By William Stallings, Comp-Comm Consulting Co.

Kerberos Keeps the Enterprise Secure

Available commercially or in the public domain, Kerberos can protect critical data without putting too much strain on network bandwidth

Data Comm MAGAZINE
NETWORK CLINIC

Client-server. Workgroup. Collaborative. Peer-to-peer. Whatever buzzwords apply and however a business distributes its information resources, corporate users require shared access to files, databases, and applications. And sharing means that security is a primary concern.

In an unprotected network environment, any client can apply for access to any server. To limit access to those with permission, servers must be able to confirm the identities of those making the requests—and each server must do this for each client-server interaction. In an open environment this creates overhead that places a substantial performance burden on the server.

Kerberos, a security software utility developed at the Massachusetts Institute of Technology (Cambridge, Mass.) as part of its Project Athena, was designed for distributed networks. Now available in both commercial and public-domain implementations, Kerberos has been declared an Internet standard and has become the de facto standard for remote authentication in networked client-server environments (see "Kerberos by the Numbers"). It can define multiple secure areas, or realms, on a single enterprise network. What's more, if it is deployed across a few large realms, Kerberos places surprisingly little drain on network bandwidth or server resources.

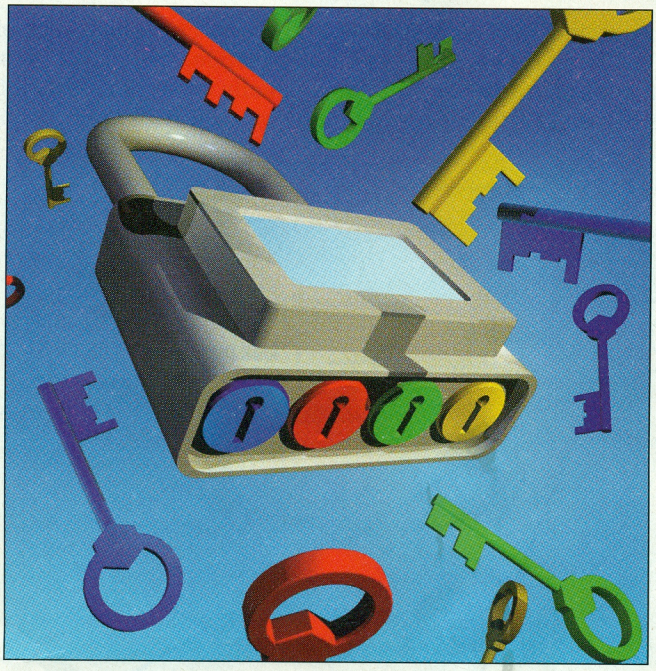


ILLUSTRATION BY GIACOMO MARCHESI

William Stallings is the president of Comp-Comm Consulting Co. (Brewster, Mass.) and a frequent lecturer on networking topics. He also is the author of more than a dozen books on data communications, the latest of which is *Network and Internetwork Security: Principles and Practice* (Prentice-Hall/Macmillan).

Unlike many of today's popular security schemes—such as password generators or biometric devices that read fingerprints or the eye's iris patterns—Kerberos tackles network security without additional hardware. Instead, it relies on encryption to secure network transmissions and a password-authentication service to verify a user's right to access a given host or server. It also imposes time limits on access requests to minimize the potential for unauthorized use of access privileges.

GUARDIAN OF THE REALM

Kerberos, which gets its name from the three-headed dog that guards the gates of Hades in Greek mythology, relies on a "trusted-third-party" authentication service. The service is "trusted" in the sense that all network clients and servers rely on Kerberos to validate their own mutual authentication procedures. In other words, Kerberos is the ultimate arbiter of security on the network. It requires that a user prove his or her identity for each service invoked and, optionally, can require servers to prove their identity to clients.

