientated "user-to-machine" Web. Web services were conceived, in part at least, as a response to the problems (such as transport through firewalls) of using earlier systems such as CORBA and DCOM between loosely coupled domains. Since there are many different interpretations of what web services are, what problems they address, and how they might be utilized, here are the interpretations taken in this paper:

A definition of Web Services:

> A vague term referring to distributed software applications and/or agents that leverage the widely deployed Web infrastructure and protocols in order to compose higher-level applications also typically accessible via the Web. The term is also used to name a category of protocol and API specifications, the implementations of which become enabling components for actual web services applications and/or agents.

What problems are Web Services meant to solve?

> Web services are envisioned to both minimize development time and effort of building distributed loosely-coupled web-based applications and/or features, and to enable making new/engaging/useful/profitable distributed web-based applications and/or features available to deployers and end users.

---

1 In both technological and deployment senses. See *Technical Comparison: OpenID and SAML* for a discussion of especially the former aspects.

A generic Web Services example:

> A hotel reservation web site that is able to additionally offer to end users airline, car, and dinner reservation capabilities, using an ever-changing set of partners, whose user-visible site presentation is also ever-changing, without site reprogramming.

The earliest attempts at Web Services, such as XML-RPC, took a Remote Procedure Call (RPC) orientated approach in which commands, parameters, and the return values are expressed using a small set of data types, encapsulated using XML and delivered directly over HTTP. While simple, this approach encouraged the development of services in which the RPC commands were bound directly to language or service specific functions, leading to criticisms that RPC-orientated approaches were too tightly coupled.

The RPC approach has, to a great extent, been superseded by message-orientated systems. These became widely visible with the submission of the Simple Object Access Protocol (SOAP) 1.1 specification to the W3C (as a "W3C Note") by Microsoft, IBM et al in the spring of 2000. SOAP consists of highly tailorable XML-encoded protocol messages capable of encapsulating arbitrary data, which can be employed in either remote procedure call (RPC) or message passing styles.

Web Services were felt by these practitioners to essentially exclusively consist of SOAP messages conveyed by HTTP even though other transports (notably MQSeries) is common in large-scale homogeneous deployments. It is possible to discern an increased interest in SOAP conveyed by Message Oriented Middleware (MoM) in situations where performance is an issue. The distinction between SOAP as it was originally intended (a transport-neutral message layer) and SOAP as it is often deployed (using HTTP) has implications for the security mechanisms used.

A year later, IBM and Microsoft produced the *Web Services Description Language (WSDL) 1.1* W3C note in Spring 2001. It essentially defines an "interface description language" for SOAP-based protocols, which is purportedly an aid to developers.

Security in both the RPC and message orientated systems – authentication, confidentiality, and integrity protection – was originally left as an exercise for developers and deployers. Two years after SOAP's initial emergence, the WS-Security specification was publicly announced by IBM and Microsoft in Spring 2002. It specifies a framework for attaching essentially any type and/or number of "security tokens" (such as a Kerberos ticket) to SOAP messages, as well as cryptographically binding tokens to messages. This provides a means for data origin authentication on a per-message basis. This can be built upon to provide high-level security abstractions, e.g. for streams of messages.

By 2002, it was clear that there was no industry-wide cohesive vision of what constituted "web services" other than a nominal agreement of them being SOAP-based; nor was there a single "home" for web services specification development. The lack of a home was, and still is, illustrated by the work on various aspects of the broad web services notion being spread across a plethora of fora: privately (e.g. the IBM-Microsoft "workshop process"), in the W3C, in the Web Services Interoperability (WS-I) organization, in OASIS, and in the Liberty Alliance Project.

Despite energetic promotion by vendors as the "standards-based" approach to implementing web services, they are often criticized as being inefficient (owing to the XML overheads) and too complex for many applications.

More recently an alternative approach to both the RPC and message orientated systems has emerged with the development of services based on REST principles. While REST was originally framed in the context of hypermedia systems (of which the Web is one instance), it is now commonly used to describe resource-orientated Web services, where the resource is identified by a URI that yields a representation of that

resource which is operated on using a constrained set of operations; for web services, these are the standard HTTP methods (GET, PUT, POST, DELETE).

An important distinction between these styles of Web services is that REST is descriptive, and not prescriptive like the message-based systems that have been discussed. Therefore, apart from the common use of URIs and adherence to RESTful principles, REST-based services' interfaces share relatively little in common; there are no standards for REST-based web services, although a number a REST-based web services may be considered as *de facto* standards.

2.1.5 Web Services Stacks and Identity

Given the fragmentation of the broader web services community noted above, it is not surprising that more than one "web services stack" has emerged. By "stack" we mean a full-featured specification suite addressing messaging and transport binding, interface description language, security framework, resource discovery, authentication and security token services, and web single sign-on.

The two major extant SOAP-based web services stacks are the Liberty Alliance's *Identity Web Services Framework* (ID-WSF) and IBM-Microsoft-et-al's *WS-**. Both meet the definition of "stack" given above. Both utilize SOAP-based messages and employ WS-Security as well as WSDL. However, they diverge as one goes further "up the stack". Yet, both stacks feature the key notion of federated identity being conveyed between the browser-accessible web-based "world" and backend web services-based systems.

ID-WSF specifies use of SAML 2.0 security tokens (though other token types may be profiled for use) and leverages SAML 2.0's Web Browser SSO Profile for web single sign-on. The authentication service uses a SASL-based authentication exchange thus supporting the plethora of extant SASL mechanisms (including Kerberos). ID-WSF is designed with the notion of a user's identity being a first-order component of the context of interactions. For example, Discovery service requests are implicitly made in the context of the identity of the user causing the request to be made (for example, "*I* am asking for *my* service locations"); it is also possible to express target identity contexts (for example, "*I* am asking for *her* calendar service"). While ID-WSF's specifications explicitly allow for extensions and profiling, these are carefully constrained; ID-WSF was designed to be concretely implementable and deployable directly from the original specification suite with no further profiling required.

This is markedly different from the WS-* stack, whose composition is much more ambiguous. First, IBM and Microsoft have different perspective on what constitutes the WS-* stack; secondly, the specifications are strewn across a landscape including their private "workshop process", as well as OASIS, W3C, and WS-I. The WS-* specifications are designed to be freely "composable" – one can pick and choose which ones to use in solving a particular use case. This is often cited as the rationale for the specifications' style of unconstrained XML element extensibility (in contrast to ID-WSF's constrained approach). Therefore, in order to concretely implement WS-* to solve some given use case(s), one must leverage the WS-I "profile" specifications in concert with the WS-* protocol specifications. The notion of identity attained in the system will depend upon the selection of which aspects of WS-* are implemented as well as the profiling choices made.

2.1.6 Project Concordia

The balkanization of the Web identity landscape has lead to a highly confused marketplace. The large vendors (for example, Microsoft, Sun, IBM, etc) have not, to date, made any serious attempts to resolve their differences. The smaller Identity-focused vendors, such as Ping, who are not aligned to any particular technologies, have typically responded by implementing the most visible technologies.

An exception to this is Project Concordia, a recently formed effort to "drive interoperability across identity protocols". Project Concordia, while originally conceived by members of the Liberty Alliance, has taken pains to adopt a neutral stance and has subsequently attracted participation from all of the main stakeholders. Although it has been operating for approximately 18 months, Project Concordia is still only at the stage of developing use cases; these focus primarily on bridging between identity systems.

Kerberos is not currently represented as one of the technologies within the metasystem that they are addressing.

2.2 Summary

The Web authentication landscape is therefore highly complex and diverse, both in terms of technology and politics. The contrast with the Enterprise space, where Kerberos is virtually ubiquitous, could hardly be starker.

Nonetheless, the use of Kerberos as an authentication mechanism for web applications has a long history in certain communities, notably Higher Education and Research as well as large scale Enterprises. Extending Kerberos to address web authentication in these communities was typically driven by a desire to reuse technology which was well understood and widely deployed. This motivation has become even more pronounced with the wide deployment of Active Directory.

There is, therefore, a significant body of experience relating to the use of Kerberos for the Web and related technologies (Web services, XML, identity federations, and so forth) which can be used to formulate a strategy for future work in this field.

This document sets out to describe a number of missing pieces of the puzzle which, when combined, can help the Kerberos Consortium realize the full potential of Kerberos as an authentication technology for the Web.

3. Stakeholder Requirements

The purpose of this section is to define a set of requirements by unpacking the high level issues identified in the previous section. These requirements are stated as "user stories", which are tools for evaluating the effectiveness of development activities and frequently appear in "Agile" development frameworks. A user story is simply a statement that describes a specific desire by a specific type of user in order to achieve a specific result. We have chosen to describe requirements as user stories in an attempt to make the requirements easier to evaluate from the point of view of the various stakeholders.

The user stories are presented in the table on the next page, grouped by stakeholder type. These types are:

- **End Users**: technically unsophisticated consumers of services provided by Enterprises (in the context of their employment) and Service providers (in the contexts of their employment and general online activities (shopping, leisure, etc))
- **Enterprises**: organizations (often large-scale) that operate complex information systems for the benefit of end users affiliated to that organization (employees, contractors, students, etc).
- **Service Providers**: organizations (often large-scale) that operate complex information systems for the benefit of end users that are not affiliated to that organization (customers, etc).
- **Federated Partners**: two or more Enterprise and/or Service Providers who wish to federate their identity systems, for the purpose of enabling federated management of access to services.

