Recommended Practices for Deploying & Using Kerberos in Mixed Environments

Introduction

This document explores some of the many issues that emerge when deploying and using Kerberos in mixed environments, and presents guidance on addressing these issues and recommends practices for minimizing problems while optimizing the value of Kerberos as a consistent, cross-platform authentication solution. The target audience includes IT professionals with responsibility for infrastructure planning, strategy, architecture, and technology refresh. While the focus is on how Kerberos can be used effectively in mixed environments, it also addresses many issues that will be encountered by people who primarily deal with homogeneous environments.

Readers of this document may find it helpful to first become familiar with the topics covered in *The Role of Kerberos in Modern Information Systems (pdf)*, which provides useful background information on Kerberos and explains how Kerberos has been integrated with other vital services. Another companion document is *Best Practices for Integrating Kerberos into Your Application (pdf)* that addresses issues of interest to software developers while also providing insights into applications integration issues.

Information security is a complex and very challenging topic that requires a multi-disciplined approach. Too often, information technologists are put in the position of retrofitting security solutions to systems that are already in place. On the other hand, Kerberos is one of the core security technologies that is increasingly built into platforms and supported by many applications. This gives Kerberos strategic relevance in any environment, but an especially important role in mixed environments.

While many of the security issues encountered in modern information systems relate directly to how these systems have evolved, this document attempts to provide a forward-looking perspective. The objective is to support strategic planning and development of sound architectures for new deployments. At the same time, problems with current approaches and issues with technology integration are documented and guidance is offered on how to avoid problems or minimize impact.

Since this is a first attempt to address issues with using and integrating Kerberos in mixed environments, feedback and suggestions for improvements or additions are much appreciated. Please provide your feedback to the MIT Kerberos Consortium.

The Nature of Mixed Environments

The term, "mixed environments," can be difficult to adequately define, and means different things to different professionals. For many, it has come to mean some other systems environment coexisting with a Microsoft Windows environment. This reflects current market realities and the shape of today's IT industry. In this view, a mixed environment might be non-Windows workstations being integrated with a Windows-based systems environment, or new non-Windows applications servers being deployed alongside familiar Windows servers. Of course, the converse situation also exists, and there are those who have an Apple, or Linux, or Sun environment where Windows workstations or servers are being added to the mix. For still others, it means freedom to mix-and-match, or to choose best-of-breed.

In addition to mixed operating system platforms, mixed environments can use any combination of directory services: generic LDAP, Windows Active Directory, and others. They can also involve a variety of access control mechanisms, including PACs incorporated within Kerberos tickets, LDAP-based access control, and application-specific access control mechanisms for enterprise file systems such as AFS and other applications. A mixed environment can use Kerberos *alongside* other authentication technologies. Even if it is all-Kerberos, it can use any combination of Kerberos implementations: MIT Kerberos, Heimdal Kerberos, Windows Active Directory authentication, Apple OS X's implementation, and others.

Architecturally, a mixed environment can use any of several approaches with authentication services and directory services: Multiple authentication or directory schemes can exist in parallel without synchronization; or with all changes routed through one, or with peer-to-peer replication of updates.

A mixed environment can use any of several approaches to application services: Each application service might use only the authentication technology that is native to its OS platform, or applications might use authentication technologies supported on different OS platforms, potentially in a simultaneous manner.

Other major trends and technology initiatives are driving greater adoption of mixed environments, or at least greater heterogeneity of systems and components. Some of the trends and technology initiatives that directly influence both security and heterogeneity of systems include:

- **Virtualization:** The ability to create multiple virtual environments on a single host system has made it much easier to mix different operating systems and application services within a data center, and even on user workstations. It is now feasible to deploy hardware independently of platform software, and to change software deployments dynamically. Among other issues, authentication of services in a dynamic, virtualized environment becomes a whole new challenge.
- **Consolidation:** The drive to consolidate systems and equipment is driven by many concerns, including costs, but also energy consumption and staff productivity. One consequence of consolidation is that systems that were once deployed within organizational units (*e.g.*, manufacturing, engineering, sales & marketing, or administration) are now being consolidated in centralized data centers.

- **Mobility:** Desktop systems have been transitioning to laptops for some time, and now laptops are beginning to be replaced by handheld devices. Laptops and handhelds are increasingly connected via wireless networks on a continuous basis. This is changing some of the traditional workstation to server integration approaches and leads to new types of mixed environments. At the same time, considerable security challenges emerge when users are both remote and mobile.
- **SaaS:** Software as a Service is gaining both credibility and market traction. This has the potential to significantly alter the nature of current environments while introducing new challenges in authenticating users to services, and services to other services.
- **Appliances:** It is increasingly common for solutions to be packaged as turnkey hardware and software packages and introduced as "appliances" into existing environments. Often appliances provide security services, specialized network firewalls, or packaged applications, such as Voice over IP (VoIP) PBXs. They may present new integration challenges, and they often require new approaches to dealing with security concerns. For example, when the application is *voice,* how do users authenticate to PBX appliances? And what are the consequences if they don't?
- **NAC:** Network Access Control represents a new effort to push critical security functions to the very outer edges of today's networks, even though many of the techniques have been around for a while. In one sense, NAC represents a completely new environment with its own integration and security challenges. NAC could also introduce other new authentication technologies that either compete with, or complement, current authentication mechanisms.
- **Multi-Factor Authentication:** There are many sources of pressure (*e.g.,* regulations) to adopt multi-factor authentication in response to the growing threats to password-based authentication. This introduces new elements into current environments, and presents new challenges to addressing authentication in a consistent manner throughout an enterprise or even with external customers or partners. In some cases, new authentication mechanisms are being introduced to retrofit multi-factor authentication to current environments.

It is beyond the scope of this document to address all of the issues resulting from the trends listed above. However, most of the issues that are addressed below do relate to these trends in one way or another.

The patterns in which mixed environments emerge vary enormously between enterprises and—even for a specific enterprise—it can often be difficult to predict how environments might change in the future. In particular, some enterprises take a top-down approach to planning the evolution of their overall IT environments while others are driven by *ad hoc* projects that add new systems or technologies on an as-needed basis. In the IT business, *change* happens, whether it's been planned for or not.

Rather than attempt to map out all of the ways that mixed environments emerge and what new changes might result from key trends, this document brings to light specific issues that are often encountered when using Kerberos in mixed environments. Before proceeding, though, it may help to briefly describe three real-world situations where Kerberos is playing a role, and where enterprises have evolved unique approaches to dealing with the challenges of mixed environments. These real-world situations give a

flavor for how issues discussed later in the document fit together to form part of an enterprise's IT strategy.

Global Internet Service Provider
This enterprise maintains two distinct environments, a Unix/Linux environment using slightly modified KDCs based on one of the popular Kerberos open source distributions, and a Windows environment using Active Directory. Both environments are deployed globally, and many locations are served by both environments. At this time, there is no integration between the two environments for Kerberos authentication. However, some common applications can be accessed from either environment using Kerberos for authentication, and these applications support collaboration between users in the different environments.

Leading Research and Academic Institution
This institution also has a Unix/Linux/Apple environment alongside a Windows environment. However, Kerberos is integrated across the environments so that users from either environment can access services throughout the institution. Furthermore, users have accounts at the institution level that are maintained in both environments, including synchronization of passwords. A central registration system is used to establish and maintain accounts, and to coordinate synchronization between Kerberos and an LDAP directory service in the Unix environment and Active Directory in the Windows environment.

Major Full-Service Financial Institution
In this Financial Institution (FI), business units have considerable autonomy, and generally choose their own IT environments and security services, including mainframe computing platforms. In many cases, business units have deployed systems that extend beyond the FI to integrate with customers or other industry partners. In some cases, these external customers and partners have influenced the technology choices. While Kerberos is commonly used, it has often been included as part of a specialized systems integration project, and there are various flavors of Kerberos deployed with many disjoint realms. The challenge for security architects and officers is that the FI is regulated by various government and industry agencies, each with expectations that certain security practices are applied consistently throughout the FI, and even to external parties. Adding to these daunting challenges are management demands to better integrate information and services across *stovepiped* operations, while working within drastically reduced budgets.

No claim is made that any of these three examples is particularly representative; they are described only to illustrate how much variation in mixed environments there can be in practice, and how each mixed environment reflects fundamental characteristics of the enterprise where it is deployed.

The rest of this document brings to light issues that are commonly encountered in real world situations, and provides guidance and recommendations that reflect accumulated experience. While all of the issues evaluated in this document apply to the three examples above, how people in each enterprise might interpret the guidance and recommendations will depend entirely on the individual enterprise and its current circumstances.

Kerberos and Microsoft's Active Directory

While Kerberos was developed independently of any particular platform technology,[1] the adoption of Kerberos by Microsoft has hugely influenced the adoption of Kerberos. Microsoft not only integrated Kerberos into its Windows family or operating systems and services, it *tightly* integrated Kerberos into the Active Directory service framework. Since most other Kerberos implementations employ much looser integration with other services, it is important to understand the consequences of Microsoft's tight integration approach, as this directly contributes to many of the issues that are likely to be encountered when mixed environments include Windows and non-Windows systems.

