Why is Kerberos a credible security solution?

Kerberos is a technology that allows for strong authentication in open, distributed networks. It is a credible security solution for four main reasons:

1. **Kerberos is mature.** It has been widely used and widely studied for a long time. In security, that counts for a great deal.
2. **Kerberos meets the requirements of modern distributed systems**. It was developed in response to a well defined and clearly thought through set of requirements for secure authentication in an open environment with insecure communications links; it has turned out that those requirements closely match the requirements of modern distributed systems operating over networks based on Internet Protocols.
3. **Kerberos is architecturally sound.** It is designed around a clear set of architectural and functional abstractions; that architectural soundness has allowed it to evolve over time, and make it easy to integrate it into other systems. This same architectural soundness makes it easy to analyze how Kerberos will behave.
4. **Kerberos is already in place.** Kerberos is already integrated into most popular operating systems and many widely-used software applications. It is an integral part of today's IT infrastructure.

A bit of history

Kerberos was originally developed for the distributed computing environment that MIT deployed in the 1980s as *Project Athena*. At over two decades, Kerberos is, by Internet standards, an *elder technology.* To put things in perspective, Kerberos and DNS emerged at roughly the same time. When it comes to security, maturity is intrinsically desirable: open, widely used and widely examined technologies are the most predictably secure, and the least likely to be vulnerable to new exploits.

Like DNS, Kerberos emerged from a clear focus on a well-defined task. In environments like *Athena*, commonplace today but novel at the time, an individual user can be expected to use many different workstations (and many different services, such as file access and printing, hosted on many different servers). The original question facing Kerberos's designers was how to provide *single sign-on*: to allow users to access a variety of systems and services without needing to enter their user ID and password repeatedly (or, even worse, without needing to remember and provide different user IDs and passwords for each of the different systems and servers they use).

While the original Kerberos was designed for a single environment, it was based on a clear architectural model that is applicable to other environments. Because of this solid architectural foundation, it has been possible for Kerberos, like DNS, to grow and to support a scale and breadth of function scarcely imaginable at the time of its original creation. Today, Kerberos provides not only single sign-on, it also provides a robust general framework for secure authentication in open distributed systems.

Cryptographically secure, architecturally sound, and easily integrated as a component in other systems, Kerberos was widely embraced as a way of providing a core set of security services for many distributed systems projects and developments, and is today an integral part of many computing environments. Nearly all popular Operating Systems (OSs) have Kerberos built-in, as do many important applications, and it is widely used by network equipment vendors. Like DNS, Kerberos is a service that most users never even realize they're using, but it enables many of the interactions that take place over modern enterprise networks.

Getting the requirements right…

A key to successful systems development is a clear sense of requirements—both the functional requirements that describe what a system must do, and the non-functional requirements that describe, among other things, the environment in which it must operate. As with most technical solutions, getting the requirements right at the outset is the most important step. The widely-publicized shortcomings of some modern network services (e.g., NFS, WEP, web browser/server interactions) often stem from a failure to get the requirements right.

What is remarkable about Kerberos is not just that the original architects had an astute understanding of the relevant requirements, but that they remained committed to taking all necessary steps to fully address them, many of which seemed like "overkill" at the time. Because many of the characteristics of the Athena environment also apply to the modern Internet and enterprise intranets, the approach remains fundamentally sound today.

The key requirements on which Kerberos is based include:

- **Single sign-on for users:** The developers of Kerberos realized that users should not have to accept constant authentication challenges from every service they access. The modern web clearly illustrates the many headaches that come from not meeting this basic requirement. What is remarkable about Kerberos is that it actually does enable mutual authentication for each and every communications session that a user establishes with services, but it manages these authentication procedures "under the covers" once a user initially authenticates successfully through the Kerberos system.
- **Operate within distributed systems based on an open Internet model:** Project Athena was a novel academic computing environment based entirely on distributed workstations and servers operating over an open Internet with few, if any, external or internal security boundaries. In this sense, it was much like the public Internet is today, with a student body known for its skill and creativity in exploiting any flaw in any system.