	User Story		
Stakeholders	**Type**	**Code**	**Description**
End Users	Simplicity	U1	End users want to reduce the number of sign-on technologies and credentials that they are required to use to access web-based service providers.
	Transparency	U2	End users want to reduce the number of authentication steps taken when using service providers.
		U3	End users want to use mobile devices when authenticating to service providers.
	Flexibility	U4	End users want to assert different identity information in different contexts, e.g. to be able to "don" different roles when interacting with either the same or different service providers (e.g. to be able to interact with a given bank in the role of either an individual consumer, or an officer of a company which is also the same bank's customer).
		U5	End users want to use untrusted devices (e.g. an airport Internet kiosk or a borrowed device) to access service providers without compromising their credentials.
Service Providers	Simplicity	S1	Service providers that consume identities from third-party identity providers want to reduce and/or minimize the number of sign-on technologies that they are required to support. This applies to both Internet-based and enterprise-based SPs.
	Risk management	S2	Service providers want to be able to manage and minimize the risks they assume in providing their service, particularly with respect to phishing in Financial services and similarly sensitive applications.
Enterprise	Risk management	E1	Enterprise security officers want secure authentication for SOA.
	Simplicity	E2	Enterprise SOA architects want flexible life-cycle management for identities used for SOA.
		E3	Enterprise administrators want to reuse existing Kerberos infrastructure when deploying web applications and web services in order to reduce the cost of security administration.
		E4	Enterprise system integrators want interoperability between web service implementations from major vendors.
	N-Tier	E5	Enterprise identity architects want SSO-support in popular browsers with credential delegation capabilities turned on by default.
		E6	Enterprise identity architects want to be able to extend existing cookie-based SSO systems with support for Kerberos backchannel authentication and credentials delegation.
Federated Partners	N-tier	F1	Deployers of web-based portal services with kerberized backend-services need to be able to use federated identity with N-tier authentication.
	Level of Authentication	F2	Grid services (in environments where PK-INIT is used) in the US Federal sector need to fulfill policy requirements that authentication be done using smartcards.
	Identity Provider Discovery	F3	Service providers with a large number of affiliated Identity Providers requires a way to determine which Identity Provider a user is affiliated with, so that it knows where to request assertions for the user'.
	Technical trust establishment	F4	Federated partners want to reduce the complexity and effort incurred in establishing technical trust between their systems.
	Governance	F5	The IT management at two or more federated partners need to define conventions, or an agreement, governing the use of a federated business process that is secured using Kerberos.

4. Use Cases

This section presents a number of use cases that are intended to illustrate the stakeholders' requirements within the context of some specific scenarios, which are used as a tool to expose the necessary development activities.

These use cases are divided between three categories:

- **Back Channel:** this describes transactions between service providers occur directly, without relying on user-wielded tools (typically, although not exclusively, a web browser) to act as an intermediary. E.g. an e-commerce site interacting directly with a user's banking site, without re-directing communications through the user's web browser. These transactions may either be invoked in response to an action performed by a user (for example, logging into a service provider) or be initiated automatically by software agents.
- **Front Channel:** this describes interactions between service (and identity) providers occurring indirectly via a user-wielded tool – typically, although not exclusively, a web browser. These interactions are normally invoked in response to an action performed by a user. For example, attempting to access a "protected" web page, and being asked to login at one's identity provider as a result.
- **Front and Back Channel combined:** this describes interactions that involve elements of Front and Back Channel activity (for example, a Back Channel transaction that is authorized using a token that was established previously in an earlier authentication event while the user was online).

4.1 Back Channel

4.1.1 Intra Enterprise

A large organization is deploying an increasing number of web services. Initial pilot deployments, which were few in number, relied on the use of shared secrets for authentication and TLS for transport security.

As the deployments of web services expands, the management of the shared secrets becomes an issue. There is also increasing concern that certain properties of shared secret authentication, or the practices used to manage them, while acceptable during the initial deployments on low-value applications (such as hard-coding secrets into applications), are not appropriate for higher-value applications and/or larger scale deployments; these concerns include the security (E1) and manageability (E2) (E3) of credentials. The most common action taken is to move to the use of client certificates which, for an organization of this size and complexity, necessitates the acquisition of a web services governance solution. Such products are often based on an ESB (Enterprise Service Bus) and sometimes employ other transports besides HTTP (for example, various message-oriented protocols such as MQ or XMPP).

The organization would like to use Kerberos with these web services in order to simplify the management of their security requirements. However, the web services are perceived as distinct from the existing business infrastructure, and so consequently these services are often treated as a separate domain isolated from everything except those business systems to which they connect. There is therefore a requirement that the Kerberos administration must integrate into the various web service governance solutions that exist (E4).

4.1.2 Business-to-Business

A business needs to integrate with another business using a SOAP-based web service. One or both of these businesses may already be in possession of an existing enterprise-wide SOAP-deployment similar to the one described in the previous use case. The requirements of the security model are reasonably straightforward (for example, there are no n-tier issues). As before, shared secrets are used authenticate endpoints, using either TLS or IPSec for transport security. The use of any transport for SOAP other than HTTP is very uncommon.

These businesses want to use Kerberos in order to leverage their pre-existing Kerberos infrastructure, and possibly an existing cross-realm trust, in order to avoid establishing a special purpose trust infrastructure (E1) (E2).

The main difference between this use case and the one that was described previously is that typically only a small number of web services are required for each pair of Business-to-Business relationships. Therefore, establishing and managing cross-realm trust must be perceived as a low barrier to the adoption of this trust-model (F4).

The use of the web service is likely to be governed by contract. This may require provisions stipulating the required properties of the security technology used to manage and protect access to the service (F2) (F5).

4.2 Front channel

Business-to-employee and business-to-customer use cases typically involve front-channel exchanges through the user's (customer or employee) browser to the service provider. Most technologies involved in front-channel authentication (for example, SAML Web SSO) currently use Form-based username and password authentication, even if other mechanisms are supported. The lack of SSO impairs the user experience; the lack of mutual authentication increases the risks imposed by phishing.

In both cases, end users expect a single sign-on experience (U1) (U2) and increasingly demand access to service providers in a variety of non-traditional settings, including mobile devices (U3) and Internet kiosks (U5).

In multiparty federations, it is often difficult or impossible for a service provider to transparently determine which identity provider an end user is affiliated with. This leads to service providers implementing their own "discovery" user interfaces, which are frequently confusing to end users (F3).

4.2.1 Business-to-employee

A business wants to provide secure (E1) (E5) (E6) and manageable (E2) (E3) access to both internal corporate resources and resources offered by third-party service providers (such as SaaS providers). Employees want a single-sign on experience, but the business does not want to encourage the user to abuse their Enterprise identity or credentials within inappropriate non-business contexts (U4).

4.2.2 Business-to-customer

A service provider wants to provide secure (S2) and standardized (S1) access to end users affiliated to a very diverse range of identity providers.

4.3 Front and Back Channel Composition

4.3.1 Credentials Delegation

A recently merged company wishes to provide the newly-united workforces with a single web portal providing access to various backend services that, owing to delays in the re-organization of the respective IT functions, will remain administratively separate for some time. The provision of email is a core function of the web portal, which therefore needs to access a set of IMAP servers in these separate administrative domains. To avoid the need to issue new credentials to users, the company wants to use SAML-based Web SSO to provide federated authentication by the SAML identity providers operated by each IT function. These identity providers could provide the web portal with delegatable credentials for their own IMAP server(s) (E5) (E6) (F1).

4.3.2 Level-of-Assurance transport

A service provider has a policy that stipulates which type(s) of credentials are acceptable (this is often called the *level of assurance*). In scenarios where the service provider itself is authenticating the user's credentials, the policy is normally trivial to enforce using technical means because the service provider has visibility of the authentication event.

However in a distributed or federated environment the authentication of user credentials is often performed by a third party (the identity provider), out of the control of the service. There are several examples from within the US federal sector where services may require the use of smartcard technology and where there is also a desire to provide a federated web front-end (normally to extend access to users affiliated with trusted organizations).

However, there is currently no way to communicate information about the authentication event (except in a few corner cases involving PK-INIT). Although SAML defines an element that enables an Identity Provider to express an *Authentication Context*, it is unspecified how this should be used in a Kerberos deployment (F2).

4.4 Deployment Case Studies

This section contains two brief case-studies that are intended to complement the scenarios depicted in the previous sections. The case-studies describes organizations that use Kerberos extensively to support large scale services (e.g. they have deployed the Andrew File System (AFS)), but nevertheless have very different views on web services. This illustrates that there is much work remaining before Kerberos is a natural choice for securing web services and web applications

4.4.1 Stanford University

In 2003, Stanford University redesigned its RPC middleware layer. At that time it was decided not to use web services because they wanted to avoid having to deploy an HTTP stack on a large number of systems and partly because initial experiments demonstrated problems using Kerberos as the basis for web service authentication. Consequently, Stanford decided to design their own RPC layer called "remctl". In this system, endpoints (servers and clients) authenticate themselves using GSS-API. A core design requirement was that Kerberos must be used for authentication because the cost of managing large numbers of username and

passwords (or client certificates) was judged to be too high. Standford has developed multiple implementations of its system (using different application environments/languages currently in use there). Although the project is considered a resounding success there is increasing interest in the use of web services, typically to integrate with vendor solutions; where they do use web services, REST is preferred over SOAP.