However, it is *not* the purpose of this document to suggest that there are any fundamental advantages to either tight or loose integration styles. Strong arguments can be made in each direction, and it really comes down to the effectiveness of actual implementations and deployments. After all, the best of systems can be weakened or made difficult to maintain if poorly configured or managed. Instead, this document attempts to provide useful insights into how the different styles of integration impact choices that must be made when supporting mixed environments.

Understanding Kerberos Use in Windows Environments

Microsoft's Active Directory (AD) and other Windows software products leverage Kerberos extensively as a core security technology. The *Role of Kerberos* document provides relevant background on how Kerberos fits within a Windows Active Directory environment. What is important to understand when dealing with mixed environments that include Windows and non-Windows platforms are the similarities and differences in the way that Kerberos is used in these different environments.

Similarities in Kerberos Support

The Windows product line fully supports Kerberos technology, primarily through the Active Directory service suite, but also in APIs and application frameworks. Kerberos is the preferred authentication technology for Windows users and services, although Microsoft continues to support other authentication schemes for backwards compatibility and interoperability. The following points summarize the key similarities between Microsoft's Kerberos support, and the support offered in other environments:

- Kerberos protocols and services are fully supported in an interoperable manner. Standard KDC functions are supported by Windows Domain Controllers, and can be used by non-Windows workstations and services.
- Applications based on GSS-API Kerberos wire level protocols can interoperate between environments. Note that Windows applications use SSPI instead of GSS-API, but the protocols are consistent.

[1] Though it should be acknowledged that Kerberos was historically developed first for various flavors of Unix, and other technological initiatives, such as OSF's DCE, also influenced the evolution and adoption of Kerberos.

- SPNEGO can be used to negotiate authentication mechanisms used between clients and services in different environments.
- Naming of principals can be consistent between AD and non-AD environments, though it is possible for AD principal names to be defined in ways that might complicate interoperability.
- Referrals were introduced by Microsoft, and are now supported by other Kerberos implementations (*e.g.,* MIT's Kerberos).
- Cross-realm relationships (a.k.a., realm trusts) can be established between Active Directory realms and other Kerberos realms.
- PKINIT is supported for initial client authentication using public key cryptography and certificates. This enables use of smart cards and other cryptographic tokens when authenticating via Kerberos, and it is possible (though not always easy) to use the same token devices in different environments with different Kerberos implementations.

For a thorough explanation of Microsoft's Kerberos implementations, refer to MS-KILE. Other useful references include Microsoft's Security Overview MS-SECO and Protocol Overview MS-PROTO documents.

Differences in Use of Kerberos

The differences between Kerberos in a Windows environment versus other environments largely come down to differences in how Kerberos is used and how it is integrated into the platforms. For example, Microsoft provides a proprietary API—the Security Support Provider Interface (SSPI)—for applications that use Kerberos and other security services on a Windows platform. However, applications based on SSPI can interoperate with applications on other platforms that use the GSS-API Kerberos mechanisms. It is also possible to use applications built with the GSS-API on a Windows platform by using third party software.

Another example of differences in usage is that Microsoft employs Kerberos to deliver authorization data about clients to services in the form of the Privilege Attribute Certificate, or PAC (ref. MS-PAC). Since the PAC is included in tickets issued to services, this can have some other side effects, such as larger ticket sizes, which may lead to the requirement to use TCP instead of UDP when communicating with KDCs. However, while these differences can be very significant for applications, Kerberos interoperability issues with use of PACs are largely resolved in current implementations.

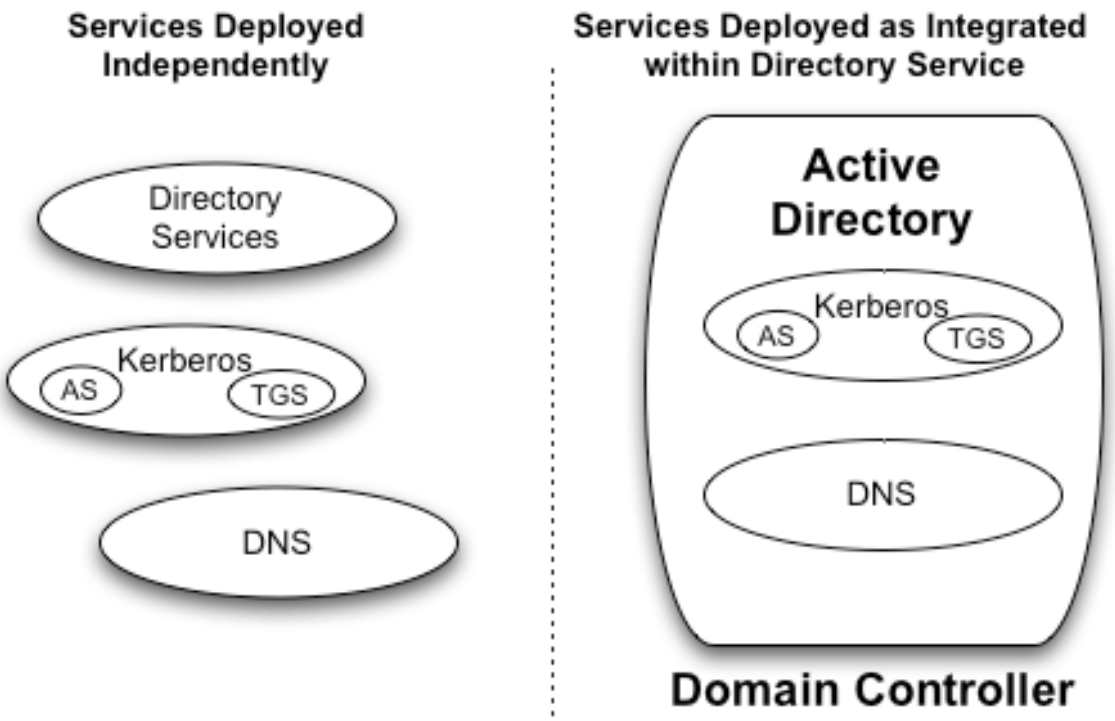

Figure 1—Contrasting the loose versus tight integration approaches

Microsoft's overall approach to using Kerberos has been to integrate Kerberos tightly with Active Directory. It is helpful to view Microsoft's Kerberos implementation as essentially Active Directory with integrated support for the Kerberos protocols. In other words, the KDC Authentication Services (AS) and Ticket Granting Services (TGS) are protocol interfaces into Active Directory. This is illustrated in Figure 1 above. DNS and other services, including LDAP directory interfaces, are similarly integrated with Active Directory,[2] and are all integral services of a Domain Controller. Active Directory is much more than a directory service.

By contrast, other Kerberos deployments tend to treat Kerberos AS & TGS services as independent of other services. While it is certainly possible to host Kerberos on the same servers as DNS or Directory Services, this is not required, and there is no direct, under-the-covers integration between Kerberos and other services. If a Kerberos KDC interacts with DNS or a directory service in a non-AD context, it does so in the same way any other application interacts with these services, though there may well be other applications that provide tighter integration of information across these services.

One consequence of the Active Directory approach is that each Domain Controller (DC) provides directory, Kerberos and DNS services in an integrated manner. From a Kerberos point of view, the Domain Controller is the KDC. Normally, users,

[2] The actual means used for integrating these services with Active Directory varies from service to service, but conceptually, it is easiest to think of these as being extensions of AD. For example, DNS is implemented as a distinct service on Domain Controllers, but it is largely administered from AD, and the DNS database is synchronized with information in AD. At the same time, Microsoft's DNS service interacts with other DNS services and shares information with the overall DNS system.

workstations, and services associated with a Domain will all use the Domain Controller for authentication and logging in to services.

Another, more profound, consequence of this approach is that every Windows Domain is also a distinct Kerberos realm. In fact, Domains and realms exist in a one-to-one relationship, though a Domain is more than just a realm. In this document, the term "AD realm" is used to refer to the realm associated with a single Domain. Typically, an enterprise will deploy multiple Domains to distribute services geographically or organizationally. Consequently, such multi-Domain systems will have an equivalent number of AD realms. Since Active Directory handles most of the configuration and operational issues associated with Domains and realms, the existence of multiple realms is largely transparent.

In traditional Kerberos deployments, there tend to be far fewer realms, and many enterprises will only need a single realm. This is because the Kerberos system scales well, and KDCs can be distributed throughout a network to provide adequate availability. In fact, the distribution model for KDCs might be similar to a corresponding distribution model for Windows Domains, particularly if the models are based on geographical considerations. This is illustrated in Figure 2.