To minimize processing overhead, Kerberos employs a distributed client-server architecture. The Kerberos client initiates all security transactions on behalf of the user. Each workstation, PC, or other network node must be running Kerberos client software. The Kerberos server consists of two pieces: an authentication server, which verifies identities, and a ticket-granting server, which gives clients permission to access various servers and applications on the network. The Kerberos server can reside on any network system, either alone on a dedicated system or sharing a system with other applications.

Kerberos' client-server design can accommodate the most complex distributed networks. A Kerberos environment—consisting of a Kerberos server; numerous

Kerberos Protection

Kerberos clients; and, typically, several application, file, and print servers—requires only two basic conditions to be fully secure. First, the authentication server must have the user ID and password of all participating users in its database. Second, the authentication server must share a secret key with each server on the network, and all servers must be registered in the authentication server's database.

An environment that meets those two conditions is a Kerberos realm. In a typical implementation, networks of clients and servers belonging to different administrative organizations usually constitute different realms (see Figure 1). It is generally not practical to have users and servers for one department or administrative domain registered with a Kerberos server that resides on a portion of the network that is managed by another department. However, because Kerberos servers can communicate across internetwork links, users in one realm may—with appropriate privileges and authentication—access servers in other realms.

KEYS TO THE KINGDOM

Because the environments it is designed to secure can be quite complex, the Kerberos service consists of several components that perform different functions.

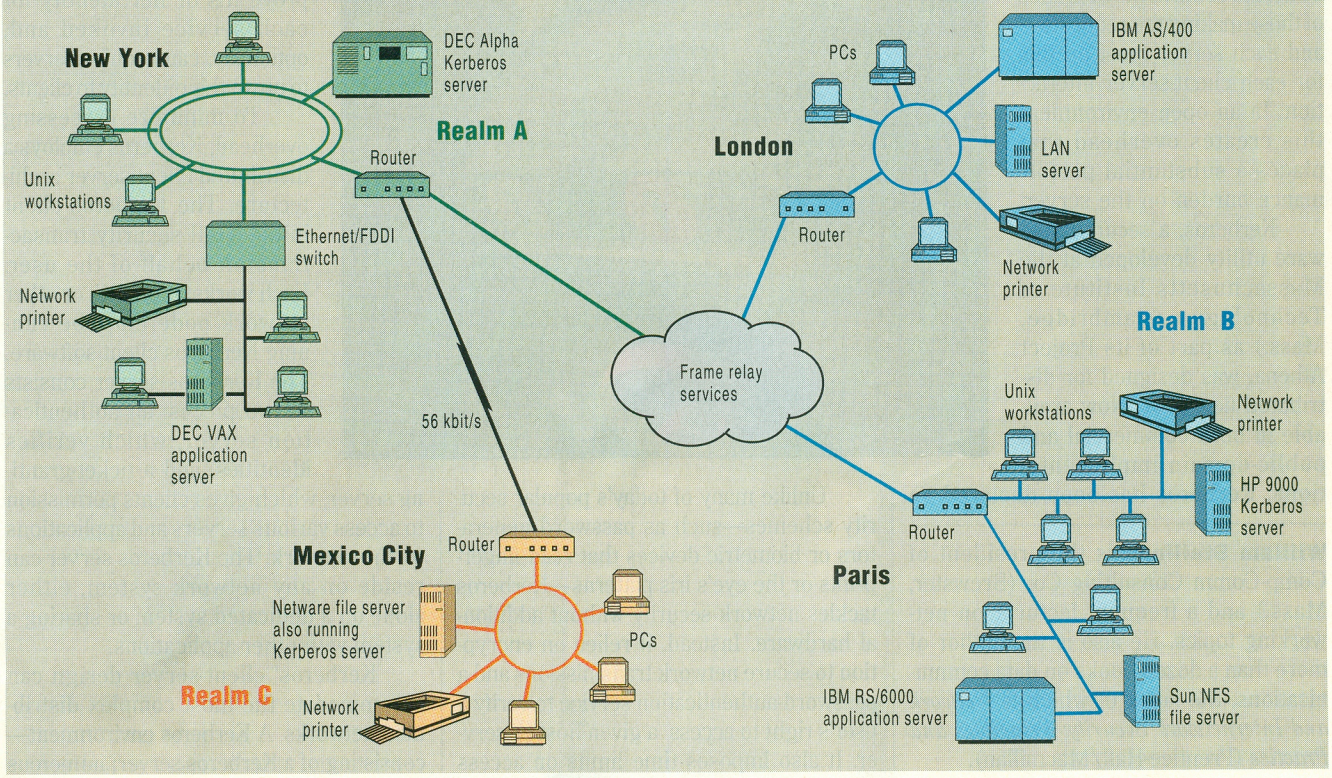
The Kerberos authentication server stores the passwords of all network users in a centralized database. Users (Kerberos clients) must log into the authentication server for identity verification before attempting to access any other server on the network. Once the authentication server has verified the user's identity, the client can proceed to request access to other servers. As needed, the authentication server passes security verification to the servers or other network devices that the client wants to access. Only when these devices have received verification from the authentication server can they safely accept service requests from the client.