4.4.2 Stockholm University

Stockholm University deploys a smaller (compared to Stanford) middleware infrastructure, implemented using SOAP-based web services. HTTPS is used as message transport for almost all web services. Server-side authentication is based on X.509 certificates using commercial PKIs. Client-side authentication is done using Negotiate. Credentials transport is sometimes done using GSS-API credentials delegation over Negotiate and in some cases using s4u2self (aka constrained delegation). Web services are often cross-platform (e.g. clients and servers run on different application environments) but servers are mostly written using .NET or J2EE. Stockholm University also uses PubCookie as an enterprise SSO which is sometimes used to delegate Kerberos credentials to browser-facing web applications.

4.4.3 Conclusions

There is a clear need for interoperability and cross-platform support for Negotiate and credentials delegation and service deployment should be lightweight. Identity lifecycle management for services is an important consideration, which is often overlooked in the early stages of web service deployments. A Kerberos-based authentication mechanism for REST-based web services is also urgently needed.

5. Security Technologies and Opportunities

The security technologies that are relevant to the Web can be divided into four overall categories: *transport*, *application*, *message* and *composite*. This section discusses these technologies, how they compare and contrast with Kerberos, and presents potential opportunities for complementing these technologies with Kerberos.

It is not, however, unreasonable to ask what value Kerberos might bring to an already crowded and complex landscape. However, as described in The Role of Kerberos In Modern Information Systems, Kerberos possesses a number of properties, sometimes unique, that might contribute towards a more satisfactory Web authentication architecture.

5.1 Web-Specific Security Technology Requirements

Therefore, this section will initially discuss the general properties that security technologies employed in the Web context ought to satisfy, and highlight where Kerberos may be able to contribute.

Note that this is not an exhaustive set of overall security requirements, nor is it an overall security analysis.

5.1.1 Authentication Method Independence

It is often desirable to make the method of end user authentication independent of the mechanism that is used to subsequently to prove the user's identity to a service, particularly within federated environments. This independence allows identity providers to implement any user authentication method that satisfies the policies of the service providers, without those services needing to support the authentication method itself.

While Kerberos deployments commonly only use shared secret authentication, the protocol does support other methods of authentication.

5.1.2 Mutual Authentication

Mutual authentication describes a situation where both parties of a communication obtain attestation as to the identity of the other party, ideally upon initial authentication and effectively simultaneously. Mutual authentication is an essential part of building trust between users and systems, and the lack of mutual authentication in many Web authentication dialogs is the root causes of many security and privacy violations on the Web, such as phishing.

Kerberos provides mutual authentication as a normal feature of its protocol operation. In other security technologies, for example TLS, authentication of both parties is optional – typically the server is authenticated to the client, and not the other direction.

5.1.3 Authorization

Authentication is normally a necessary criterion for satisfying many typical service providers' policies, but is rarely sufficient in isolation from authorization. For example, service providers commonly require evidence

that an authenticated user has been accorded a set of privileges necessary to satisfy the service provider's security policy.

Kerberos service tickets can be embellished with authorization data describing the privileges accorded to the user. This facility is used extensively by Microsoft's Kerberos implementation to assert a user's authorizations within a data structure called the Privilege Attribute Certificate (PAC).

5.1.4 Credentials Delegation

It is useful to distinguish between two different scenarios that delegation can be used for: *Enterprise* and *User driven*.

5.1.4.1 Enterprise Web SSO and Credentials Delegation

Identity management in large-scale deployments of web applications has resulted in the deployment of governance solutions for web single sign-on (Web SSO). Open source products like Shibboleth, WebAuth, CAS, and PubCookie, as well as commercial products like Passlogix, ActivIdentity, Avencis SSOX, are often deployed as central points of authentication for multiple web applications in the enterprise. These systems are often seen as meeting several different requirements involving compliance (e.g. Sarbanes-Oxley (S-OX)) and auditing. However, the enterprise SSO makes Kerberos credentials delegation quite difficult. Some products (e.g. PubCookie) have the ability to forward TGTs (Kerberos “Ticket Granting Tickets”) from the point of authentication to the web application but these are exceptions rather than the rule.

Combining enterprise SSO with kerberized backend services is also often difficult if one wants to retain the "end-to-end principle"; which in this context requires maintaining a cryptographic association between the ticket sent to the backend service and the end user credential used to originally authenticate with the IdP in the single sign-on environment. The success of Active Directory led to the introduction (by Microsoft) of a new way to do credentials delegation: s4u2self (aka "constrained delegation"). In this model (as opposed to the end-to-end end model) the KDC delegates the right to a "service" principal to impersonate a user to a set of services. In the webmail example, the service principal would be associated with the web application itself and the KDC would delegate to it the right to obtain IMAP service tickets for any user. The web application is trusted to properly authenticate the user, by whatever means, allowing the KDC to delegate part of its responsibility to this service principal. This idea is sometimes called "air gap" security and while not as strong as end-to-end n-tier authentication it is nevertheless often good enough for many applications.

5.1.4.2 User Driven Credentials Delegation

The advent of "Web 2.0" web applications, notably social networking sites like Facebook, Linked-In and large scale-software-as-a-service (SaaS) providers like Hotmail and Gmail led to the creation of yet another type of credentials delegation paradigm: user-driven delegation in which a user is asked to provide authorization every time a service needs to be able to access another service with the rights of that user. A typical example is a social networking site requiring access to the user's email service's address book. The OAuth protocol (discussed below) supports this type of case-by-case delegation. Although the Kerberos V5 specification has allowances that would permit implementations to craft such "user driven" behavior, such behavior is not typically implemented and/or presented to users.

Kerberos provides the ability to forward tickets; that is, the ability to delegate credentials to remote services where they can be used by that service to act on behalf of the user. When the service is a simple traditional host-based service (for example, ssh or telnet) the delegation process is relatively uncomplicated. This

changes dramatically with the advent of web-based applications which need access to user credentials in order to authenticate to kerberized backend services; this is sometimes referred to as "n-tier". The classic example is the web front-end to a kerberized IMAP server. As long as the authentication mechanism for the web application is basic access authentication or form-based authentication (so that it has access to the user password) the web application can obtain a TGT – but at the cost of compromising the secrecy of the user's Kerberos password.

However, credentials delegation in Kerberos still remains one of its truly unique properties – no other widely deployed security protocol has the ability to pass credentials between members of a multi-tiered architecture.

5.1.5 Security Token Size

The "chatty" nature of some web interactions, and the necessity of each individual HTTP request being authenticated, implies that the security token size will likely be an important consideration.

It is worth noting that the inclusion of authorization information in the Kerberos protocol can have adverse side-effects. When Kerberos is used as an HTTP authentication mechanism (c.f. “5.3.1 Negotiate”, below), as opposed to being hidden behind an SSO abstraction layer such as an intra-Enterprise or Federated SSO system, any extension to the ticket that adds significantly to the payload of the HTTP handshake may be noticeable by users in terms of overall system response time.

Experience with Active Directory in large-scale deployments, where the PAC can often grow to ~20k, indicates that the use of Kerberos can significantly degrade a Web application's performance, especially when the browser needs to open and process several connections in order to render a single "page" to the user. Modern browsers are optimized for keeping multiple HTTP connections open in parallel and it is not uncommon for single user views ("pages") to result in hundreds of connections to the server. Microsoft reports customer feedback to this effect.

The same would likely also be the case for any other "large" chuck of information attached to service tickets, and/or the HTTP requests themselves.

5.1.6 User Interaction

Some security technologies, including Kerberos, are designed with the notion of signing the user on once; subsequent authentication interactions with participating entities (such as other service providers) occurs in manner that is transparent to the end user.

However, in some contexts – especially those in which principal attributes beyond baseline principal name and secret are conveyed – it may be necessary for there to be some interaction with the end user, at least whenever the user agent is being compelled to convey more information beyond some configurable baseline minimum, in order to prevent violations of the end user privacy.

This is one of the features of Information Cards' "Identity Selector" user agent, and the underlying "Identity Selector Interoperability Profile" protocol.

5.2 Transport security

Transport security is provided by the underlying transport protocol. The security service only acts at the socket or datagram layers between two hosts, and bindings to high-level security measures tend to be non-existent. Credentials are usually issued to hosts (if datagram-based) or services (if socket-based).