- **Integrate with existing technology:** As a system requirement, though, Kerberos had to coexist and support a wide variety of standard and custom services using the IP protocol stack. It also needed to support binding of application sessions to verified IP addresses of the communicating parties.
- **Mutual authentication of parties must take place before any information is exchanged:** Not only must a server be convinced of a client's authenticity, but also the client must be convinced of the server's authenticity. In an open environment, rogue servers stealing client data are just as much a threat as imposters impersonating real clients. Many authentication systems tend to focus on user authentication, which leaves users exposed to attacks from rogue services that pose as the service a user relies upon. For example, today's phishing attacks tend to exploit weak authentication of services to entrap users and capture login credentials or other sensitivity information. To Kerberos's credit, it fully supports mutual authentication, though authentication of the service by the client is optional.
- **Passwords should never be exposed during authentication:** A password that is never disclosed or sent over a network is much more difficult for an attacker to purloin. Consequently, Kerberos authentication of users does not require that passwords be presented to the authentication service. Instead, the Kerberos authentication service uses cryptographic protocols whereby the user can prove possession of a password without actually revealing it.
- **Central administration of authentication secrets:** In a distributed environment, it would be awkward in the extreme to maintain shared secrets such as passwords on every client and server that needs to authenticate requests. Furthermore, distributing shared secrets across many systems increases potential vulnerabilities in direct proportion to the number of systems—a problem exacerbated by the "weakest link" phenomenon. Kerberos addresses this requirement by maintaining a centralized database that is distributed across only a few authentication servers. While overall security is critically dependent on protecting this central database, it is much easier to harden a few special-purpose servers against attacks than to protect many general-purpose systems. The central control over authentication secrets also makes it easier to issue new credentials, revoke existing ones, and recover from compromised credentials.
- **Use of cryptographic measures:** At the time Kerberos was originally developed, cryptographic measures were rarely employed as a means for achieving security, and it was actually considered radical to require use of encryption. Fortunately, the Kerberos developers recognized that cryptography was more than just clever mathematics; it was essential to prevent disclosure of authentication information in an open environment to which hostile parties have access. Despite numerous non-technical challenges, the Kerberos developers stuck with this requirement, and their convictions have withstood the test of time, and many adversaries along the way. By adopting newer cryptographic algorithms as they have come along, Kerberos has maintained this core advantage.
- **Support arbitrary distribution of services and users:** Kerberos imposes no restrictions on how users interact with services in a distributed environment, or even how services interact with each other, yet it allows every interaction to be preceded by strong authentication of the parties, and mutual authentication as needed. To illustrate this point, the Athena model assumed that users could walk up to any

available workstation, authenticate, and the workstation would cache information ("tickets") that could be used to authenticate to all services they subsequently interacted with. When they finished using the workstation, they would log out locally, which would clear all established sessions and tickets, thereby allowing another user to take over use of the workstation without being able to access any information our services associated with the prior user. While the subsequent availability of inexpensive personal computers tended to obsolete this model, aspects of the original model are re-emerging today, as users tend to roam with their computers to different subnets and Internet Wi-FI hot spots.

- **Trust no party until authenticated:** Academic networks have typically employed a base policy of "allow everything except what is expressly disallowed," while non-academic networks start with the opposite policy of "disallow everything except what is expressly allowed." Despite its academic origins, Kerberos is actually closer to the non-academic policy regime in that it assumes that no party can be trusted until authenticated. The Kerberos developers realized they needed to support authentication, authorization and access control in a hostile environment that was based on the "allow everything" policy regime. They developed a compromise approach that allowed users and services to decide if the other party needed to be trusted and, if so, trust would only be extended to parties that could authenticate via the Kerberos services.
- **Operate in a hostile environment:** The Kerberos developers assumed that anyone could eavesdrop on network traffic, could claim to be any user, and could set up rogue servers capable of posing as any legitimate service, including the Kerberos services themselves. Encryption was used to prevent eavesdropping attacks, and session keys were introduced along with timestamps to prevent replay attacks. When users (or hosts/services) authenticate to the Kerberos authentication service, the authentication service in turn authenticates itself to the user (or host/service) by proving it knows the previously established shared secret. A by-product of these counter-measures is that Kerberos provides protection against man-in-the-middle attacks, which were generally regarded as infeasible at the time, and for more than a decade after Kerberos was initially deployed. Sadly, man-in-the-middle attacks are no longer mere conjecture, and are all too common in today's Internet web, which was *not* designed with a hostile environment in mind.