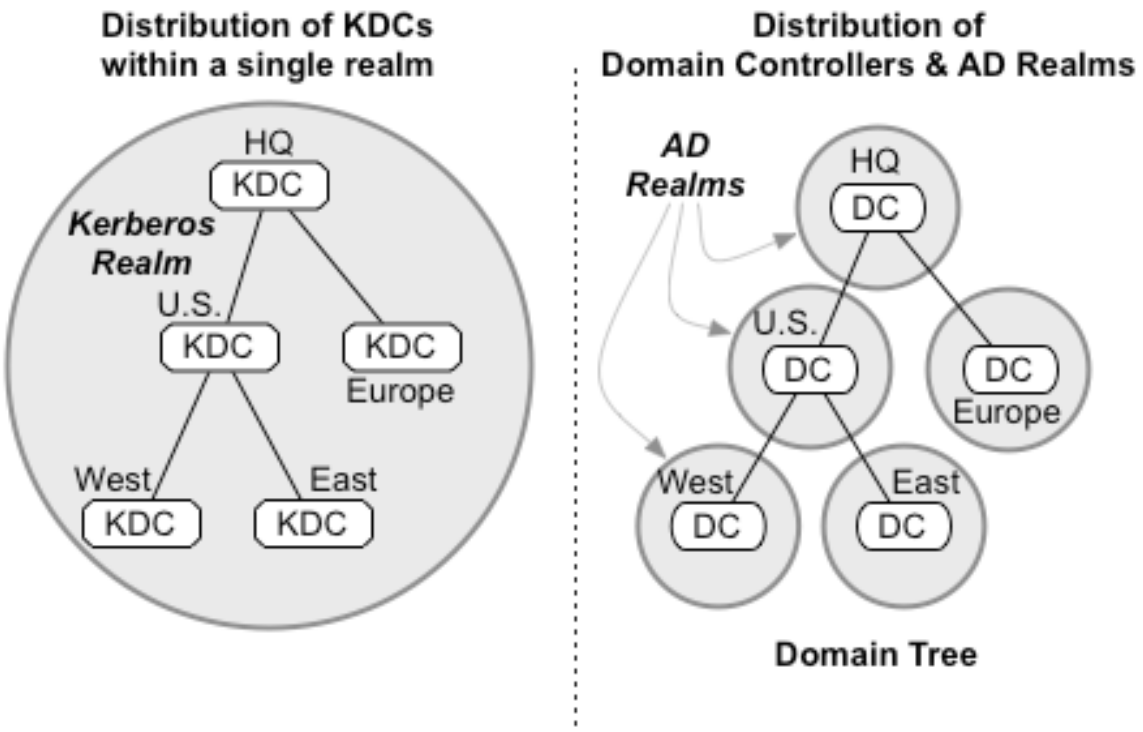

Figure 2—A single Kerberos realm with distributed KDCs versus a Windows environment with distributed Domains and AD realms.

If an enterprise operates a mixed environment where one environment is Windows-based, then the realm deployment strategies are likely to be quite different between the Windows and non-Windows environments. This can result in co-existing environments with significantly different realm topologies. For example, an enterprise might deploy Windows Domains in the manner shown in Figure 2 alongside another environment

with a single realm topology. This tends to complicate strategies for implementing cross-realm relationships between the environments, but it is certainly feasible to establish such relationships.

Microsoft refers to the relationships established between Domains in a tree as *trusts.* While these trusts employ Kerberos cross-realm relationships, trusts are also secure associations between different nodes in the Active Directory tree. Domains also represent different administrative regions. Consequently, trusts are used to distribute directory information, including all account information, and to provide the organizational structure for administering an entire enterprise intranet. The establishment of trust connections between Domains in an Active Directory tree or forest is largely automated once Active Directory and Domains are configured. Considerable documentation exists on Active Directory trust relationships, including the Microsoft TechNet article: *Domain and Forest Trusts Technical Reference*.

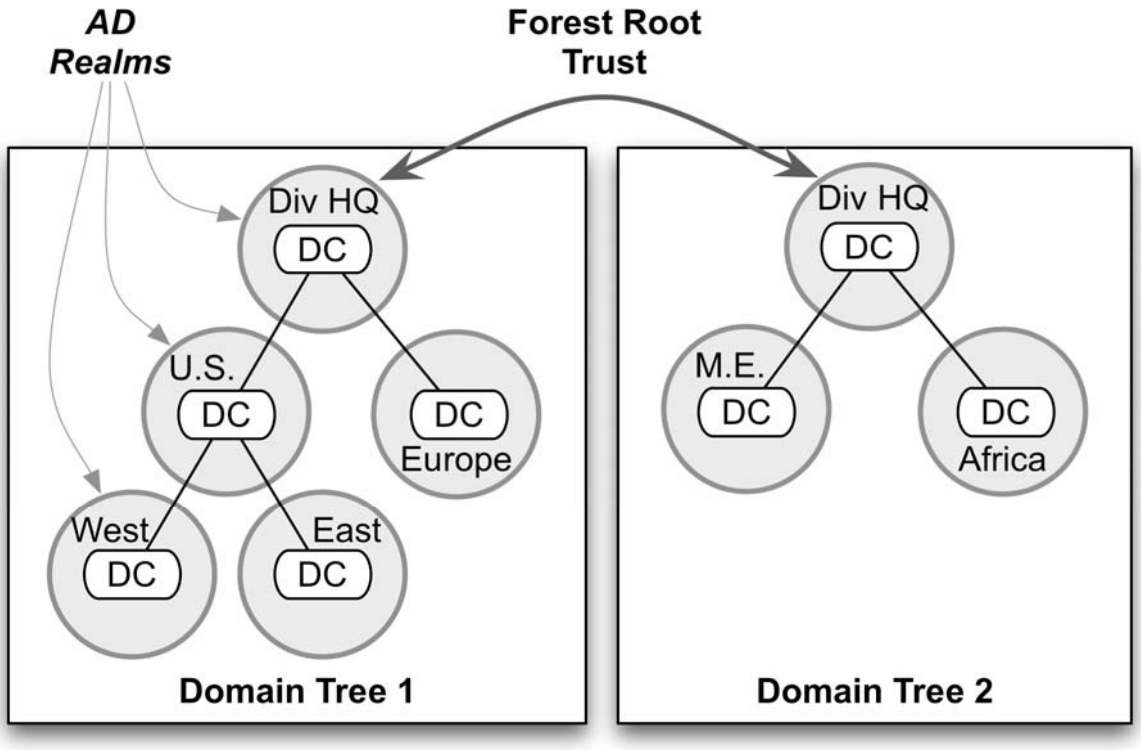

Figure 3—Active Directory forest with two Domain trees

Active Directory uses trusts to allow multiple Domain trees to be combined into a new directory structure called a "forest," as illustrated in Figure 3. This is useful in situations where an enterprise includes two or more autonomous divisions, or where the organizational structure of an enterprise is more easily managed if split across multiple Domain trees. Forests are analogous to transitive Kerberos cross-realm relationships (refer to *Role of Kerberos* section on multiple Kerberos realms). However, this analogy should not be taken too far, since a *forest* is a *directory construct* that happens to use Kerberos for authentication and security services in a distributed context. Furthermore,

Active Directory handles configuration of trust paths—the equivalent of *capaths* in Kerberos that specify allowed cross-realm transitive relationship paths.

Figure 4 shows how the same enterprise as illustrated in Figure 3 might deploy Kerberos realms for a non-Windows environment. If there are only two realms, then a simple, direct cross-realm relationship could be used, which would be conceptually similar to the *forest trust* used in Active Directory. While Figure 4 shows networks of KDCs in each realm that mimic the topology of the Domain trees in Figure 3, this is only illustrative. It really doesn't matter how the KDCs are deployed in a network, as all KDCs in the same realm use the same database, and they are essentially interchangeable.[3] This is in sharp contrast with the Active Directory forest, where the distribution of Domain Controllers in a network determines the structure of the directory itself, as well as where AD realms are located. Furthermore, while a Domain Controller performs the function of a KDC, this functionality is integral to the DC, and cannot be redistributed in the same way as non-Windows KDCs. The functions are the same, but the purposes are different.

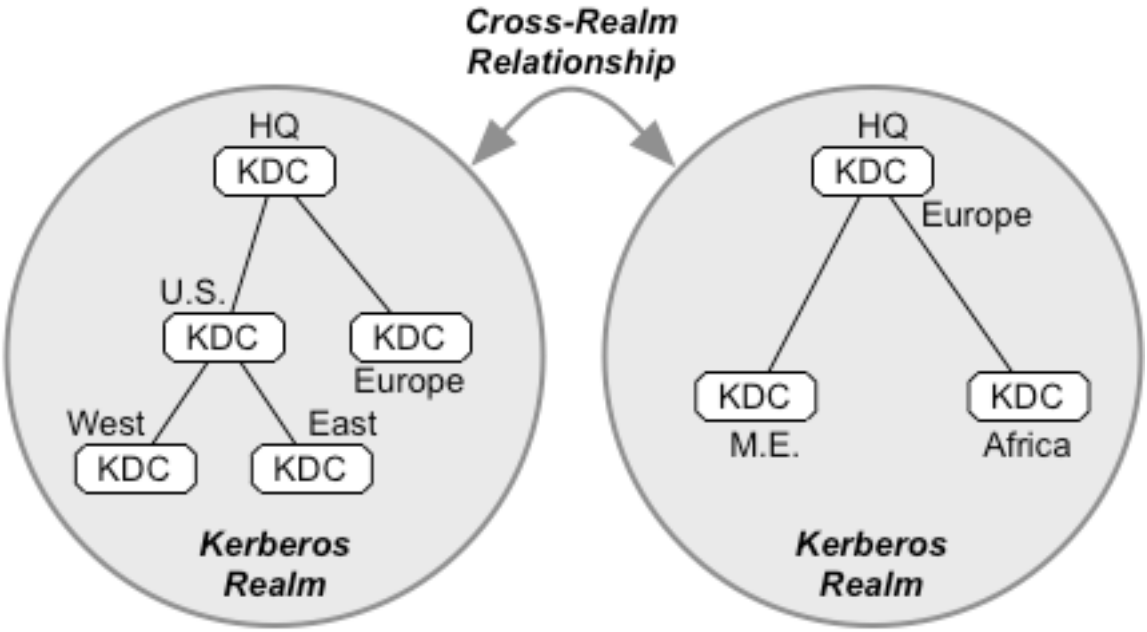

Figure 4—Two Kerberos Realms in the same enterprise

If a mixed environment includes other directory services in addition to Active Directory, then it is likely that the enterprise will want to also integrate directories across environments. While this is not directly a Kerberos issue, the fact that Kerberos is tightly integrated with Active Directory does mean that coordinated planning for Kerberos and directory services will be necessary to at least some degree. Kerberos may also play a role in securely connecting the directory services between distinct