That protocol may sound sufficiently Byzantine to satisfy security mavens, but there's a catch: How does Kerberos secure communications among the authentication server, the client, and the server? It simply won't do to have the client send the user's password to the authentication server over the network, because someone could observe the password on the network and reuse it. Likewise, it won't do for Kerberos to send a plain message to a server validating a client's security. An unauthorized user might then impersonate the authentication server and send false validation to a server.

This is where encryption becomes

Figure 1: A Kerberos Secure Enterprise Network

A Kerberos enterprise implementation may consist of several Kerberos realms, each of which typically contains a Kerberos server; Kerberos clients; and several application, file, and print servers. Clients and servers belonging to different administrative organizations usually constitute different realms. Using interrealm communications between Kerberos servers, clients in one realm may, with the appropriate privileges, obtain access to servers in another.



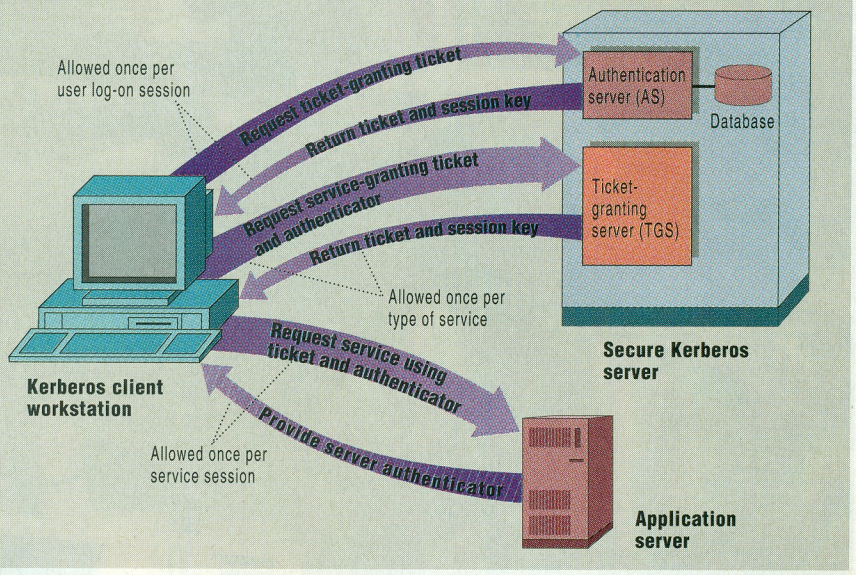
important. The authentication server shares a unique secret key with each server on the network. These keys have been distributed physically or in some other secure manner as part of the Kerberos installation. When the authentication server sends a verification message to a server, it encrypts the message so that only the server with its unique key can decode it (see Figure 2). Kerberos uses the DES (Data Encryption Standard) algorithm to generate the keys and encrypt messages.