Kerberos provides a "model" for authentication and authorization

Kerberos is more than a set of protocols—it provides a *system model* for authentication and subsequent authorization in a peer-oriented, distributed computing environment. Important aspects of the model are:

1. It characterizes the environment and the interaction between agents in a way that is compatible with most distributed system approaches, making it easy to integrate Kerberos into applications and systems.
2. It concentrates the maintenance of secrets (i.e., stored passwords) in a small number of places (that can be hardened appropriately) rather than distributing them all over the system.
3. Kerberos separates **authentication from the services themselves**. The file server, for example, does not know, or ask for, the user's password. Instead, it delegates

that job to Kerberos, and relies on information provided by Kerberos to determine the authenticity of a request.

4. It does not require that all the communicating parties have prior relationships with each other, nor to have previously shared any authentication information with each other. Instead, all parties establish prior relationships with the Kerberos service and rely upon it to verify credentials and authorize sessions.

The full architecture and protocol is described elsewhere. This deliberately oversimplified description highlights the salient points.

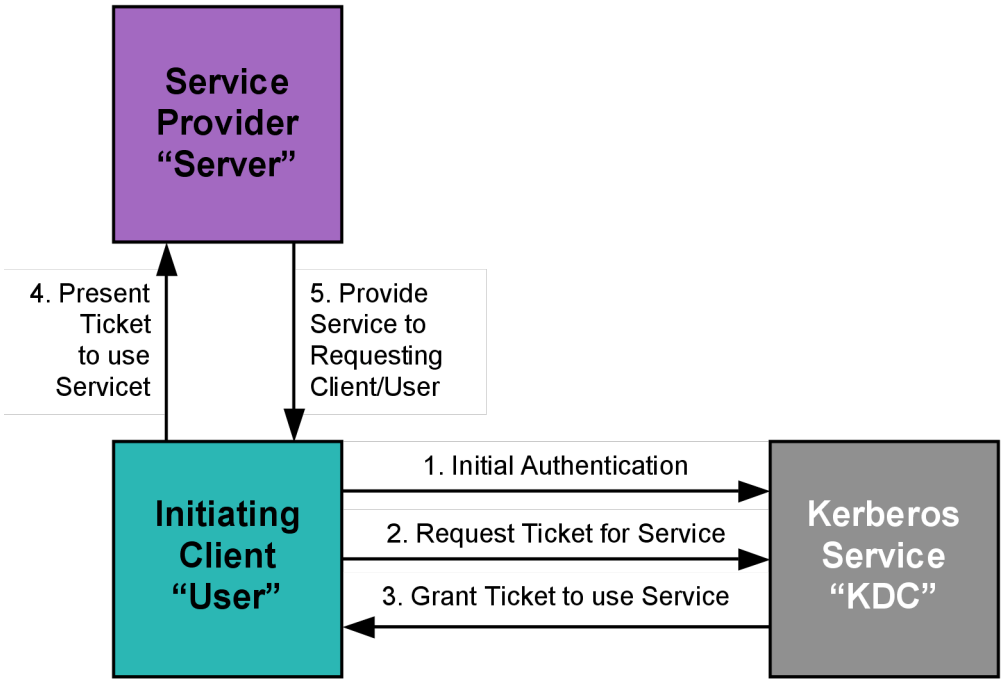

Figure 1—Simplified Kerberos System Model

The Kerberos architecture is designed around messages exchanged among three kinds of entities:

1. **Clients** wishing to use services,
2. **Servers** that provide services (note that clients and servers are collectively referred to as **principals**)
3. Servers that **manage the Kerberos protocol** itself. These servers are often called "KDCs" (Key Distribution Centers), and actually comprise several modular services.