[3] However, it does matter that users and services utilize the KDC that is *closest* to them from a performance and availability perspective. Active Directory does provide mechanisms for users and services to find the closest Domain Controller.

environments, and Kerberos is a preferred authentication technology for accessing LDAP directory services in *any and all* environments.

The essential differences between Kerberos deployed in Windows and non-Windows environments can be summarized as follows:

Windows Environment	Non-Windows Environment
SSPI is the API for Windows applications that use Kerberos authentication. It is compatible at the wire level with GSS-API Kerberos. GSS-API can be installed on Windows for applications that use it.	GSS-API is the primary API for applications that use Kerberos.
Active Directory tightly integrates Kerberos along with other services, notably DNS. Kerberos is essentially an extension of Active Directory, and its database is provided by the directory.	Kerberos is deployed as an independent service that coexists with other services such as DNS and directories in a loosely-coupled manner.
Kerberos plays a significant role in providing authorization within the Active Directory context.	Kerberos primarily plays an indirect role in authorization.
Privilege Attribute Certificates (PACs) are inserted into Kerberos tickets issued by Domain Controllers. PACs are authorization data delivered securely via tickets.	The option to include authorization data is available in any Kerberos implementation, but it is not widely used in non-Windows environments.
Domains and realms exist in a one-to-one relationship, with the implication that there are potentially many realms within an enterprise.	Realms are deployed only as needed to satisfy requirements for autonomous administrative regions—generally only a few realms per enterprise.
Active Directory establishes *trusts* between Domains to extend the directory service and provide distribution of policy and account information. Trusts typically utilize Kerberos cross-realm relationships.	Kerberos realms are integrated by establishing cross-realm relationships that allow for secure granting of tickets from one realm to another.
Since Kerberos is integrated with Active Directory, there can be greater flexibility in naming principals, since the directory can map names to Domains/realms.	Principal names must conform to Kerberos specifications. DNS naming conventions must be followed, since DNS lookups may be used to find services and realms.

Active Directory Trusts and Cross-Realm Relationships

As noted above, Active Directory establishes trusts between Domains in a forest. There are many ways to configure trusts in order to satisfy diverse security requirements in complex enterprise environments. However, one type of trust, called a *realm trust,* is essentially equivalent to a Kerberos cross-realm relationship, and may be used between Active Directory and other environments. Some characteristics of realm trusts are:

- Realm trusts can be configured as one-way or two-way trusts, which corresponds to the same concept in cross-realm relationships. A one-way trust would allow users/clients from one realm to authenticate to services in the other realm, but not vice versa. Realm trusts default to one-way, which is the opposite for other trusts, which default as two-way.
- While the default trust mode is non-transitive, a realm trust can be configured as transitive. Note that other trusts default as transitive, since AD can manage trust pathways that follow Domain tree structures.
- When a non-Windows Kerberos realm has a realm trust (one-way) with an AD realm (*i.e.,* the AD realm is *trusted* by the Kerberos realm), then any account in the AD realm (Domain) can authenticate to services in the Kerberos realm.
- When an AD realm has a realm trust with a non-Windows Kerberos realm (*i.e.,* the Kerberos realm is trusted by the AD realm), then account mappings can be assigned to determine which principals in the non-Windows realm can access services in the AD realm. Account mappings (sometimes called *proxy accounts)* can be one-to-one, or many-to-one. For example, a single AD *proxy* account can be configured with many principals in the non-Windows realm mapped to that account. Account mapping allows AD authorizations to be applied to non-Windows Kerberos principals, including insertion of PACs into tickets.

Use of non-transitive realm trusts means that only a single AD realm (Domain) can use the trust for cross-realm authentications. However, since AD realms tend to be smaller in scope than non-Windows Kerberos realms, this presents some challenges. To illustrate, consider Figure 2, where the Kerberos realm on the left could have a cross-realm (realm trust) relationship with any one of the AD realms on the right. Since the Kerberos realm on the left represents the entire enterprise, while AD realms on the right represent only a subset of the enterprise, there is a discrepancy in realm scope between these two environments.

Account mapping can be used to resolve this discrepancy. Each AD realm (Domain) that has a realm trust with a non-Windows realm would only map an appropriate subset of principals from the Kerberos realm into the local AD realm. For example, the U.S. west coast office in Figure 2 could establish a non-transitive trust realm between the Domain and the enterprise-wide Kerberos realm, but set up account mappings in the Domain for only west coast staff members. Other access controls could be used to restrict west coast staff members who authenticate from the Windows Domain into the Kerberos realm to be able to only access services in the west coast office. Of course, the actual usage patterns are typically much more complex than just restricting uses to a single physical site, so this example is illustrative, but probably not realistic.

Using non-transitive realm trusts, it is possible to set up relationships between Kerberos realms and AD realms as needed. This might be an effective strategy if there are only one or two Kerberos realms that need to interact with AD realms. It would become onerous to set up and maintain if there are more than a couple of non-Windows Kerberos realms, or if nearly every AD realm needs cross-realm support with the non-Windows environment.

Transitive realm trusts can be used in situations where many AD realms, or entire trees of AD realms, need to interoperate with non-Windows Kerberos realms. A transitive realm trust set up between a Kerberos realm and a Domain tree root could support access between the entire Windows Domain tree and non-Windows environments. However, this requires that authentication paths be defined from the Kerberos realm through the AD tree structure. In the AD environment, authentication paths (trust paths) are managed by AD itself, and automatically configured. On the non-Windows side, however, path information will have to be extracted and configured on clients and KDCs. This can be automated, and some other directory service could be used to provide similar capabilities to what AD provides in the Windows environment.

A hierarchical transitive approach (see *Role of Kerberos*) can be used to simplify configuration challenges when using transitive realm trusts between Windows and non-Windows environments. A Domain root could be used as the central/core realm for a hierarchical set of cross-realm relationships. However, such hierarchical transitive schemes require that DNS names be unique in each environment. This means that separate DNS name hierarchies must be used in the Windows and non-Windows environments, but with a common suffix.

However, a Domain forest does *not* have a true root, and often does not have a common DNS name suffix between the separate Domain trees in the forest. This can significantly complicate use of hierarchical transitive relationships and realm trusts to establish a cohesive Kerberos system operating across different environments. If it is not possible to establish a common DNS root name that can serve as the overall root for a hierarchically organized system, then other techniques will be needed integrate the environments. Probably the most promising approach would be to use a directory service in the non-Windows environments to automate the configuration of transitive relationship paths in much the same way that Active Directory manages trust paths.

DNS and Kerberos in Mixed Environments

As discussed in the *Role of Kerberos*, DNS services are used by Kerberos. First, DNS SRV records are used to locate KDCs. Second, Kerberos needs to be able to map a given DNS host name to a corresponding realm name. Finally, many non-Microsoft implementations of Kerberos use DNS to help resolve host name aliases or “CNAME records” to the resulting host. Proper configuration of DNS in a mixed environment can significantly simplify the configuration of workstations and servers.

The first concern is often to select which DNS server to use. One choice could be the DNS server provided by Active Directory. It is also possible to use some other DNS server, which may, or may not, be Windows based. Windows environments can

leverage the ability to dynamically update hosts names as new hosts are created (i.e., *join* the Domain) and as hosts are issued addresses by the DHCP integrated in the Domain Controller. However, most other modern DNS servers also support this capability. It is important to have *one DNS infrastructure* rather than a parallel but separate DNS infrastructure for Windows and non-Windows machines—*i.e.,* the domain names and addresses seen by Windows machines need to be the same as those seen by non-Windows machines. In this case, *one infrastructure* does not mean homogeneous DNS servers, it simply means that whatever DNS servers are deployed interoperate to provide a common namespace across all environments. It is sometimes easier to standardize on one DNS server technology for an administrative regime. For instance, one zone (*e.g.,* sales.some-corp.com) can use a Windows DNS server while another (*e.g.,* hr.example.com) uses some other DNS server technology.