Kerberos-based alternatives (existing and proposed) for securing transports include:

- Kerberos with TLS[2]
  - A specification describing a Kerberos-based cipher suite for TLS exists (RFC 2712), and has seen at least two implementations (OpenSSL, JGSS), but is regarded as having shortcomings[3].
  - A recently-expired Internet Draft (draft-santesson-tls-gssapi-03) describes extensions to RFC 4279 to enable dynamic key sharing in distributed environments using a GSS mechanism, and then import that shared key as the "Pre-Shared Key" to complete the TLS handshake. The TLS Working Group has argued that user principal authentication is an application-level concern, and so this work appears to have stalled.
  - Nicolas Williams[4] advocates another approach nominally termed “GSS-TLS” (which he also refers to as the "TLS-wrapped-as-GSS” design) where TLS is essentially specified as a GSS-API mechanism. He argues there are implementation advantages to this approach, among other things. There is no Internet-Draft or other document presently describing this approach, however.
  - University of Michigan CITI has produced a suite of middleware leveraging Kerberos for the provision of a lightweight PKI known as: Kerberos-leveraged PKI (K-PKI), and occasionally identified by one of its components, KX.509. The protocols do not appear to be officially standardized as yet. They are described in this CITI Technical Report: *Kerberized Credential Translation: A Solution to Web Access Control*. The LionShare project at Penn State produced a similar but more general service called SASL-CA that supports Kerberos and other authentication methods.
- Kerberos with IPSec
  - Some deployments apparently use Kerberos in concert with IPSec. We do not have good data on how well this works, but there are obvious issues with NAT boxes and firewalls in the path, as well as heterogeneous environments.

---

2 Some Kerberos stakeholders, of both deployer and vendor types, have strongly expressed desire for a new, more viable solution in this space that eliminates the need for certificates and a PKI. Some deployer stakeholders favor a Kerberos-based TLS ciphersuite approach, a la RFC2712. They feel that their development staff and the applications they create/maintain already “understand” TLS, hence a Kerberos-based ciphersuite approach will integrate into their IT shops and deployed infrastructure more easily. Although we agree to a first order approximation, we however feel that they may be overlooking subtle-but-important aspects such as having an existing TLS-aware application be handed a Kerberos service ticket rather than a certificate+key – consider the amount of GUI that would need re-working. Additionally, there are considerations with respect to incurring trade-offs between one's investments in one's deployed applications and one's deployed “TLS concentrators” (and possibly various network boxes, e.g. firewalls), as well as considerations with respect to TLS implementations in various deployed platforms, e.g. servers, that may be minimalist in-kernel, or exist in “user land”.

3 It assumes unfettered client access to the KDC, the Kerberos session key is used as the pre-master secret, violations of Kerberos Version 5 library abstraction layers, incompatible implementations from two major distributions (Sun Java and OpenSSL), and its lack of support for credential delegation (the latter three points being described in the introduction of draft-santesson-tls-gssapi-03).

4 Long-time contributor to various IETF Kerberos- and security-related Working Groups.

5.2.1 Opportunities

Specify the use of Kerberos with TLS

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O1	U1, U2, S1, E1, E3, E4.	High	High	High	TLS, Kerberos, GSS-API.	IETF Kerberos and TLS WGs.

The ability to use Kerberos via GSS-API as a ciphersuite option in Transport Layer Security (TLS) would avoid the problems inherent in using Kerberos at the application layer within HTTP (see the discussion on Negotiate for more information), thereby allowing general protection of HTTP-based services including REST. The same technique could also be used for the other styles of Web services (RPC and Message), thereby providing a common approach to securing Web services.

Investigate Leveraging and Standardizing K-PKI

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O1a	U1, U2, S1, E1, E3, E4.	Medium	Medium	High	TLS, Kerberos, GSS-API, KX.509 et al.	IETF Kerberos and TLS WGs, NSF NMI, CITI.

Short-lived Kerberos-leveraged key pairs and certificates could act as a key link between PKI-based transport security and Kerberos-based infrastructure.

5.3 Application security

Application security is provided by the application protocol itself; credentials are usually allocated to an application, and the users accessing those applications. Applications may, in some circumstances, leverage user credentials to act on a user's behalf. This is referred to as "impersonation" and/or "delegation" depending upon the detailed context. Application security mechanisms are typically defined independent of transport security, although in practice implementations of application security often depend on transport security.

5.3.1 Negotiate

Negotiate is the collective name of a loosely defined set of HTTP authentication mechanisms that provide a simple 2-way GSS-API handshake with the HTTP server. The most commonly deployed variant, "HTTP Negotiate Authentication Scheme" defined in RFC 4559, *SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows*, uses the SPNEGO GSS-API mechanism (RFC 4178). Other implementations use the Kerberos 5 GSS-API mechanism (RFC 4121) instead. To add to the confusion, the term SPNEGO is sometimes said to denote both the GSS-API mechanism and the HTTP authentication mechanism.

It is worth noting that Negotiate authentication does not protect the HTTP request because it is sent unprotected during the GSS-API handshake, which is itself encoded within an HTTP header. This

authentication method is therefore not advised in some use cases; in particular, REST-style web-services cannot in general be protected using Negotiate alone (this is the case with all other authentication techniques that are conveyed within HTTP message headers, including Basic and Digest), which implies use of transport layer security mechanisms, e.g. TLS, in conjunction with Negotiate.

Negotiate is observed to suffer from a lack of mutual authentication and effective support for multiple-roundtrip GSS-API mechanisms in a "real world" Web filled with "TLS concentrators" and HTTP proxies. Earlier attempts (in the IETF) of introducing SASL-based authentication for HTTP showed that in order to gain wide acceptance any multiple-roundtrip HTTP authentication mechanism has to be able to deal with consecutive HTTP/1.1 connections being sent to different TCP endpoints (e.g. if a concentrator is deployed at the server). This implies that the authentication mechanism needs to support some form of state management. At least two different proposals have been put forward for solving this problem; one in draft-johansson-http-gss where state management is done in the HTTP layer and one in draft-zhu-negoex where state management is done in the GSS-API layer.

5.3.1.1 Opportunities

Update Negotiate

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O2	U1, U2, S1, S2, E1, E3, E4.	Low	Low	Medium	Kerberos, GSS-API.	IETF Kerberos WG, Microsoft, other implementors of Negotiate.

Investigate the proposed modifications to Negotiate to perform state management.

5.3.2 Information Card

Information Card is an identity technology based on the notion of a smart client, called the *identity selector*, that resides on the end user's device. Under the control of a user, the identity selector can interact with a service provider, and also optionally an identity provider, to perform application sign-on and delivery of user information to the service provider.

The Identity Selector has a user interface that presents the user's authentication options in the form of cards. A card can be "managed", meaning that it is issued by an Identity Provider, and requires authentication to that identity provider when used. Alternatively a card can be "self-issued", meaning that it is generated within the identity selector itself, and uses a key pair managed by the identity selector to authenticate to the service provider.

These interactions use the WS-Trust protocol, part of the WS-* suite of specifications. WS-Trust is intended to be a "meta" authentication protocol that can, in theory, permit a security token to be delivered to the service provider in whatever format it requires, independent of the method the client uses to authenticate to the identity provider.

Although Microsoft is the initial developer and evangelist of Information Card, many other companies and projects are developing compatible software and contributing to the technology specifications. The Information Card Foundation (http://www.informationcard.net/) was formed in 2008 to promote the technology. Microsoft's identity selector implementation is called CardSpace; because it is the best known

implementation, many people incorrectly refer to the whole technology using this name. The Information Card concept was originally developed by Kim Cameron of Microsoft as part of a comprehensive design called the "Identity Metasystem." The technology is now being standardized in an OASIS technical committee called the Identity Metasystem Interoperability TC (http://www.oasis-open.org/committees/imi/).

The Information Card specifications define the authentication methods supported between identity selectors and identity providers (in the case of managed cards); Kerberos is one of these methods. The other methods include X.509 certificates, username and password, and the self-issued card (which is technically a SAML token).

5.3.2.1 Opportunities

Kerberos authentication from Identity Selector to identity provider

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O3	U1, U2.	Low	Low	Medium	Kerberos, WS-Trust, Information Card.	Enterprise deployers.

In a managed-card scenario the Identity Selector can authenticate to an identity provider using Kerberos, via the *WS-Security Kerberos Token Profile* bound to WS-Trust. An identity provider supporting Kerberos authentication would be an application server from the Kerberos point of view. However, none of the existing Information Card identity provider implementations supports Kerberos authentication, and so this presents an opportunity for improvement. Microsoft's identity provider (which probably won't be available until 2009) will support all four defined methods, including Kerberos.

Kerberos token delivery to the service provider

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O4	U1, U2, S1, S2, F3.	Low	Low	Medium	Kerberos, WS-Trust, Information Card.	Enterprise deployers, Kerberos community.

WS-Trust operates by delivering security tokens to service providers to authenticate clients. WS-Trust claims to be token agnostic – meaning it can carry anything – but for a token to be used its nature and processing must be specified and implemented by both the issuer and consumer of the token.

As mentioned previously, Information Card defines the use of SAML tokens for self-issued cards and these have commonly been used as tokens in managed cards also. It would be possible to define a Kerberos token. Such a token, which presumably would be based on a KRB_AP_REQ, would be generated by the IdP, carried by WS-Trust from the IdP via the user's Identity Selector to the relying party, where, after removing the WS-Trust wrapper, it could be processed by Kerberos software at the RP. There is no such token defined at this time. Recent comments from Microsoft about supporting cross-technology delegation imply that it might be implementing Kerberos tokens in WS-Trust as described here, hence there may be an opportunity for producing an interoperable implementation.