The Kerberos client sends a message to the authentication server that includes the user's ID and a request for what is known as a ticket-granting ticket. The authentication server checks its database to find the password of this user and responds with a ticket-granting ticket and a one-time encryption key, known as a session key. The ticket-granting ticket will be used to gain access to the ticket-granting server, the other major component of the Kerberos server; the session key will be used in additional encryption and authentication procedures. Both the ticket-granting ticket and session key are encrypted with the user's password as the encryption key.

When this message arrives back at the user's workstation, the Kerberos client prompts the user for his or her password. Once the user supplies the password, the client generates a key and attempts to decrypt the incoming message. If the user supplies the correct password, the ticket-granting ticket and session key are successfully decrypted. Notice that the authentication server verifies the user's identity without requiring the password to travel over the network.

Figure 2: Overview of a Kerberos Secure Transaction

Kerberos clients log into the authentication server (AS) to obtain a ticket-granting ticket and a session key. The ticket-granting ticket allows access to a ticket-granting server (TGS), from which the client may request a service-granting ticket that allows access to application, file, or print servers in that Kerberos realm. Session keys are used to encrypt transactions across the network and create authenticator messages for verifying access privileges.



TICKETS TO RIDE

The ticket-granting ticket indicates that the authentication server has accepted the client and its user. It contains the user's ID, the ticket-granting server's ID, a time stamp, a duration period (the time for which the session is valid), and a copy of the client's session key. The entire ticket-granting ticket is encrypted using a secret DES key, which is shared by the authentication server and the ticket-granting server. This encryption ensures that no one can tamper with the ticket-granting ticket, and only the specified server or the authentication server can decode it.

However, Kerberos does not require the client to log in and obtain a new ticket-granting ticket from the authentication server every time the user wants to access another server or device on the network. The other component of the Kerberos server, the ticket-granting server, handles these access requests. The ticket-granting ticket provided by the authentication server permits the client to access the ticket-granting server, which then supplies tickets that permit access to other hosts and servers on the network.

This reusable ticket-granting ticket is encrypted two times: first, with a secret key known only to the authentication server and the ticket-granting server, then again with the user password. The first encryption prevents anyone on the network, including the user, from altering the ticket-granting ticket. Only the authentication server and the ticket-granting server know its code. This ensures that no one can forge or commandeer a ticket-granting ticket, fool the ticket-granting server, and obtain tickets to other servers.

The second encryption of the ticket-granting ticket uses a key based on the user's password, so that only the user can recover it. This enables the user to access the ticket and use it to request access tickets for other servers on the network. Note that the user can decode only the second encryption: The ticket-granting ticket itself remains encrypted with the key known only by the authentication server and ticket-granting server.

The ticket-granting ticket includes the user's ID, network address, the secret key that the client received at login, and a time stamp indicating the date and time it was issued and the time for which it is valid. This time period, which is established by the network administrator, can be configured according to the needs and rights of the user or the security requirements of the network; it could be 30 minutes or eight hours, depending on the environment. Making the ticket-granting ticket time-sensitive prevents an unauthorized user from capturing it and using it at a later time.

THE GREAT AUTHENTICATOR

Despite the layers of password protection and key encryption, the client still

Kerberos Protection

Kerberos by the Numbers

The most widely used version of Kerberos is version 4, which has been around for several years. MIT's Project Athena, the group that developed Kerberos, recently released version 5, which contains several useful enhancements.

Among the most important of the new features is the ability to use security algorithms other than DES (Data Encryption Standard). Recently, there has been concern about the strength of DES, and many system administrators wanted to use more complex, hacker-resistant encryption algorithms. So far, there has been relatively little consensus on other algorithms, but one likely alternative is Triple DES, an enhanced version of the DES encryption scheme.

To identify the encryption algorithm in use, version 5 tags encrypted messages with an encryption algorithm identifier. This identifier informs all participating clients and servers which algorithm the Kerberos server is using. All Kerberos clients and servers must be configured to use the same algorithm for transactions to take place. A client or server using a different algorithm than the Kerberos server will be excluded from all secure transactions.