Enterprises typically need some software to feed hosts into DNS as they are created and to keep DNS synchronized with DHCP. Multiple options are available and from a Kerberos standpoint it does not matter which option is chosen. What does matter is that DNS names are consistently assigned, and that DNS names can be queried from anywhere within the enterprise.

Locating KDCs

DNS should include *SRV records* to locate KDCs and password changing services. This eliminates the need to configure KDC information on each client.

Host to Realm Mappings

By far the easiest configuration for Kerberos is when each host's FQDN has a suffix that matches that host's realm name. For example, configuration is easiest if "fileserver.east.some-corp.com" is in the Kerberos realm "EAST.SOME-CORP.COM." Without explicit configuration, non-Windows implementations will try using the suffix of a host's FQDN to find the realm for that host. By implication, Windows Domains should adhere to similar naming conventions.

Windows KDCs (a.k.a., Domain Controllers or DCs) will provide referral information (see the *Role of Kerberos* for details) to tell clients what realm a host is in. That means in a pure Windows infrastructure, DNS need not match the breakdown of hosts into realms. In a mixed environment, Windows clients can still take advantage of this information when talking to hosts registered in an AD realm provided that either the user's account or the workstation's account is registered in an AD realm. MIT Kerberos introduced support for taking advantage of this information from Windows KDCs in version 1.6. So, enterprises that use modern MIT Kerberos may be able to take advantage of the Active Directory mechanisms for mapping hosts registered with an AD realm to other realms, even for non-Windows clients.

If the default suffix rule is inappropriate and referrals are insufficient, then clients need to be configured. Clients typically support configurations that place all hosts with a particular DNS suffix into a configured realm. So, deployments where all, or almost all, of the hosts with a given DNS suffix live in the same realm will be much easier to configure.

Another option is available: "TXT records" can be entered into DNS to map the names of hosts to a corresponding Kerberos realm. This option is reasonably usable but presents significant security concerns. Reliance on TXT records cannot be recommended, since an attacker can direct a client to the wrong realm by providing false responses to DNS queries. If this option is used, administrators must carefully limit what service principals are registered in any realm in the enterprise to limit the scope of these attacks.

Reverse Resolution and DNS Aliases

As mentioned, Kerberos implementations may, but do not always, attempt to resolve CNAME records or reverse-resolve the resulting IP address. Consequently, if "PTR records" are included in the reverse resolution zone, they *must* point to the name of the host that is registered with Kerberos. If CNAME records are used, then both the CNAME and the name it points to should be registered with Kerberos and the service should accept authentication for either name.

Kerberized Applications in Mixed Environments

Deploying Kerberos-aware (or *Kerberized*) applications in mixed environments[4] means ensuring that the application can interact with the environment's Kerberos implementation. In practice, it also requires that the application interact with the environment's directory service.

Different applications are written to support Kerberos in one or more of several different ways:

- An application can natively implement the Kerberos protocol messages
- An application can be written to Microsoft's SSPI API
- An application can be written to the GSS-API

The constellation of applications to be supported, and the way in which those applications support Kerberos, has a strong impact on the integration strategy to be used, as discussed in the following section on integrating workstations.

Workstations and Kerberos

The issues around supporting Kerberos on workstations in mixed environments fall into three basic areas:

- Making it possible for users to use their Kerberos credentials to log into the workstation (*i.e.*, to authenticate to the workstation's operating system)
- Enabling the workstation to cache the user's credentials and participate in the environment's single-sign-on scheme

[4] This section is about deploying existing applications in a mixed environment. For advice on writing applications to be deployed with Kerberos, see the companion document, *Best Practices for Integrating Kerberos into Your Application.*

- Enabling application software that runs on the workstations to take advantage of Kerberos

The specific approach taken depends upon whether or not the workstation is a Windows workstation, and whether or not the overarching Kerberos environment is Windows Active Directory or a generic Kerberos implementation

Integrating Windows WSs into Mixed Environments

There are two approaches for accessing Kerberos resources from a Windows workstation. The first is to use the native support for Kerberos built into Windows. Typically, this approach is the only approach used by Windows workstations in a pure Windows environment. The second approach is to install a third-party Kerberos implementation such as the MIT Kerberos Consortium's *Kerberos for Windows (KFW)* product.

A number of factors influence which approach should be used. A significant factor is the range of applications that the workstation will be expected to support, and the way in which those applications support Kerberos. A given Windows application might support Kerberos:

- Not at all, or
- By implementing the Kerberos protocols by itself, or
- By use of the "Security Support Provider Interface" (SSPI) API, or
- By use of the "Generic Security Service Application Programming Interface" (GSS-API)[5]

An application that implements the Kerberos protocols on its own can, in theory, interoperate with any Kerberos KDC. On the other hand, the protocols do not specify how such an application interoperates with the local workstation's Kerberos implementation (*e.g.*, how the application knows where the user's tickets are cached, or even how it knows where the environment's KDC can be found). Those wishing to support such application must therefore understand how the application is expecting to interact with the local workstation's Kerberos implementation, and choose an integration strategy that supports it. Fortunately, such applications are rare.

An application that supports only SSPI (for example, Microsoft Outlook, Internet Explorer, and Windows File Sharing) *must* use the native Windows Kerberos implementation.

An application that supports only the GSS-API *must* use Kerberos for Windows.

An application that supports both SSPI and GSS-API (as do many third party applications that support Kerberos) may use *either* the native implementation or Kerberos for Windows.

[5] These APIs are discussed in further detail in *The Role of Kerberos.* Advice for those developing applications can be found in *Best Practices for Integrating Kerberos into Your Application.*

Unfortunately, this means that the questions of which applications to support (or which may be required in the future) and which integration strategy to adopt are tightly coupled. The different strategies provide a different set of supported environments, and the layout of the mixed environment affects which applications can be used.

The native Windows Kerberos implementation should be used when it will work in the environment. However, it has three key limitations:

- The native Windows Kerberos implementation works well when Domain or other Kerberos accounts are used to log into the workstation. It tends not to provide an acceptable experience if a local account is used to log into the computer and Kerberos is used to access some resources, especially if these are non-Windows resources.
- The native Kerberos implementation works best when all resources can be accessed using the login account. Resources in realms other than the user's home realm can be accessed through cross-realm relationships. However, it is difficult to access resources that will require authentication as a different principal. As an example, the native implementation works poorly when Windows and non-Windows environments are completely separate without cross-realm relationships (or trusts). Facilities for using a separate account with some servers are provided (See Credentials Management below) but Kerberos for Windows often provides a more usable experience.
- The native implementation only works with applications built using SSPI. Applications that require the GSS-API will require Kerberos for Windows.

Kerberos for Windows works well in environments where these limitations are a constraint. There are two major disadvantages to Kerberos for Windows. First, it must be installed, as it does not come with the operating system. Secondly, it does not support applications written for SSPI. KFW is most useful in the following situations:

- A workstation that is not part of a Windows AD Domain needs to access Kerberos resources.
- A workstation needs to access Kerberos resources that cannot be accessed from the login account, either because a different principal is needed or because no cross-realm relationship (trust) is in place.
- It is necessary to run applications on the workstation that were built using GSS-API, but not SSPI. This is most often the case when applications are ported to Windows from other platforms.

The remainder of this section discusses details of how to use each integration approach to access resources in a mixed environment.

Realm Trusts and Domain Configuration

The easiest way to access resources in a mixed environment with the native implementation is to configure appropriate trusts and group policy with the workstation's domain. Microsoft provides facilities to establish a trust (similar to a cross-realm relationship) between a Windows domain and a non-Active-Directory realm. If such a trust is established and configuration of realm flags is pushed out to

members of the domain, then workstations in the domain can access services in the non-AD realm.

The *ksetup* Utility and Local WS Configuration

Another approach is to configure the local workstation directly rather than using domain resources. Windows provides the *ksetup* utility that can configure realm information including the location of KDCs and realm flags. The ksetup utility can also join a workstation to a non-AD realm so that realm can be used for login.

Ksetup is definitely the right tool to use when workstations will use Kerberos for login from some realm but will *not* be joined to a Domain. However, configuring a realm on a Windows workstation that *is* joined to a Domain without also configuring a trust from the Domain to the non-AD realm has limited value.

Kerberos for Windows

Kerberos for Windows (KFW) can be installed on a workstation. When installing KFW, administrators need to decide how much separation is appropriate between KFW and the Windows native installation. Several options are available:

- KFW can attempt to obtain credentials using the password supplied at login. This mode is useful if a workstation is not joined to any Domain or realm. It is also useful if a workstation is joined to a Domain, but non-AD Kerberos realms are separate. In this case, KFW can be configured to obtain credentials in the non-AD infrastructure while the native Kerberos obtains credentials for the AD infrastructure. Of course, using the login password only works if password synchronization is used between the two environments.
- KFW can be configured to be completely separate from the native Kerberos and to obtain no credentials on login. As credentials are needed, KFW will prompt the user to obtain credentials. This works best if a workstation needs to use multiple accounts for different resources. In this case, passwords do not have to be synchronized between different environments. For example, a user could use one account in the Active Directory environment, and a separate account in some other environment.
- KFW can import native credentials if they are available. This permits KFW to default to using the same account as the login user. However this default can be overridden if a different account is needed to access some resources.
- KFW can be locked to the native Windows credentials. This mode is most useful when KFW is being used because GSS-API applications are needed in an environment with full cross-realm relationships or trusts.