Clients and servers authenticate each other by means of a protocol involving the exchange of **tickets**: cryptographically secure, time stamped data structures that contain authentication information and other particulars about a specific proposed interaction between a client and a server.

- When a new principal (client or server) is added to the system, or when credentials are changed, a **secret** (e.g., a password) is shared between the

principal and a Kerberos server. *This set-up stage is the only time that secrets need to be exchanged.*

- Subsequently, when a client wishes to log into the system, it obtains a ticket from a Kerberos server. *The client does not reveal its secret in the course of obtaining a ticket. Instead, the ticket is constructed such that only the possessor of the client secret can decrypt and use it.*
- When clients wish to access servers, they authenticate themselves to the servers by presenting tickets. Once again, no secrets are exchanged; a service ticket is encrypted and delivered in such a way that only the legitimate client could have obtained it, and only the legitimate server can decrypt it.
- *Note that there is no real-time communication between the server and the Kerberos infrastructure.*

The Kerberos model has stood up well over the past two decades. More importantly, this model has allowed Kerberos protocols and system specifications to evolve to address new requirements without having to change the basic architecture. The current Kerberos standards (referred to collectively as "version 5") represent the third generation of Kerberos evolution, but remain remarkably consistent with the original model. This has allowed significant changes to occur "under the hood," without changing the essential form and function of the Kerberos services. Today's Kerberos protocols are extensible, support multiple cryptographic algorithms, allow alternative authentication mechanisms, support scalable systems, and allow authentication and authorization to take place across organizational boundaries, but the essential model of interactions remains the same.

Kerberos is in place, mature, and stable.

The fact that the Kerberos system model has remained intact through a couple of decades of evolution speaks to the underlying strength of the approach. Of equal importance though, is the degree to which Kerberos has been standardized and integrated into the infrastructure of enterprise networks.

While some standards merely attempt to reflect the details of a particular implementation, Kerberos standards have actually been used to guide multiple independent implementations—both within the open source communities, and by mainstream vendors, such as Apple, Cisco, Google, Intel, Oracle, Microsoft, and Sun. Kerberos is somewhat unique as an open standard for which there are both open source and proprietary implementations. Furthermore, Kerberos evolution has been driven by both open source developers and mainstream vendors.

The IETF serves as the platform where Kerberos specifications were originally standardized and it continues to be the community where Kerberos standards have evolved. The Kerberos working group in the IETF (krb-wg) is actively working on further revisions and extensions of the Kerberos specifications—*i.e.,* the evolution continues.

Kerberos has also leveraged, and been leveraged by, other standards to facilitate integration with new technologies and applications. For example, Kerberos now

supports the AES encryption standard (FIPS 197) and is working toward support for improved hash algorithms as specified by FIPS 180. On the application front, the IETF's Generic Security Services Application Programming Interface (GSS-API) was adopted by Kerberos developers early on as a means for facilitating integration of Kerberos into applications. In the early 1990s, The Open Group adopted Kerberos as the primary authentication technology for use in their Distributed Computing Environment (DCE) family of specifications, which has subsequently influenced many other products.

Commercial products of all types and stripes continue to adopt Kerberos, and it is now built into many common product offerings. For example, Kerberos has been the default authentication technology within the Microsoft Windows family of client and server operating systems since the introduction of Windows 2000 Professional and Windows 2000 Server. Similarly, it is the default authentication product within Apple's Mac OS X family, and plays an expanded role in the latest 10.5 (a.k.a., "Leopard") release. Sun Microsystems has long used Kerberos within Solaris and other Sun software products, including Java, as a common authentication platform. Network technology vendors, such as Cisco and TeamF1, have also adopted Kerberos as a preferred authentication technology for system administration and other access control requirements. Intel even uses Kerberos to authenticate access to the remote system administration functions built into its latest family of microprocessors.