Identity Selector as the standard authentication user interface

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O5	U1, U2, S1, S2, F3.	Medium	Medium	High	Information Card, Kerberos.	Enterprises, end users, application protocol designers.

Information Card technology has been promoted primarily and energetically as an approach for browser authentication to web applications, given the urgency of a better web authentication solution. The identity selector concept, however, can apply to any instance of user authentication. This might involve extending identity selectors to support other protocols, or adding the use of WS-Trust for authentication to existing systems. In particular, the Identity Selector could be a user interface for Kerberos initial authentication in some scenarios.

5.3.3 OAuth

OAuth (http://oauth.net/) is a specification for interoperable support of delegated access to web-based resources. It is designed to support a particular common use case: a user of one web-based service (for example, a photo-sharing service, called the OAuth service provider) has protected resources (for example, photos) that the user would like to make accessible via another web-based service (for example, a social-networking service, called the OAuth consumer). This is often done today by the consumer service obtaining from the user their username and password on the service provider. The goal of OAuth is to avoid this practice. This done by defining service points to permit an access token to be produced by the service provider and delivered to the consumer via the browser. This token can then be used (after some transformation) to access the service provider via an OAuth-defined HTTP authentication method.

OAuth was developed by an ad hoc group, based on similar specs from a number of web technology companies (e.g. Google, Yahoo!). Its current version is 1.0. There are several implementations for different languages and platforms, and some initial deployments. Google in particular defines extensive support (http://code.google.com/apis/accounts/docs/OAuth.html).

OAuth has no defined relationship to Kerberos or any other existing authentication technology at this time. Note that since the tokens are both issued and consumed by the service provider, their internal format and semantics can be whatever the service provider wants. For example, the validity period of the token is not defined in the spec or represented in the protocol; it is entirely up to the issuer.

The standard OAuth scenario is referred to as "three-legged" since it involves the consumer, the provider, and the user who must indicate consent and convey the access token (via their user agent). There is interest in "two-legged" scenarios where the OAuth HTTP authentication method is used simply between consumer and provider, not on behalf of some user. In this case it would be an alternative to other HTTP authentication methods such as Basic or X.509/SSL.

5.3.3.1 Opportunities

OAuth augmented with SAML or Kerberos

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O6	U1,U2,E3,E6,S1	Medium	High	High	Kerberos, SAM, OAuth.	End Users, Enterprise, Service Providers, OAuth community, SAML community

There has been recent discussion about profiling OAuth with SAML in order to permit the encapsulation of SAML assertions (that might convey, for example, information about the user) in the OAuth protocol exchange, in conjunction with the otherwise OAuth-specific cryptographic credentials. As an alternative, Kerberos tickets could be sent either instead of or in conjunction with such SAML assertions. This could be appealing for a Kerberos-oriented OAuth service provider.

OAuth Kerberos-based trust bootstrap

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O7	U1,U2,E1,E2,E3,S1	Medium	High	High	Kerberos and OAuth expertise.	End Users, Enterprise, Service Providers, OAuth community, SAML community

OAuth defines an initial registration and key-exchange step between consumer and service provider. In a scenario where consumer and provider use Kerberos this step could be skipped if a method to bootstrap OAuth keys from Kerberos were defined.

5.3.4 OpenID

OpenID (http://openid.net/) is a specification for a web SSO system. It has many interoperable implementations and has been deployed at many sites, including large Internet providers such as Yahoo! and AOL. Originally proposed by a single developer, it was first formalized (in version 1.1) by an ad hoc group. Based on industry-wide interest, the technology is now managed by the OpenID Foundation (OIDF), which deals with intellectual property rights, specification process, and technology evangelism. The current version of the core authentication specification is 2.0, but there are several extensions at various levels of maturity.

OpenID's founding principles are (a) being web-oriented, (b) being simple, and (c) promoting ubiquitous interoperability of deployments. Being web-oriented means that the scope of the system is browser-based sign-on to web resources, the protocols use plain old HTTP (or HTTPS), and the preferred user identifiers are HTTP URLs. Being simple means, at least in version 1.1, addressing only mainstream authentication use cases, crafting the specification in a concrete fashion (i.e. it is not readily profilable), and avoiding reliance on other technologies (e.g XML – by using only name-value pairs to convey data). Promoting interoperability means providing a one-size-fits-all concrete solution to the "IdP discovery problem" (via URLs as userids), and providing a dynamic key establishment mechanism.

Like many web SSO schemes, OpenID leaves user initial authentication out of scope, so OpenID Identity Providers (OPs) can use any means to authenticate users. In practice almost all use username/password with form-based authentication. Since OpenID's deployment model promotes spontaneous interaction between OPs and SPs that may not have interacted before, the phishing vulnerability inherent in any username/password web SSO system is arguably exacerbated.

5.3.4.1 Opportunities

Negotiate authentication against OpenID identity provider

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O8	U1,U2,E3	Low	Medium	Medium	Kerberos and OpenID.	End Users, Enterprise, OpenID community.

OpenID identity providers could use Kerberos to authenticate web users using SPNEGO/HTTP. The Identity Providers would the existing Kerberos infrastructure, enabling enterprise users to sign on to OpenID service providers thereby reducing the risk that users might reuse their enterprise credentials.

OpenID leveraging Kerberos for trust bootstrap

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O9	E1,E2,E3,F4	Medium	Medium	High	Kerberos and OpenID.	Enterprise, Service Providers, OpenID community.

Normally OpenID identity providers and service providers engage in a dynamic key setup protocol at their first interaction. This protocol is subject to DNS attacks (at least). If the identity provider and service provider were part of a common Kerberos infrastructure (perhaps via cross-realm trust) this setup protocol could use Kerberos and thus be performed more securely.

5.4 Message security

Message security describes an approach where application protocol messages are individually cryptographically bound to security tokens they convey directly and/or indirectly (by reference). With such an approach, the protocol messages may be conveyed over different transports (e.g. HTTP, SMTP, TCP, XMPP, etc), transit intermediate systems, or be retained on endpoint systems, while retaining intact their originating security context (modulo any allowances for message modification in-transit). This is in contrast to security context derived from the transport layer which typically reflects only the immediately communicating endpoints, that is it is "hop-by-hop", and the originating security context is lost if a message transits more than one hop, or tier.

Thus message security provides discernible endpoint-to-endpoint benefits in the face of multiple hops and/or differing transports.

5.4.1 Security Strategies for Message-based architectures

Good message-based implementations understand that there are multiple SOAP transports and provide an interface for plugging in new ones, however in practice SOAP transports other than HTTP rarely occur when performance isn't an issue. Where multiple web services are deployed from the same application server it is common to use a web service governance solution which abstracts the service business logic from the transports.

SOAP method calls are either secured at the transport or message layer or both. Even though various web services specifications and stacks specify and promote using WS-Security to secure SOAP messages, it is common to use transport-layer security especially in the cross-domain cases. In the latter cases it is not uncommon for IPSec to be used to provide a tunnel between the client and the server and for the HTTP-based SOAP-calls to be unauthenticated in this tunnel. When tunnels are not practical it is common for HTTPS (HTTP/TLS) to be used in combination with a username and clear-text password.

The reasons for using Kerberos may be any of:

- Kerberos is easier to use once deployed and if the service endpoint is one of many published by "A" then maintaining username and passwords will be impractical for the same reasons that maintaining lots of password-based user identities would be impractical. In this case Kerberos provides life-cycle management for the identities used to secure the web service.
- Kerberos is the default security service used by the application framework used to develop and deploy the service endpoint.
- Kerberos is chosen because security policy mandates the use of Kerberos for identities.

There are therefore two ways to authenticate SOAP using Kerberos (if we ignore the possibility of using Kerberos with IPSEC for now):

- by securing the transport (this was discussed in a previous section of transport security).
- by securing the SOAP messages.

5.4.2 WS-Security Kerberos Token Profile

The WS-Security Kerberos Token Profile defines how to use Kerberos service tickets (specifically, the KRB-AP-REQ message) as a security token in conjunction with the WS-Security specification, the latter specifying a particular approach to obtaining SOAP message security. Although it is a piece of the broad WS-* composite web services specification suite, it may be employed in other contexts, for example, the Liberty ID-WSF web services framework also employs it.

The Kerberos message is attached to the SOAP message using the <wsse:BinarySecurityToken> element, however the acquisition of the Kerberos service ticket contained in the KRB-AP-REQ message is out-of-scope. Six different types of KRB-AP-REQ message are supported, including: RFC 1510-style, RFC 4120-style and their equivalent GSS-API encapsulations. The octet sequence is encoded using the indicated method.

5.4.2.1 Opportunities

Update WS-Security Kerberos Token Profile specification

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O10	U1,U2,E1,E2,E3,E6,F1.	High	High	High	Kerberos, WS-Security, OASIS	End Users, Enterprise

This specification does not support the KRB-AP-REP message, and so consequently neither Kerberos mutual authentication, nor GSS-API channel bindings, are supported. In practice, therefore, this specification needs to be combined with transport layer security (e.g. TLS).