Kerberos version 5 also supports a technique known as authentication forwarding. version 4 does not allow a client to access a server, forward its credentials to the server, and have that server access another server on the client's behalf. This may not sound like much of a problem because the client could always request a service-granting ticket and gain access itself, but there are instances where this can be decidedly inconvenient.

For instance, suppose a client wants to issue a request to a print server, which then needs to access a network file server to obtain the file that the user wants to print. Without the client's credentials, it can't. Version 5 provides this capability. Similarly, thanks to authentication forwarding, version 5 also supports a method for inter-realm authentication that requires fewer secure key exchanges than version 4. —*W.S.*

needs more than a ticket-granting ticket to convince the ticket-granting server that it is authorized to use services on the network. As mentioned, the initial message from the authentication server to the client contains a secret session key, which is shared with the ticket-granting server and buried in the ticket-granting ticket. The client uses this session key to encrypt what is known as an authenticator. The authenticator contains the ID and address of the user and another time stamp. Unlike the ticket-granting ticket, which is reusable, this authenticator is intended for one-time use and has a very short life span—typically a few minutes.

The Kerberos client next sends a message including the ticket-granting ticket and the authenticator to the ticket-granting server requesting a ticket for access to a desired server (for example, a file or application server). The ticket-granting server decodes the ticket-granting ticket, makes sure that it has not expired, and verifies the user's ID encrypted in the ticket. Then, to authenticate the source of the request, the ticket-granting server compares the user's ID and network address encrypted in the ticket-granting ticket with the address and ID of the incoming message.

At this point, the ticket-granting server is almost ready to grant a service-granting ticket to the client. But it still must verify the user's identity. When the ticket-granting server receives the client's request, it uses the client's session key, which it receives from the authentication server, to decrypt the authenticator.

Once the authenticator is decoded, the ticket-granting server can check the user ID and network address from the authenticator against that included in the ticket-granting ticket and against the source information of the incoming message. If all match, the ticket-granting server is assured that the sender of the ticket is indeed the ticket's real owner.

Note that simple possession of a ticket-granting ticket does not prove the user's identity. It is merely a way to distribute keys securely. It is the authenticator that proves the client's identity. Because the authenticator can be used only once and for a limited time, it is nearly impossible for an unauthorized user to steal both the ticket-granting ticket and an authenticator for later use. Each time it applies to the ticket-granting server for a new service-granting ticket, the client sends its reusable ticket-granting ticket plus a fresh authenticator.

COMPLETE SECURITY

When the ticket-granting server has successfully verified the source of the request and authenticated the user's identity, it sends the client two things: a reusable service-granting ticket (for access to the requested server) and a new session key. The service-granting ticket is encrypted with a secret key shared only by the ticket-granting server and the server that the client wants to access. The service-granting ticket also contains within it a copy of the client's new session key.

This entire message is encrypted with the client's old session key, so that only the client can read it. Once the client decodes the message, it then sends both the service-granting ticket and an authenticator encrypted with the new session key to the server it wishes to access.

At this point, the client also can choose to verify the authenticity of the server before proceeding with its transaction or request for service. This mutual authentication procedure prevents any possibility of an unauthorized user impersonating a server in attempt to gain access information from a client.

If mutual authentication is desired, the client requests that the server reply with the value of the time stamp from the authenticator, incremented by 1, and encrypted in the session key. The server then extracts the copy of the session key embedded in the service-granting ticket, uses it to read the authenticator, increments the time stamp by 1, re-encrypts the message using the session key and returns it to the client. The client then decrypts this message to recover the incremented time stamp, which assures the client that this is

Kerberos Protection

not a replay of an old reply.

At the conclusion of this process, client and server are assured of secure transactions. They also now share a session key, which they can use to encrypt future messages.