On Windows Vista, recent versions of KFW can establish new Windows login credentials. This allows KFW to obtain credentials for a machine not joined to a domain and for these credentials to be used with SSPI applications.

Credentials Management

Windows provides a Credentials Management interface to applications. Applications can use this interface to obtain new Domain credentials to access some resource using a

different principal name than the one used for login. If applications support this functionality, it may be a useful option.

A facility is also provided to store network passwords for accessing services. If a password is stored, then Kerberos is one of the authentication mechanisms that will be tried in order to access the services. Wild-card service names are possible. This facility may be problematic from a security policy standpoint because it involves storing Kerberos passwords for accessing non-AD realms on the local workstation.

Finding Kerberos Resources

The windows workstation's Kerberos implementation must be able to identify and contact the appropriate KDC. In the case of the native Windows Kerberos implementation talking to a Domain Controller, this is handled automatically through DNS as described above in the section entitled "DNS and Kerberos in Mixed Environments." In an environment where the KDCs are other than Windows AD controllers, it is necessary to manually create the appropriate SRV records in the DNS system. For environments that do not use the DNS system to locate Kerberos KDCs, appropriate configuration parameters must be set on each workstation, which becomes a significant system management challenge in large and/or dynamic environments.

Integrating Non-Windows WSs into Windows Environments

Accessing Kerberos resources from a non-Windows workstation is straightforward: the non-windows implementation is expecting to talk to a KDC, and either a Windows Domain Controller or a generic Kerberos KDC will work. There are some issues (for example, Active Directory environments typically include a much larger number of realms than a generic Kerberos environment), but the basic mechanisms for configuring and accessing Kerberos resources are the same irrespective of whether KDC is an AD Domain Controller or some other KDC implementation.

Initial Kerberos credentials are obtained at login via PAM modules on Linux workstations, and similar modules on other flavors of Unix. If a user logged in using one set of credentials needs to use a different set to access some services (*i.e.*, if accessing a service in another realm without cross-realm arrangements) there are command line tools to enable the user to obtain and cache credentials.

Although authentication is straightforward, Active Directory provides group membership, authorization, and other configuration information, and it is often desirable for a non-Windows workstation to be able to take advantage of such information. There are a few options for using AD infrastructure on non-Windows workstations as explained below.

Third Party Software

Several companies sell products that support non-Windows workstations accessing AD resources. Products are available for Mac OS X, Linux and most flavors of UNIX. Capabilities of these products typically include:

- Automatically configure Kerberos, including dealing with authentication paths inside the forest and finding the closest domain controller.
- Take advantage of group policy configuration on non-Windows workstations.
- Use AD to populate accounts and authorization on the workstation.

Winbind Service

The *Samba* product includes a component called *Winbind*. This service will access Active Directory services to lookup account and group information. Winbind also includes a PAM module. However, this module should *not* be used for *authentication;* instead, a Kerberos module should be used. On the other hand, the "account" role of this module can be used for *authorization.*

Pam_ldap and Nssldap

The *pam_ldap* module provides account authorization services. It can access an LDAP directory to determine which accounts should be used.

The *nssldap Name Service Switch* module will use a directory to look up account and group information. If this module is used, then potentially all the accounts in the directory can be available to a workstation. This module is available for Linux and Solaris.

One drawback of these modules is that they require the POSIX directory schema be available.

The pam_ldap module should *not* be used for authentication; instead, a Kerberos module should be used.

Kerberos Co-existence with other Authentication Mechanisms

In nearly all real-world environments, Kerberos must co-exist with other authentication mechanisms. In some cases, other authentication mechanisms are necessary for backwards compatibility, or there are alternative mechanisms that are better suited to specific applications or user groups. It is rarely feasible, or even desirable, to exclusively use just one authentication mechanism. At the same time, most environments have accumulated some baggage, including a few outmoded authentication mechanisms.

The challenge in mixed environments is to reduce or eliminate dependencies on older and weaker authentication technologies while creating opportunities to converge on a few stronger, more manageable technologies, such as Kerberos.

Of particular concern is the potential for non-Kerberos authentication mechanisms to adversely impact Kerberos authentication, especially through exposure of Kerberos passwords or other shared secrets. This topic is addressed in some depth below.

This section also includes discussion of multi-factor authentication and public key technologies that can be used with Kerberos to further strengthen authentication and reduce password vulnerabilities.

Exposure of Kerberos Shared Secrets (Passwords)

Kerberos authentication protects users' long-term shared secrets: these secrets are only used in initial authentication and are not directly sent over the network. Typically, users' shared secrets are derived from a password. If these passwords are sent over the network through processes other than the Kerberos exchange, they may be disclosed to attackers. Kerberos passwords are often more valuable than other passwords because Kerberos is a single-sign-on technology. Once an attacker has access to the Kerberos password, the attacker can access any resource that the Kerberos principal is authorized to access.

Various practices can lead to the disclosure of Kerberos shared secrets. For example, when some application or system does not directly support Kerberos, it is often relatively easy to send a password to that application and then to use some mechanism to confirm that this is the correct Kerberos password. The user gets to use their same password while minimizing any modifications needed to the application. This may be a reasonable risk tradeoff in some situations, especially when the password is protected by TLS or some other mechanism. It is definitely not as strong as using Kerberos directly. However, because the password is disclosed to another party, it is important to understand when such a mechanism is used and to evaluate the associated risks.

Understanding when passwords are disclosed is particularly difficult in a mixed environment because administrators may not be aware of all the technologies involved and how they may interact to disclose passwords. This section explains common situations that can cause password disclosures for both Microsoft and non-Microsoft environments. One recommended approach for finding out where passwords may be disclosed is to take a newly created account and to use that account to access the various applications in the environment. In the ideal case, the password will be requested at login but at no other point. If the user is requested to enter in their password other than at login, then there may be a password disclosure. The application that asked for the password should be examined carefully. This method is a good starting point, but it is not perfect, since the login process itself may disclose the password.

Of course, one way to prevent password disclosures is to avoid the use of passwords. If PKINIT and smart cards are used for Kerberos authentication, then passwords cannot be disclosed, because there is no password to disclose.[6] However, if the disclosure of passwords is enabling some application to work that would otherwise fail, then PKINIT will likely cause that application not to work. Of course, it will also make the points of password disclosure obvious.

Password Exposures in Windows Environments

A number of components of Windows systems are given access to the user's password during the login process. For example, third parties can install network providers that are given access to the user's password in order to gain network resources. Classically,

[6] Though the user may have a password or PIN to unlock their private key. This will depend on the technology used to protect private keys.

these providers were used to access network file storage, although the mechanism has been used for a variety of other uses.

A number of Microsoft applications will accept passwords supplied by users and verify them against Kerberos:

- Exchange will accept email and other passwords. This is used by Outlook, web and other clients especially when those clients are not part of a domain.
- IIS supports Kerberos authentication to web pages. However, it also supports password-based authentication. Depending on the exact form of the URL and on configuration settings, Internet Explorer is likely to prefer passwords to Kerberos.
- A number of applications, including file sharing, will use NTLM authentication when Kerberos is unavailable. NTML authentication is not as strong as Kerberos but is certainly better than sending passwords over the network

In addition, Active Directory provides a number of mechanisms that third-party applications can use to verify passwords:

- Applications can perform initial Kerberos authentication against Active Directory. As discussed in the "Role of Kerberos" section on Pluggable Authentication Modules, it is important that such verification also uses a service principal and perform authentication to that service principal. Unfortunately, this is often not done. Using Kerberos initial authentication is the recommended mechanism for workstation login. However when used to verify a password received by a network service, this mechanism involves a password disclosure before Kerberos is involved.
- Active Directory provides a full LDAP implementation. One part of LDAP is the ability to perform a "simple bind" to a directory account. Doing so will verify the password. This involves a password disclosure between the domain controller and the application doing a bind. However, if the application is itself a network service, there is an additional password disclosure involved.

Password Exposures in Non-Microsoft Environments

The easiest ways to expose passwords on non-Microsoft environments involve using *PAM* or *LDAP.* PAM is discussed in the *Role of Kerberos* section on Pluggable Authentication Modules. PAM provides a technology-neutral way to authenticate a user given a password or other information. PAM should be configured to authenticate local users of a workstation against Kerberos in order to perform authentication and to obtain Kerberos credentials for those users; this use of PAM does not involve a password disclosure. However, PAM can also be used with any application, not just local login applications. When PAM is used with network applications, such as mail servers, web servers, or database servers, it sends the password to the network application, which discloses the password to another party.

As discussed above, LDAP provides a mechanism to bind to an account using a password. The intended purpose of this mechanism is to access a directory by use of password-based authentication. Of course, even this intended purpose involves a

password disclosure. However, the mechanism is also used to verify the password for applications such as web servers.