On the Unix/NetBSD/Linux front, Kerberos is thoroughly integrated, and delivered as a default authentication scheme for many popular distributions. It is also incorporated into many embedded flavors of Linux, and is therefore finding its way into network appliances of all types.

The real merit of a standard is not in the pedigree of the standards body that issues specifications, nor in the variety and breadth of adoption, but *in the ability for diverse systems to interoperate.* In this regard, Kerberos is remarkably successful. In many of today's enterprise networks, Kerberos is relied upon to provide a common authentication and authorization solution that allows end users and systems administrators the benefit of single sign-on to everything from database servers to email services to printers to network appliances. Furthermore, it doesn't matter what OS the user platform is running, or what OS a service runs on.

Kerberos—tempered in the flames of real-world challenges

Any security technology that purports to provide strong authentication and authorization in a distributed, Internet protocol environment is going to be severely tested in the real world. Certainly, Kerberos has weathered countless assaults over a long period in many very hostile environments. It has been subjected to all forms of attack, often by inside attackers with considerable knowledge of the way that Kerberos services have been configured. While there is no perfect security, the consensus is that Kerberos has stood up well to these challenges. If it hadn't, it would certainly have received a lot of negative publicity.

Furthermore, Kerberos has been deployed in many diverse environments. This has expanded the range of threats far beyond what might be seen in an academic or enterprise environment. For example, Kerberos has been used to protect the information

assets of government agencies, including agencies that face state-sponsored assaults on information. Kerberos also tends to be used to control administrative access to the high-priority targets found in every network, such as firewalls, routers, switches, servers, and DNS/directory services. This expanded threat context has resulted in better understanding about how Kerberos can deal with a broad spectrum of threats.

Over its lifetime, Kerberos has been extensively analyzed by security researchers and penetration testers who have sought to discover vulnerabilities that need to be corrected. For the most part, this analysis has led to problems being detected and corrected before viable exploits have been deployed in the wild. Again, no system can be immune to all forms of attack, but extensive analysis by a broad community that includes developers, users, system administrators, and security researchers does help mitigate risks. Put another way, Kerberos has been thoroughly scrutinized by friend and foe alike, and it has held up pretty well.

Why is Kerberos worth considering for today's systems?

Ease and quality of integration

As a mature, widely-adopted, open-standard security solution, Kerberos benefits from a broad community of developers that includes several open source initiatives that work in parallel with commercial developments from mainstream vendors. The Kerberos development community interacts via industry standards bodies, the MIT Kerberos Consortium, and the interplay of products in production environments. As vendors like Apple, Microsoft, and Sun have developed extensions to the Kerberos protocols, the trend has been for these extensions to become part of the "standard," and to be incorporated into other Kerberos implementations.

A broad developer community working with Kerberos has also resulted in integration with other technologies, such as directory services, policy management, multi-factor authentication, Java, and file/database services. In many cases, these efforts to integrate Kerberos with other technologies have resulted in new insights into how to improve and extend Kerberos. The result has been a beneficial cycle of development and integration that has yielded many synergistic results.

Kerberos can also be directly integrated into applications that operate in a distributed or transactional context where authentication and authorization are vital to securing operational environments. For example, the major database platforms come with built-in support for Kerberos. Given the difficulty in getting authentication/authorization "right," Kerberos is also a smart way for new applications, including custom in-house applications, to address this vital security requirement. In addition, applications based on Kerberos also benefit from centralized policy management and an established regime for system administration and audit.

The Generic Security Services API (GSS API) was introduced in the early 1990s, and Kerberos was one of the first security services to support this important industry standard for application integration. With all the major OS vendors also supporting

Kerberos natively, today's application developers have numerous options for integrating Kerberos support in ways that benefit users and operators.