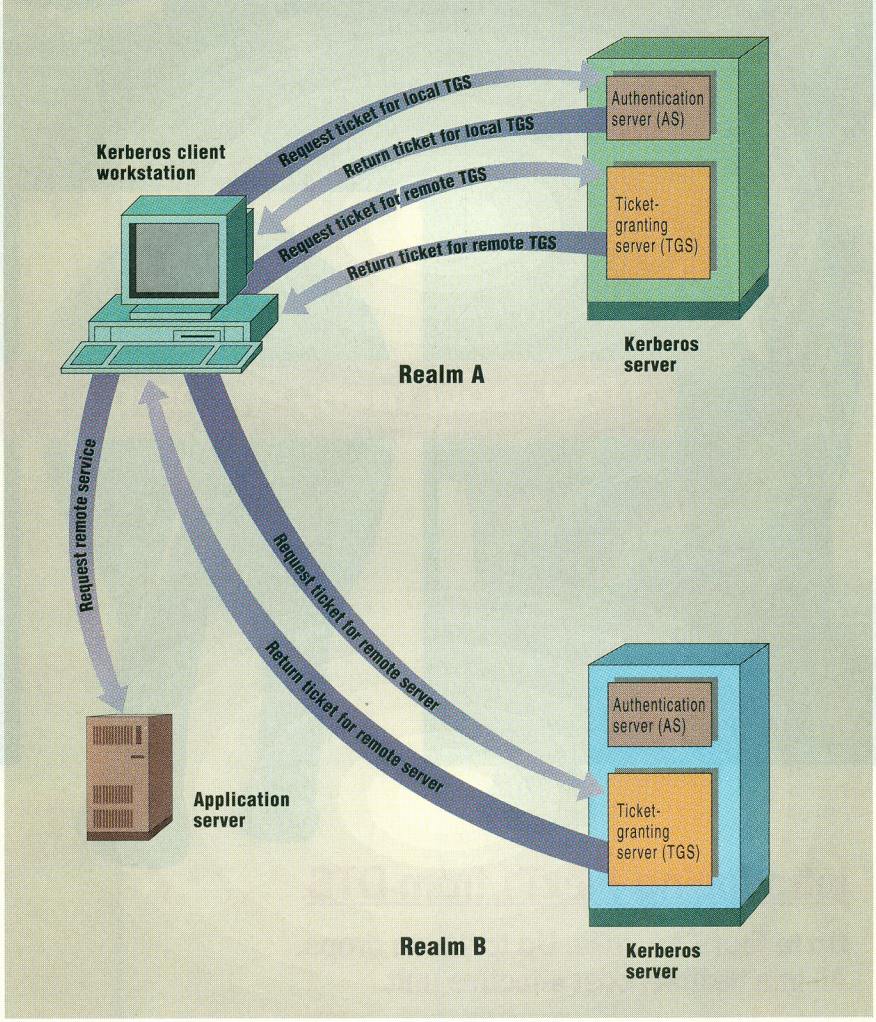
Kerberos also provides a mechanism for supporting inter-realm authentication. The two Kerberos servers are registered with each other, which means that the Kerberos server in one realm shares a secret key with the server in another realm. The scheme requires that they trust one another to authenticate users. Furthermore, the participating servers in the second realm also must be willing to trust the Kerberos server in the first realm.

A client desiring access to a server in a different Kerberos realm would first request a ticket from its own ticket-granting server for access to the ticket-granting server of the other realm (see Figure 3). With this ticket in hand, the client then applies to the remote ticket-granting server for permission to access the desired server. This service-granting ticket, which the client presents to the remote server, indicates the realm in which the user was originally authenticated. The remote server then chooses whether to honor the client's request.

The one problem presented by this approach is that it does not scale terribly well. If there are n realms, then there must be $n\ (n-1)/2$ secure key exchanges so that each Kerberos realm can operate with all other Kerberos realms. That creates a lot of overhead on the network and the Kerberos servers themselves. For this reason, the most successful Kerberos implementations will use a few relatively large realms rather than a vast number of small discrete realms.