There is one particularly problematic interaction: the combination of PAM and LDAP together. There is a PAM module called "pam_ldap" that will integrate PAM into a directory. It seems attractive to use this module with Active Directory. However, the authentication ("auth" role) of this module *should not be used* when Kerberos is available. The authentication component of pam_ldap sends the password to the directory, creating an exposure even for local accounts. It would be better to use a Kerberos PAM module, which would verify the password using a more secure mechanism and obtain credentials for the user. On the other hand, the "account" role of pam_ldap is useful in a mixed environment, because it can rely on the directory for retrieving authorization information.

Password Exposures in Java Environments

Java provides the *Java Authentication and Authorization Service* (JAAS). JAAS is similar to PAM and can be configured to verify passwords against Kerberos. As with PAM, JAAS should be used to verify passwords and obtain credentials for local services. Where available, JAAS may be configured to use existing Kerberos credentials. However, JAAS can also be used by network services to obtain credentials from a password; doing so causes a password disclosure.

Multi-Factor Authentication

Concerns with reliance on passwords as a means for authenticating users continue to mount as attacks that are more effective in compromising or purloining user passwords continue to increase. In many cases, passwords and user habits represent the weakest link in access control security, even when everything that can be reasonably done to protect passwords has been done. As the resulting threats increase, pressures are mounting for stronger authentication technologies.

One option for reducing exposures to password vulnerabilities is to introduce multi-factor authentication where additional *factors* are used as tests or proofs when determining the authenticity of claims. Traditionally, factors used in authentication are classified as:

- Knowledge of a secret—*something you know,* such as a password
- Possession of a unique device—*something you have,* such as cryptographic token or One-Time Password (OTP) fob
- Physical attributes of the claimant—*something you are,* such as fingerprints, iris images, or other biometrics

When authentication uses two or more independent factors in confirming authenticity, then confidence in authentication results will generally be much higher. Given the concerns about reliance on passwords, there is increasing pressure to adopt some form of multi-factor authentication. In some cases, use of multiple factors is being mandated by regulators, government agencies, or new security policies.

Effective deployment of multi-factor authentication is a difficult problem compounded by many subtle issues. While this broad topic is outside of the scope of this document, there are issues that should be considered within mixed environments.

The Kerberos Authentication Service (AS) has been architected to support multi-factor user authentication through the pre-authentication option. Specifications have been introduced into the IETF for supporting two classes of multi-factor authentication:

- IETF-Draft OTP Pre-Authentication details various options for using One-Time Password (OTP) devices and services with Kerberos
- RFC 4556 defines the PKINIT specifications for using public key cryptography with Kerberos, and RFC 4557 further specifies how Online Certificate Status Checking (OCSP) can be used with PKINIT

Based on these specifications, or possibly other proprietary extensions, Kerberos can play a valuable role in facilitating multi-factor adoption, since it can front-end the initial user authentication where multiple factors are employed without impacting the subsequent client-service authentication procedures. In other words, existing applications that use Kerberos for authentication do not need to be modified when multi-factor authentication technologies are introduced, as only the Kerberos AS is affected by the changes. Furthermore, different populations of users can use various multi-factor technologies, yet access shared applications that are insulated from the user authentication step.

By implication, efforts to integrate Kerberos throughout a mixed environment can also facilitate introduction of multi-factor authentication. Users could, for instance, sign on within one environment using multiple factors, and subsequently access services in some other environment through cross-realm relationships. However, if policies stipulate that an application service should only grant access when multiple factors have been used to authenticate the user, then the application will need some means for determining that tickets it receives are associated with users that were authenticated with multiple factors. Such policy constraints will tend to complicate mixed environment integration and interoperability.

Account mapping techniques represent one option that can be used to enforce multi-factor policies. However, this assumes that users can be mapped only to accounts that are associated with authorizations that indicate whether or not multi-factor authentication was enforced. Other authorization schemes can be extended to reflect whether users are required to authenticate with multiple factors. One approach is to include the type of authentication required for a user in the user's account record in a directory service. Information about the type of authentication performed for a user could also be included in tickets issued for services, and Kerberos does provide a simple *hw-authent* flag that can be set in a ticket when some form of hardware-based authentication was performed. The problem with all of these approaches is that they can be difficult to successfully utilize within one environment, never mind in a mixed environment with other administrative and policy regimes in the mix.

Multi-factor authentication introduces other challenges that will complicate deployments in mixed environments, including:

- Software drivers for authentication devices (*e.g.,* smart card readers, biometric scanners) that might be needed on user workstations, laptops, and handhelds
- Additional interoperability standards for specialized information exchanges and data structures (*e.g.,* PKCS 11, X509 certificates, OTP exchanges)
- Third-party services that might be required for OTP checking, matching biometric parameters, or issuing certificates
- Interactions with workstation login software that may be required to initiate Kerberos authentication with multiple factors
- Proprietary solutions that may not work across different environments
- Integration with KDCs may be platform or implementation specific—i.e., a multi-factor solution might not work with all types of KDCs found within a mixed environment

"Forewarned is forearmed" is what this document can offer. Real-world deployments of multi-factor authentication schemes will require careful coordination with vendors, effective deployment strategies, selection of interoperability standards, substantial testing, and patience. When planning for mixed environments, it is important to recognize that multi-factor authentication requirements can significantly impact plans. However, if confronting requirements to support multi-factor authentication, then Kerberos integration may be essential to rationalizing plans and reducing project risks. It may also be possible to combine multi-factor techniques with Kerberos to eliminate or reduce other integration challenges. In this regard, PKINIT is worth considering.

Public Key Certificates and Kerberos PKINIT

The PKINIT extensions to Kerberos allow use of public/private key pairs instead of user passwords or shared secrets (refer to the *Role of Kerberos* document for background). Furthermore, the RFC 4556 proposed standard specifies optional use of public key certificates as a means for associating users with their public keys. Use of certificates requires some reliance on a Public Key Infrastructure (PKI) to issue and manage certificates. PKI is often associated with directory services that can be used to request certificates for specific user accounts. Collectively, Kerberos, PKINIT, PKI, and LDAP directory services represents a large body of industry standards that have been maturing over the past decade and that are now widely supported by platform vendors, other technology vendors, applications developers, and service providers.

While use of PKINIT with multi-factor solutions presents all of the same challenges noted above, it also offers some advantages as well as opportunities to minimize some integration challenges. The *Role of Kerberos* document provides further background on some of the benefits of using PKINIT, with the key points summarized as:

- Users do not need to share a password with the KDC since they provide their public key instead
- Inherent weaknesses with user-memorable passwords are eliminated
- Accounts can more easily be moved or established in other realms (Domains), since the password-sharing step can be eliminated

- Certificates and key pairs can be installed as software objects on user workstations instead of requiring hardware multi-factor devices, yet still provide the benefits of reducing exposures to weak passwords
- Numerous hardware options already exist for using public/private keys and certificates as the basis for multi-factor authentication, including smart cards, USB tokens, and even some biometric scanners
- Certificate issuing can be outsourced to third parties, or managed in-house
- Other authentication mechanisms and security services can also utilize public key certificates, including TLS (SSL) and VPN tunnels
- PKI provides additional options for enforcing compliance with policies, especially when combined with directory services

Without minimizing the challenges and difficulties associated with deploying multi-factor technologies based on PKI and PKINIT with Kerberos, this approach does offer advantages, and the maturity of the standards improves options for achieving interoperability. In a mixed environment, this approach may be the best strategy for meeting multi-factor requirements, and it is a credible path to pursue going forward.

General Guidance on Kerberos Deployments

This section highlights a few issues that, while generally important in Kerberos environments, are particularly important in mixed environments:

Clear Functional Architecture

A mixed environment may take on any of several patterns:

- Windows Active Directory domains and generic Kerberos realms, with the two completely unconnected.
- Windows AD domains and generic Kerberos realms, with the two kept synchronized:
    - With all updates *only* on the Windows side, with changes propagated to the generic Kerberos realm
    - With all updates *only* on the Unix side, with changes propagated to the Windows AD domain
    - With updates permitted on either side, with changes replicated to the other
- Various permutations for different sub-realms or sub-domains within an overall realm or domain.

Misconfiguration bugs can be insidious. It must be clear which pattern is being used, with adequate documentation, and analysis to verify that any process capable of updating credential information is configured appropriately.

Similarly, although not strictly a part of Kerberos, the relationship between attribute, group membership, and authorization data maintained in the directory by the Windows Active Directory environment and the same type of data maintained by directories

within the generic Kerberos environment must be made equally clear. Appropriate planning should be conducted with the objective of rationalizing maintenance of information that needs to consistent across different environments.

Deprecated Versions

Use of deprecated versions of Kerberos, or of underlying operating system components, should be discontinued, especially in mixed environments where there might be unanticipated interactions. In particular, avoid continued use of Kerberos v4, which has known security vulnerabilities. If any applications require v4, such applications should either be upgraded or isolated to their own independent realms.