Scale and federation

Of equal importance, Kerberos has proven ability to scale to the largest of organizations. It also supports separate administrative realms and cross-realm services that can be used to extend authentication and authorization across organization boundaries. This can help facilitate interactions between business partners and customers in today's increasingly federated models of interaction.

The original Kerberos model was designed to address the needs of an entire university campus with thousands of users and systems. This model also worked well for many enterprises and government agencies. However, as the scale of the Internet has grown along with many intranets—networks that operate within the boundaries of large, global organizations—the Kerberos model needed to be extended to allow sufficient scalability.

This was achieved by introducing Kerberos "realms" that operate autonomously, but extend authentication and authorization services from one realm to another. For example, a large corporation may deploy separate Kerberos realms at each of its major sites, but users would still be able to leverage cross-realm services to authenticate and gain access to services throughout the corporation.

This is analogous to the way that Microsoft Windows services are deployed as inter-connected "domains," and, in fact, a Windows domain is essentially a Kerberos realm as well. Similar strategies are also used for scaling directory services across a large organization, and a Windows domain will provide a Kerberos realm that uses the Windows directory service for administering user accounts. By taking an integrated services approach, modern distributed systems that leverage Kerberos can be deployed at global scale, yet allow users to work throughout the enterprise while system administrators can manage compliance with policies both locally and globally. Apple and Sun, along with open source distributions, employ similar strategies for integrating Kerberos realms with directory services to simplify overall system deployment and administration.

The growth of the Internet has also led to increasing needs for users to access services beyond a single enterprise's borders. eCommerce, outsourcing, and industry "extranets" have all contributed to the growing need for authentication and authorization to cross organizational boundaries. One approach for addressing this challenge that is gaining acceptance is the "federated model" for authentication, whereby a legal and technical framework is established that allows one organization to rely on another for authentication of parties, whether individual users or services.

The extended Kerberos model that includes realms with cross-realm authentication was also devised to address the need for cross-organization authentication, and is essentially an early implementation of the federated model. Given the maturity and broad adoption of Kerberos, it is already playing a significant role in real-world federated authentication/authorization systems.

Policy enforcement and audit-ability

Kerberos has long been used to provide central administration of policies regarding password use, registration of users and servers, and revocation of accounts. As externally mandated policies have emerged from many regulatory and pseudo-regulatory bodies, policy administration has become increasingly important. Kerberos policy administration tools have evolved over the years, and have been augmented by numerous developments. Kerberos administration is often integrated with other system services, such as directory services and user administration. This helps avoid duplication of effort in establishing and enforcing polices at the system level.

The flip side of policy administration is the ability to audit a system for policy compliance. The Kerberos model, used in conjunction with other tools, such as central system logging, provides in-depth support for audit of systems, along with mechanisms for taking corrective actions when policy violations are detected. There are even third-party developers (*e.g.,* Centrify) that provide tools for auditing Kerberos-based systems and integrated policy administration for systems that include Kerberos alongside other security measures.

Real-world interoperability

The old adage about standards—we have so many of them to choose from—becomes more true with each passing year. While standards are necessary, what really matters today is interoperability across products and services from many providers. For authentication and authorization services, interoperability is also needed across different classes of systems, ranging from network appliances to mainframes, from handhelds to workstations, or from a local application to "Software as a Service" ("SaaS"). Kerberos meets all of these interoperability challenges today, and is playing a larger role in new standards that attempt to knit together the ever more diverse threads of information technology.

Another aspect of interoperability that is sometimes overlooked is the need to coexist with alternative solutions. For good and valid reasons, and because it's just the way it is, there are, and will continue to be, multiple methodologies for addressing the problems of authentication and authorization. As a mature solution that has been in the IT "kit" for a long time, Kerberos has influenced other security solutions, and has been influenced by other security developments. This give and take has resulted in various strategies for coexistence of alternative security approaches within the same environment.