5.5 Composite Security

This category refers to technologies where various aspects of the other categories are employed in various fashions. Such technologies are often defined in extensive, full-featured specification suites. Some technologies require further profiling in order to yield an implementable system that will address real-world use cases. Others feature concrete profiles as a part of the specification suite.

5.5.1 OASIS Security Assertion Mark-up Language (SAML)

SAML provides a suite of XML-based protocols that enables different security domains to exchange security data about subjects. A subject might be a user principal in a particular use case, but doesn't have to be, similar to how Kerberos is often deployed with explicit service principals.

Broadly, the SAML specifications define:

- **Assertion:** a security token that can express a subject's authentication status, authorizations and other general attribute information.
- **Protocols:** that are primarily for requesting assertions about subjects.
- **Bindings:** defining how protocols are bound to certain transports (such as HTTP or SOAP).
- **Profiles:** constraining and concretely defining the use of assertions, protocols and bindings to realize specific use cases (such as the "Web SSO Profile").
- **Federation Metadata:** a format for expressing information (primarily configuration data, such as a public key) about SAML endpoints for consumption by partner endpoints.

Although many existing SAML profiles focus primarily on generating, obtaining, and conveying SAML assertions in order to federate identities in different security domains for Web SSO applications or web services, the architecture of SAML allows discrete functional elements to be re-used in other contexts, allowing a broad range of potential applications. In other words, SAML itself is an abstract framework (see: How to Study and Learn SAML). For example, the SAML assertion is re-used in many other technologies – even technologies, such as WS-Federation implementations, which are competitive with SAML as a whole.

SAML does not stipulate the use of any particular trust model(s), although X.509-based trust establishment and TLS is widely used in practice.

5.5.1.1 Opportunities

Leverage SAML metadata to enable large-scale Kerberos cross-realm communities

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O11	U1,U2,E2,E3,F4	Low	High	Medium	Kerberos, SAML	End users, Enterprise, Service providers.

SAML Metadata is a key enabling technology for large-scale multi-party SAML federations (e.g. the InCommon Federation or the UK Access Management Federation). Via metadata trusted site configurations can be aggregated, annotated, signed, and distributed. There is no similar capability with Kerberos cross-realm technology today, which may be a barrier to growing Kerberos-based trust communities. The SAML metadata specification could be extended to support describing Kerberos entities. Alternatively, XRDS is a similar technology that is beginning to be used in OpenID, XRI, OAuth, and other related identity systems. It could also be profiled to describe Kerberos entities.

Investigate, document and promote existing methods of using Kerberos to authenticate against a SAML identity provider

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O12	U1,U2,E2,E3	Low	Low	Medium	Kerberos, SAML.	End users, Enterprise.

There are a number of ways a SAML identity provider can use Kerberos today to authenticate end users, such as validating user-supplied Kerberos credentials or Negotiate. The main benefit offered by these approaches is that they allow end users to access resources protected by both SAML and Kerberos, offering end users greater convenience. These methods are often achievable with little or no modifications to software, but nonetheless are not widely used. This activity would aim to promote the use of these methods.

Specify a SAML profile supporting the generation of SAML assertions containing Kerberos tickets ("Kerberos-in-SAML")

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O13	U1,U2,E1,E2,E3,E6,F1,F2.	Medium	High	High	Kerberos, SAML	End users, Enterprise, Service providers.

This would allow a SAML service provider to acquire a Kerberos ticket as part of a SAML transaction. This could be used by the service provider to authenticate, as another principal, against a Kerberos protected resource, thereby providing support for n-tier web-based use cases (such as the webmail use case). This would offer greater convenience to users and possibly reduce the abuse of user credentials that might otherwise be provided to applications with delegatable credentials.

Extend Kerberos to permit the inclusion of a SAML assertion in KDC-issued authorization data ("SAML-in-Kerberos")

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O14	U1,U2,U4,E1,E2,E3,E6,F1,F2, F3.	Medium	High	High	Kerberos, SAML.	End users, Enterprise, Service providers.

This would allow a Kerberos service to acquire a SAML assertion within a Kerberos ticket. This could be used to complement Kerberos-based authentication and/or provide richer authorization.

Specify a SAML profile that allows automatic discovery of the end user's identity provider based on the Kerberos-realm in the GSS-API authentication request

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O15	U1,U2,U4,F3.	Low	High	Medium	Kerberos, SAML.	End users, Service Providers, Enterprise.

This would reduce or eliminate the use of GUI-based IdP discovery (the commonest form of IdP discovery), which scales poorly and introduces considerations such as localization, accessibility, and phishing, significantly improving the end user experience.

Include a SAML attribute assertion within a GSSAPI protocol exchange

Code	Contributes towards addressing requirement(s)	Effort	Risk	Reward	Skills	Stakeholders
O16	U1,U2,U4,E1,E2,E3,F1,F2,F3.	Low	High	Medium	Kerberos, SAML, GSS-API	End users, Service Providers, Enterprise.

While not strictly tied to Kerberos this may provide a way to tie in with other uses of the GSS-API naming extensions.

5.5.2 Liberty Alliance Identity Web Services Framework

The Identity Web Services Framework (ID-WSF) is a set of specifications from the Liberty Alliance that builds on SAML to provide a full-featured web services stack that leverages WS-Security as its encapsulating security token format. Thus it is possible to leverage Kerberos-based authentication in an ID-WSF-based web services context, although at this time the limitations noted above with respect to the WS-Security Kerberos Token Profile will apply.

5.5.3 The WS-* Suite

WS-* denotes a suite of web services specifications promulgated by IBM, Microsoft and various collaborators. Unlike the SAML and ID-WSF specifications, only a few of the WS-* specifications have

been subject to an open standardization process and subsequently their quality can be generously described as variable.

Of these specifications, there are three that are central to the overall security of the typical WS-*-based system:

- **WS-Security:** this specification defines how security tokens are bound to messages. WS-* has defined security token profiles for SAML and Kerberos, amongst others, that can be leveraged by this specification. WS-* officially employs WS-Security-based *messaging* security techniques almost exclusively.
- **WS-Trust:** this specification defines how to exchange (request, issue, renew, cancel) security tokens with web services. These tokens may be used in a variety of ways; for example, as evidence to authorize access to a web service.
- **WS-Federation:** this specification is a specialization of WS-Trust that defines how to exchange security tokens in a federated context.

In the Web and Kerberos context, the most important of these is WS-Trust as it provides the mechanisms for using tokens (that might be a Kerberos ticket or a SAML assertion) to establish and leverage trust. WS-Trust also plays a key role in the Information Cards technology, another specialization of WS-Trust.

5.5.3.1 Opportunities

The most straightforward opportunity is to enhance the WS-Security Kerberos Token Profile as noted above. Such enhancements ought to subsequently carry throughout the WS-* suite by virtue of many of the suite's other specifications directly depending upon WS-Security and its token profiles.

6. Implementation and Deployment Technologies

This section lists the browsers and application environments and libraries deployers often use when constructing web-based applications. Specific features and ramifications of their use in the context of a web application environment are mentioned.

6.1 Browsers

Most browsers support Negotiate by default, but most either disable the feature (requiring the user to enable it manually) or limit its use to "local" sites or sites explicitly trusted. It is unclear what the threat-model looks like and this should be investigated further.

6.1.1 Safari

Safari supports Negotiate with SPNEGO since Tiger. There is an information card identity selector for safari here: http://www.hccp.org/safari-plug-in.html.

6.1.2 Mozilla/Firefox

The Mozilla and Firefox browser family includes native support for Negotiate using the Kerberos GSS-API mechanism. This feature is not turned on by default though, and must be turned on manually by editing the "prefs.js" file. Credentials delegation is supported but must also be turned on manually. There are a few plugins implementing Information Card for Firefox/Mozilla. It is unclear if any of them support Kerberos authentication for Information Card-based IdPs.

6.1.3 Internet Explorer

Negotiate was invented by Microsoft and is widely deployed through the implementation in the Internet Explorer family of web browsers.

6.1.4 Konqueror

Konqueror is the browser included in the KDE desktop framework. Konqueror supports Negotiate and enables Negotiate for sites in the same domain as the client. There is no support for Information Card yet in Konqueror.

6.2 Application Environments and Libraries

This section touches briefly on Kerberos and Negotiate HTTP mechanism capabilities in a number of application development environments. The number of potential environments of interest is large, and details of capabilities complex. Future work should focus on those environments of most importance to the Kerberos community stakeholders.

6.2.1 IIS

IIS naturally has native support for Negotiate using the SPNEGO GSS-API mechanism.

6.2.2 .NET

The .NET framework supports the WS-Security Kerberos Token Profile as well as Negotiate natively.

6.2.3 Apache

Apache has add-on modules supporting Negotiate using both SPNEGO and Kerberos GSS-API including http://modauthkerb.sourceforge.net/.

6.2.4 Perl

The LPW::Authen::Negotiate module implements Negotiate using the GSSAPI perl module which in turn uses either Heimdal or MIT Kerberos libraries. This means that LPW::Authen::Negotiate can support both SPNEGO and Kerberos GSS-API.