In a mixed environment situation, it is recommended that a careful inventory be conducted of all applications and administrative tools used to provide or interact with Kerberos services. Such a practice is recommended for any security technology that is widely used, but it is particularly important when extending Kerberos services across various administrative regimes and environments.

Pre-Authentication

All modern implementations of Kerberos support pre-authentication of users as part of the initial authentication process when the user's client requests a TGT from the Authentication Service (AS) of a KDC or Domain Controller. The essential purpose of pre-authentication is to force mutual authentication between the client and KDC/AS *before* the AS returns a TGT to the client.

When pre-authentication is used, an initial authentication request from a client is returned by the KDC/AS with a notice that pre-authentication is required along with a random session key encrypted with the user's secret. The client must respond using the session key to prove that the client possesses the shared secret. This means that the AS is able to confirm that the client is acting on behalf of the user before the AS provides a TGT. At the same time, the clients (and users) are able to confirm the authenticity of the KDC (AS). Without pre-authentication, the KDC is not able to confirm the authenticity of the user/client until the client makes its first request for a service ticket using the TGT. Pre-authentication is also used with public key initial authentication (PKINIT) and with some multi-factor schemes, especially OTP devices.

Windows Domain Controllers always use pre-authentication, but this is an optional feature with other Kerberos implementations. It is recommended that pre-authentication be required in *all* Kerberos realms. This improves security and makes it easier to assure policy compliance for cross-realm authentications that take place in mixed environments. It also strengthens the audit value of system logs (see Establishing Audit Trails below).

Forcing use of pre-authentication is a configuration option that can be invoked on KDCs. The potential negative consequence of concern is that some older clients may not handle pre-authentication properly. Such clients should be upgraded.

Avoiding use of Weak Cryptographic Algorithms

Kerberos supports multiple encryption algorithms, including some older algorithms that are considered weak by today's standards—in particular, variants of DES.[7] While use of deprecated cryptographic algorithms is widely recognized as a concern, it is necessary to carefully audit systems to insure that default conditions have not resulted in situations where weak algorithms will be used unintentionally.

To be clear, the preferred symmetric cryptographic algorithm today is AES[8] with 256-bit key lengths. RFC 3961 and RFC 4120 provide technical details on what combinations of cryptographic algorithms are available, and how they can be selected. In addition to symmetric encryption algorithms, there are also hash (a.k.a., message digest) algorithms and keyed hash (a.k.a., HMAC) schemes that are used in protecting Kerberos messages. In terms of hash algorithms, SHA-1 is preferred, but MD5 is an acceptable choice when used with keyed hash schemes. It is beyond the scope of this document to discuss the criteria that apply to selecting cryptographic algorithms. What *must* be noted, however, are the issues that emerge in mixed environments that can impact the choices of algorithms, sometimes in unexpected ways.

Historically, most security technologies developed in the past couple of decades that employed cryptographic algorithms included support for the DES family. However, external policies, such as government export/import restrictions, led to curtailment of support for DES in some commercial products. This caused Microsoft to first release Kerberos support in Windows NT and 2000 with only support for the RC4[9] algorithm, which supported variable-length keys, and could thus be exported if restricted to using short keys.

While many of the past policy constraints no longer apply, history has dealt some cards that impact current practices. Although Microsoft did not originally support DES, Windows 2003 server and XP added support for DES for compatibility with other Kerberos implementations. Similarly, MIT Kerberos, along with some other implementations, now supports the RC4 algorithm in a manner consistent with Microsoft's implementations. Furthermore, the current standards have been updated to reflect all cryptographic algorithms available for Kerberos use. The standards also stipulate that AES support is required.

Preferred Cryptographic Algorithms

As noted above, the preferred symmetric encryption algorithm for modern applications is AES with 256-bit keys, though use of 128-bit keys may be adequate for some low-risk applications. The preferred hash algorithm is SHA-1. For backwards compatibility with

---

[7] Data Encryption Standard—originally established as a Federal Information Processing Standard (FIPS) by the U.S. Government in 1976.

[8] Advanced Encryption Standard—published by the U.S. National Institute of Standards and Technology (NIST) in 2001 as FIPS 197.

[9] Originally, a proprietary algorithm belonging to RSA Laboratories and developed by Ron Rivest. It has since been published in various places, including in this IETF Draft: http://tools.ietf.org/id/draft-kaukonen-cipher-arcfour-03.txt

the large installed base of Windows 2000 and 2003 servers, and Windows XP workstations, RC4 support is necessary, though it is certainly desirable to begin the process of deprecating use of RC4. Since Kerberos only supports various combinations of algorithms as explained in RFC 3961, the options that should be allowed in priority order are:

(1) AES with 256-bit keys used with SHA-1 and 96-bit values
RFC 3961 etype of 18 (aes256-cts-hmac-sha1-96)

(2) AES with 128-bit keys used with SHA-1 and 96-bit values
RFC 3961 etype of 17 (aes128-cts-hmac-sha1-96)

(3) RC4 with HMAC—for Microsoft compatibility (pre-Vista and 2008)
RFC 3961 etype of 23 (rc4-hmac)

Fortunately, AES performance tends to be better than for older algorithms such as DES and RC4. Furthermore, in most practical situations, there are no performance disadvantages with using 256-bit keys with AES. However, some Kerberos implementations only support 128-bit keys due, again, to concerns about export restrictions. This can complicate choice of key size, but these issues can often be dealt with on a case-by-case basis.

Where interoperability is required with Microsoft systems introduced prior to Vista and Windows 2008 server, the RC4 algorithm will need to be supported. However, AES should be used with current and forthcoming Microsoft products.

Issues in Selecting Cryptographic Algorithms

Unfortunately, it may not be obvious what cryptographic algorithms are actually being used in deployed Kerberos systems. This can be due to several complicating factors, combined with a tendency for implementations to default to using DES algorithms. Mixed environments can definitely contribute to situations where algorithms are used in unintended ways.

For example, while Windows systems will prefer AES or RC4 algorithms, the defaults when adding support for non-Windows systems will be to use DES. While these defaults can be overridden, system administrators should be aware of this behavior. Other vendor implementations are known to take similar defaults, so the recommendation is to avoid allowing defaults to govern what cryptographic algorithms are used unless it can be concretely demonstrated that the default choice is appropriate. The caution is that a default for configuring one type of system may be different when configuring another type of system.

Another point to note is that many administrator configuration tools will generate keys for multiple algorithm types. This is done to improve interoperability so that two systems can negotiate the algorithm they will use. However, this leaves to *chance* what should be *explicitly determined.* Since the KDC databases have all keys for all principals known to the KDCs in a realm, it is recommended that these databases be audited on a regular basis to make sure that keys for older algorithms are not installed. Clearly, if a DES key is not available, then DES cannot be used. Again, the caution is that supporting

Kerberos in a mixed environment may result in keys being generated for algorithms that should not be used.

Key Generation and PRNGs

If cryptography has an *Achilles' heel,* it is in the way that random numbers are generated. While brute-force attacks on the keys used with modern cryptographic algorithms are not considered feasible, this is only true when there is no way of guessing any of the bits in the key. However, if a software pseudo-random number generator (PRNG) is not sufficiently random, then an observer may be able to determine some of the bits in the key, which can lead to key compromises. The essential point is that where, and how keys are generated matters a lot to the overall security of an operational system.

In a mixed environment, keys are likely to be generated on systems in one environment that will then be used in other environments. This increases the scope of concern for security administrators who have responsibility for key management. As recent events have taught,[10] it may be very important to keep track of where keys were generated as well as what key generation machinery was used, and even how it was used. While it might be obvious what the key generation procedures are in a homogeneous environment, in a mixed environment, it is recommended that the inventory of keys include information about where and how keys were generated. Then, if a problem is found with one specific key generation procedure, then it will be possible to re-generate the suspect keys, or even to disable access based on which keys are suspect.

Establishing Audit Trails

One area where mixed environments can substantially complicate system management is in keeping track of system logs and events that may be important from a security auditing perspective. Many enterprises have policies that require tracking all attempts to sign onto services. Kerberos, by its very nature, provides a useful audit point, since it provides centralized authentication services. Furthermore, KDC logged events are generally much harder for attackers to corrupt. Kerberos event logs also provide significant event records that can be correlated with events captured by other systems and services. Active monitoring of KDC events can be an effective means for detecting certain types of attacks.

However, in a mixed environment, system logging in one environment will generally not see potentially important authentication events that take place in other environments. Put another way, system logging is yet another application that often needs to be integrated across different environments, and this is an application with its own unique access control requirements.

[10] An illustrative example is that it was discovered a couple of years after the fact that a serious coding error was introduced in the Debian Linux branch that substantially weakened the PRNG included in the OpenSSL package. This wasn't the first time such problems have been discovered, and it won't be the last.

There are many tools available today for capturing log records in a distributed manner with intelligent filtering and data aggregation. There are also tools for analyzing log records and producing stable archives. Using such tools to capture and analyze Kerberos events throughout an enterprise is a recommended practice, both in support of audit objectives and to improve system management oversight. Given the added complexities of mixed environments, an effective, comprehensive system logging capability will likely prove invaluable for debugging some types of system problems and for finding unexpected system behaviors.