One illustration of this is in the OS offerings from major players such as Apple, Microsoft, and Sun. Each of these OS vendors has thoroughly integrated Kerberos into their software offerings, but they also support alternative approaches and provide application developers with the tools to support the authentication measures preferred by the users. In many cases, applications can be built to APIs (including the GSS API) that let the users determine how authentication and authorization should be performed for that application in their environment. The open source Linux and NetBSD OSs also take similar approaches to allowing coexistence.

Another aspect of coexistence is support for the various schemes for providing multi-factor authentication. Although Kerberos was originally deployed using a password-based approach to authentication, today, it supports many of the popular multi-factor schemes, including One-Time Password (OTP) tokens, smart cards, and various biometric scanners.

Choice of authentication mechanisms

In addition to support for digital certs and public key infrastructure (PKI) for authentication, many other authentication technologies have been adapted to Kerberos, including various multi-factor solutions. For example, smart cards and other devices that rely on certs tend to work "out of the box" with Kerberos, and are supported by major vendors.

Various One-Time Password (OTP) schemes have also been adapted to work with Kerberos, including popular "dongles" that display unique codes for each authentication operation. In addition, Kerberos has been integrated with common biometric scanners thereby providing further multi-factor authentication options.

Integration of new authentication technologies into Kerberos is a straightforward development exercise that takes advantage of the consistent Kerberos model to allow developers to improve adoption by leveraging of the broad Kerberos installed base. Since Kerberos insulates applications and services from the initial user authentication process, the benefits of new multi-factor authentication technologies can be easily extended to many existing applications or services.

Cost-effective in real-world deployments

As a technology base that comes integrated with many off-the-shelf platforms, Kerberos can be more cost-effective to utilize than other technologies that must be "bolted on." System administrators tend to deal with Kerberos as part of the whole systems they are already managing, and new deployments or upgrades include Kerberos as part of the overall package. At the same time, if new authentication technologies need to be introduced (e.g., in response to new regulatory requirements), Kerberos can provide the ready-made "sockets" into which these new technologies can be plugged in.

Kerberos support is usually part of the "package deal" as well. Existing support contracts and in-house staff support represent prior commitments of resources, so having an authentication technology that is included in such support arrangements helps avoid piling on new support costs.

Where Kerberos often pays dividends is in its ability to extend authentication and authorization across diverse platforms. To illustrate, if an organization deploys a primary platform that includes Kerberos integrated with a complementary set of directory services and system administration tools, then these services can be applied to other platforms, network equipment, facilities management, and interactions with customers or business partners. In mixed platform environments, this can represent significant reductions in support costs, while making it easier for users and system administrators to work across mixed platforms.

The cross realm facilities inherent in modern Kerberos also allow organizations to have different administrative realms, yet enable authentication and authorization to extend beyond these realms. For example, within a single organization, Engineering might use one platform, Manufacturing and Business Operations might use another platform, while Sales and Marketing use a third platform. Each of these platforms can support Kerberos services that are administered as distinct realms, yet standardized cross-realm services would allow interaction between users and services throughout the organization. The operational and opportunity cost savings from being able to support interactions across vertically organized business units can be significant.

Numerous support options from diverse providers

As an open standard that is supported by multiple open source development communities, the Kerberos technology is available to anyone to use, enhance, or integrate into their applications or products. Technically capable organizations can therefore fully support their own use of Kerberos since they have full access to the technology as well as access to the development communities. They can even develop their own extensions to Kerberos.

Organizations that integrate Kerberos into applications or products also have access to specialty support firms that can provide expertise in integration and testing. Furthermore, many other products and tools are already integrated with Kerberos, which can further reduce development efforts. When applications are delivered to users or customers, their support burdens will also be reduced since Kerberos is likely already in use in the target environment.

However, most organizations will not want to support Kerberos at the source code level. Typically, they will acquire support from their existing platform and application vendors, and may not even think of Kerberos as a line item in their support contracts. While there's not much that is free in this world, Kerberos support comes close, especially if the many benefits and other cost savings are factored in.