6.2.5 Ruby

There is a GSS-API library for ruby which similar to the perl GSS-API library supports either MIT or Heimdal Kerberos (http://www.h5l.org/). However the Ruby HTTP client library does not natively support Negotiate (it does support NTLM though). There have been working patches but they are not maintained. Ruby is one of the language of choice for "Web 2.0" applications especially in combination with the Rails framework. These applications are known to prefer REST-style web services to more "enterprise"-style SOAP-based web services.

6.2.6 Java

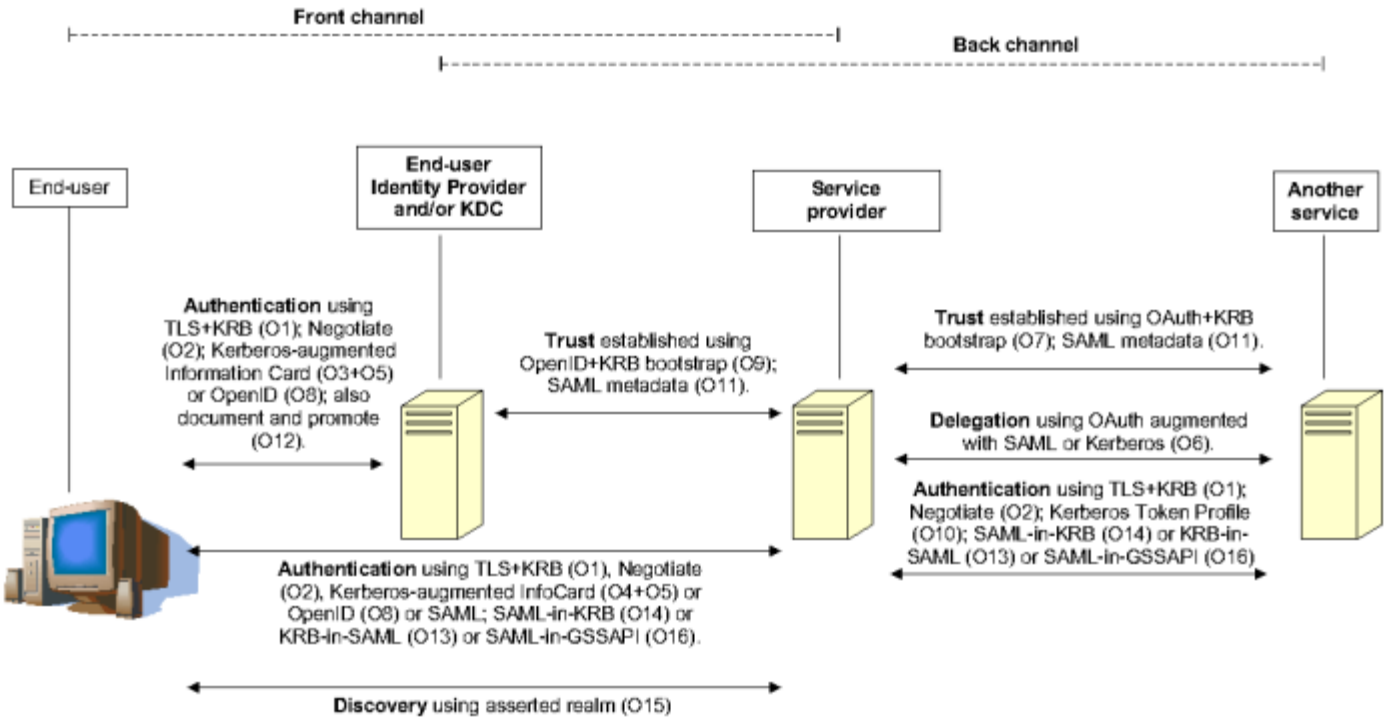
Java has native support for the Kerberos GSS-API mechanism since JDK 1.4. A number of implementations of Negotiate exists for various application servers (e.g. Tomcat and JBoss). There are not free implementations of SPNEGO for JDK 1.5 and later although there are at least two commercial implementations. Since JDK 1.6 there is native support for SPNEGO.

6.2.7 WSS4j

WSS4J is the primary free implementation of XML security (from the apache project). It does not include an implementation of the Kerberos token profile.

7. Analysis

The diagram below summarizes the opportunities previously proposed.



We believe that, as a whole, the opportunities revealed through the back channel use cases are, in general, addressed more easily than those from the front channel use cases, in a technical sense. We also believe that satisfying the back channel use cases would yield greater benefits for relatively less effort. Many of these opportunities are shared with the front channel use cases; therefore, greater emphasis should be placed on them, even if they cannot be immediately realized within the front channel context because of other dependencies. Consequently, we recommend that the Kerberos Consortium focus primarily on the opportunities within this class of use cases[5]. These opportunities, in suggested priority order, are:

- O1 - Specify the use of Kerberos with TLS
- O14 - SAML-in-Kerberos: Extend Kerberos to permit the inclusion of a SAML assertion in KDC-issued authorization data
- O13 - Kerberos-in-SAML: SAML profile supporting the generation of SAML assertions containing Kerberos tickets
- O10 - Update WS-Security Kerberos Token Profile specification
- O11 - Leverage SAML metadata to enable large-scale Kerberos cross-realm communities
- O2 - Update Negotiate

Four of these opportunities (O1, O2, O13, O14) are variations on the theme of improving or enhancing the transport of Kerberos and/or SAML tokens. All begin with specification work. Various stakeholders are strongly in favor of having a more viable solution to O1, as noted in our discussion in Section 5.2, above.

5 Note that some of the "authentication" opportunities, notably O1, and O2, span the front- and back-channel delineation.

Opportunities O13 and O14 are superficially similar, in that they share SAML, Kerberos and a goal (authentication) in common, but are quite different architecturally. In particular O13 is, in essence, somewhat of an elaboration of the conventional SAML Web SSO Profiles, whereas O14 could be also used in other non-Web contexts. Both O13 and O14 may depend on or be significantly enhanced by O11, but are arguably more complicated and so given marginally higher priority.

- Opportunity O10 is likely to yield benefits immediately, unlike O11 which is essentially a common dependency for other longer-term opportunities whose requirements will become more apparent as those opportunities are taken forwards. We therefore recommend that O10 is given a higher priority.

Another interesting distinction to draw between these activities is the potential impact on the client. It is likely that neither O13 nor O14 would require changes to existing client Kerberos or browser implementations, whereas O1 and O2 would probably require updates to the TLS provider and client browser implementations respectively. The inertia exerted by legacy client systems, particularly in the consumer context, would impede wide proliferation of implementations. But they can be utilized in either the front or back channel domains, and O1 may be quite useful from an enterprise perspective.

The priority given to O2 probably depends on the importance given to the ability for a client to authenticate using Kerberos directly to a service provider rather than being used merely as evidence for some other type of authentication method, such as a SAML assertion. Given the widespread use of technologies such as SAML it would appear pragmatic to leverage these initially, particularly given the challenges presented by Kerberos cross-realm operation. While Negotiate has some flaws, these are generally manageable where it is used within a domain (for example, to authenticate against an Enterprise's identity provider). Consequently, we recommend that O2 is given a lower priority.

Strategically, the Kerberos Consortium should consider whether its future work emphasizes Kerberos as a technology complementing existing Web authentication technologies, which we've termed"Complementary Kerberos". Or whether its efforts should emphasize strategies that are less tightly bound to other authentication technologies, and the Web in general, which we've termed "Kerberos Centric".

A Complementary Kerberos approach could be realized more quickly but might be viewed as dilutory (O13, and to a lesser extent O14) by some stakeholders. While the Kerberos Centric approach would require significant updates to clients perhaps incurring greater cost and delay.

Some of the opportunities within the front channel use cases are reasonably straightforward (in a technical sense) and could deliver benefits relatively quickly:

- O12 - Investigate, document and promote existing methods of using Kerberos to authenticate against a SAML identity provider
- O3 - Kerberos authentication from Identity Selector to identity provider
- O8 - Negotiate authentication against OpenID identity provider

We believe that opportunity O12 would be useful as a means of encouraging adoption of Kerberos for web authentication, raising awareness of the Kerberos Consortium's interest in this area, and "laying the foundations" for the more sophisticated opportunities discussed in this document. For example, MIT's Touchstone system could provide a case-study that would be of considerable value to other organizations attempting to leverage Kerberos for web authentication. Opportunities O3 and O8 would usefully complement existing federated systems, and these might also contribute towards O12.

Two other front channel opportunities exist that would yield significant benefits, but incur greater effort and time to realize owing to the necessary client updates.

These are:

- O5 - Identity Selector as the standard authentication user interface
- O15 - Specify a SAML profile that allows automatic discovery of the end user's identity provider

The primary benefits of these opportunities are that they would improve the user experience and reduce the exposure to phishing attacks. These opportunities are therefore somewhat duplicative in their goals, but technologically highly dissimilar. The most appropriate approach would probably become more apparent when the strategic question concerning "Complementary Kerberos" or "Kerberos Centric" is resolved. In the former case, O15 is probably more appropriate; in the later case, O5 is probably more appropriate.

In summary, the table below provides prioritized lists of the opportunities discussed to this point, delineating the two overall strategic approaches described above and offering an approximation of the relative risk and effort associated with each.

Opportunity code	Opportunity name	"Kerberos Centric" Relative Priorities	"Complementary Kerberos" Relative Priorities	Effort	Risk
1	Specify the use of Kerberos with TLS	Medium (2)	High (3)	High (3)	High (3)
14	SAML-in-Kerberos: Extend Kerberos to permit the inclusion of a SAML assertion in KDC-issued authorization data	High (3)	High (3)	Medium (2)	High (3)
13	Kerberos-in-SAML: SAML profile supporting the generation of SAML assertions containing Kerberos tickets	Low (1)	High (3)	Medium (2)	High (3)
10	Update WS-Security Kerberos Token Profile specification	High (3)	High (3)	High (3)	High (3)
11	Leverage SAML metadata to enable large-scale Kerberos cross-realm communities	High (3)	High (3)	Low (1)	High (3)
2	Update Negotiate	High (3)	Low (1)	Low (1)	Low (1)
12	Investigate, document and promote existing methods of using Kerberos to authenticate against a SAML identity provider	Low (1)	High (3)	Low (1)	Low (1)
15	Specify a SAML profile that allows automatic discovery of the end user's identity provider	Low (1)	Medium (2)	Low (1)	High (3)
3	Kerberos authentication from Identity Selector to identity provider	Low (1)	Medium (2)	Low (1)	Low (1)
5	Identity Selector as the standard authentication user interface	Medium (2)	Low (1)	Medium (2)	Medium (2)
8	Negotiate authentication against OpenID identity provider	Low (1)	Medium (2)	Low (1)	Medium (2)
Approximate overall *relative effort*[6]		37	45		
Approximate overall *relative risk*[7]		50	63		

6 $\sum (Relative\ Priority * Effort)$

7 $\sum (Relative\ Priority * Risk)$

The remaining opportunities are:

- 01a - Investigate Leveraging and Standardizing K-PKI
- O4 - Kerberos token delivery to the service provider
- O6 - OAuth augmented with SAML or Kerberos
- O7 - OAuth Kerberos-based trust bootstrap
- O9 - OpenID leveraging Kerberos for trust bootstrap
- O16 - Include a SAML attribute assertion within a GSS-API protocol exchange

We are not at this time assigning priorities amongst these remaining opportunities.

8. Recommendations to the MIT Kerberos Consortium

On the basis of the analysis presented in the previous section, we conclude by presenting a set of recommendations.

8.1 Recommendation 1: Determine the overall strategic approach in consultation with relevant stakeholders

In our analysis, we presented two possible directions: "Kerberos Centric" and "Complementary Kerberos". The risks and effort associated with both courses are roughly equivalent, although we believe that the latter is likely to yield faster results given the opportunity to leverage existing Web authentication systems and minimize alterations to the client.

These options are not, of course, mutually exclusive; it would be possible to develop a hybrid strategy that, for example, assumes a "Complementary Kerberos" direction initially to achieve some results sooner and transition towards a "Kerberos Centric" direction once this strategy's more complex dependencies are resolved. This is the recommendation of the authors although we accept that, given the complexity and number of variables involved, there are probably a number of other approaches that would be difficult to argue against.

We recommend that the Kerberos Consortium continues this discussion with all the relevant stakeholders.

8.2 Recommendation 2: Initiate activities to address those opportunities stakeholders expressed interest in

There are several opportunities which stakeholder feedback indicated are of higher priority, and whose applicability is overall independent of the strategic direction chosen; these are:

- O1 - Specify the use of Kerberos with TLS
- O14 - SAML-in-Kerberos: Extend Kerberos to permit the inclusion of a SAML assertion in KDC-issued authorization data
- O13 - Kerberos-in-SAML: SAML profile supporting the generation of SAML assertions containing Kerberos tickets
- O10 - Update WS-Security Kerberos Token Profile specification
- O11 - Leverage SAML metadata to enable large-scale Kerberos cross-realm communities"), and
- O12 - Investigate, document and promote existing methods of using Kerberos to authenticate against a SAML identity provider

The MITKC should consider initiating work on these items in parallel with carrying out the other recommendations.

8.3 Recommendation 3: Plan and prioritize the most critical subsequent activities

In our analysis, we presented two possible strategic directions. While recognizing that other strategies probably exist, in this recommendation we provide a list of what the present authors consider to be the most urgent opportunities for each of these strategies.

These activities should be planned in parallel to the development of the activities initiated previously to ensure that these key outputs form part of a cohesive architecture.

- "Kerberos Centric"
    - O2: Update Negotiate.

- "Complementary Kerberos"
    - O3: Kerberos authentication from Identity Selector to identity provider.

8.4 Recommendation 4: Develop an overall architecture

It will be necessary to develop an overall architecture that presents a conceptual model describing how the relevant components are coupled and work together.

While certain elements of this architecture are relatively obvious (thus the recommended activities in recommendations 2 and 3), it is the authors' opinion that the space of acceptable and feasible architectures (given the requirements and constraints that we know about *today*) is of significant volume and therefore difficult to make concrete without making grievous assumptions.

However, in the process of planning and implementing the activities recommended in recommendations 2 and 3, it is likely that new constraints (technical, political or otherwise) will present themselves. These will, most probably, significantly constrain the space of acceptable and feasible architectures and highlight gaps that might not be evident to the authors at the present time. If gaps are identified their impact should be analyzed and, if necessary, steps taken to remedy them.

We therefore recommend that the Kerberos Consortium develops an overall architecture, and that this should proceed in parallel with the activities initiated in the previous recommendations.

9. About the Authors

JEFF HODGES

Jeff Hodges is a consulting Protocol Architect working in the areas of identity, distributed infrastructure, and security. His interests lie in the nature of "online identity" and its realization via composition of authentication, security, and directory technologies.

He participates in the OASIS Security Services Technical Committee (SSTC/SAML), various Internet Engineering Task Force (IETF) working groups including those whose topics involve security as well as Session Initiation Protocol (SIP), as well as the "Concordia" effort.

In the recent past, he contributed to the Liberty Alliance effort as an editor and co-author of several of the Liberty ID-WSF and ID-FF protocol specifications. Earlier, he served as co-chair of the OASIS SSTC, shepherding and contributing to the development of the Security Assertion Markup Language (SAML) v1.0, as well as contributing to v1.1 and v2.0.

His prior work has included contributions to the design of the LDAPv3 directory access protocol, as well as contributing to the design and deployment of Stanford University's SUNet ID and Registry/Directory infrastructure. He's held architecture, engineering, and management positions at NeuStar, Sun Microsystems, Oblix, Stanford University, and Xerox.

JOSH HOWLETT

Josh Howlett is the Technical Lead on JANET(UK)'s Middleware programme. In this role he guides a number of Middleware activities; these primarily concern the technical infrastructure and associated policy required to support the management of access to networks, applications and content within the JANET community. He also contributes to Internet2 MACE, TERENA ECAM, and a number of Middleware-related activities within TERENA and DANTE's network development programme. Outside of the Research and Education community, he is an Invited Expert within the Trusted Computing Group, a member of the UK Information Commissioner's Working Group on User-Centric Identity and an occasional contributor to IETF activities.

LEIF JOHANSSON

Leif Johansson is the Deputy CIO for Technology at Stockholm University. Leif is part of the technical expert team for the Swedish Alliance for Middleware Infrastructure (SWAMI) and a member of the technical reference group of the Swedish University Network (SUNET). He is also part of the operations team for the SWAMI identity federation, responsible for the technical operation of SAML federation infrastructure.

Leif is a member of Internet2 MACE, Terena ECAM and an active member in the IETF where he works on matters related to identity, security, applications and information modeling. Leif is or has been a participant in other Terena activities such as SCS, refeds, tf-ace, tf-emc2 and tf-mobility.

He is the author of internet-drafts on (among other things) Kerberos server information models and HTTP authentication using GSS-API. He is the SUNET contact for the Liberty Alliance as well as a contributor/owner of various open source-projects.

Leif works on information modeling and model driven architecture using UML and semantic web technology (OWL/RDF) in the andromda.org and openmetadir.org projects.

He has previously worked on DoD orange-book certification and computer viruses for the Swedish defense materials administration. Leif has co-authored papers on topics ranging from identity management to scientific papers in mathematics where he has conducted research in the field of A_{\inf} spaces. Leif has long operational experience with core technologies such as Kerberos, X.509, AFS, SAML, and LDAP.

RL “BOB” MORGAN

RL "Bob" Morgan is Senior Technology Architect for the Computing & Communications Department at the University of Washington. In this role he contributes to designing, implementing, and documenting distributed computing and security infrastructure for the UW. He is the Chair of the Middleware Architecture Council for Education (MACE), providing guidance for the Internet2 Middleware Initiative. He is a primary contributor to a number of Internet2 middleware projects, notably Shibboleth, a system for secure access to inter-institutional web resources. He is also active in standards activities with the Internet Engineering Task Force (IETF) and the Organization for the Advancement of Structured Information Standards (OASIS), where he has helped to develop the Lightweight Directory Access Protocol (LDAP) and Security Assertion Markup Language (SAML) standards.