Figure 3: Interrealm Requests

When a client in one Kerberos realm wants to access a server in another realm, the client sends its local ticket-granting server (TGS) a request to access the remote ticket-granting server. If the client is authorized, it is issued a ticket-granting ticket for the remote realm, which it uses to request a service-granting ticket for an application, file, or print server.



KERBEROS AND PERFORMANCE

As client-server applications become more popular, larger and larger distributed networks are appearing. The more important these networks become to the businesses they support, the more important it is to have Kerberos-caliber log-on authentication. But with all of its message encryption and key exchanges, what kind of effect will Kerberos have on network performance and bandwidth consumption?

Fortunately, when Kerberos is implemented and configured properly, network performance degradation is negligible. Because tickets are reusable, the amount of bandwidth required for the ticket-granting requests is modest. Although the transfer of a ticket for log-on authentication consumes some bandwidth, this log-on exchange must take place anyway. The extra overhead is negligible. Paul Cormier, engineering manager for Kerberos products at Digital Equipment Corp. (Maynard, Mass.), reports that response times with Kerberos installed are essentially the same as they would be without it—even on very large networks with tens of thousands of nodes.

A related issue is whether it is best to dedicate a specific hardware system to the Kerberos server software or install it on the same server as another application. Technically, Kerberos doesn't care, but it is not wise to run the Kerberos server on the same machine as a resource-intensive application, such as a database server or financial application. Moreover, the security of the Kerberos database is best assured by placing the Kerberos server on a separate isolated machine.

Finally, for large networks, will implementing multiple Kerberos realms—dis-

persing Kerberos servers throughout the network—help maintain performance? Probably not. The 32,000-node application cited above is supported by a single Kerberos server. The primary motivation for segmenting the network into multiple realms is ease of administration. If geographically separate clusters of machines must be serviced, each with its own network administrator, then one realm per administrator may be convenient. However, this is not always the case.

For example, DEC recently installed a Kerberos system for a bank with major computing concentrations in New York, Paris, and London, each location with its own network administrator. DEC initially recommended three Kerberos realms. However, the customer wanted to allow all users to roam freely around servers at all three locations, so the Kerberos facility was configured as a single-server, single-realm system with the Kerberos server located in the New York office. Despite the need for wide-area communications between all European clients and the Kerberos server, users noticed little deterioration of response times.

KERBEROS RISING

Kerberos does not solve all of the security problems facing network managers, but it does address many of the concerns that arise in client-server environments. There are several other approaches organizations can use to secure networked hosts and servers, but many require specialized hardware, which can be cumbersome and expensive.

Systems that use one-time passwords thwart attempts to guess or capture a user's password. However, these systems, the most notable of which is Secure ID from Security Dynamics Inc. (Cambridge, Mass.), rely on special equipment such as smart cards or synchronized password generators.

Another approach is a biometric system—an automated method of verifying a user's identity on the basis of unique physiological characteristics. A fingerprint or iris pattern, or even handwriting or keystroke patterns, can serve to validate a user's identity. Although technologies like these are effective, the additional cost and inconvenience they entail makes them unattractive for widespread use.

There are a few software alternatives that provide security for distributed environments, but most are limited to a single network operating system or a limited range of platforms. Novell Inc. (Provo, Utah) offers a proprietary authentication capability as a Netware NLM (Netware Loadable Module), and IBM offers Net SP for its LAN Server PC networks and for Netware. CA-Unicenter from Computer Associates International Inc. (Islandia, N.Y.) offers multiplatform, distributed security, but the server currently runs only under HP-UX, IBM AIX, and SunOS. The vendor does sell the security component as a separate product, but users must purchase the whole suite of system management tools to obtain it.

Kerberos, in contrast, is highly portable across operating systems and hardware platforms. It is intended for use in all types of environments and, because it is in the public domain, it can be implemented by any vendor or user at relatively low cost. For this reason, standards bodies have championed Kerberos; even vendors that currently offer competing schemes may eventually adopt it or integrate it into their existing products.

Kerberos has been gaining increasing acceptance of late. The Open Software Foundation's Distributed Computing Environment (DCE) uses Kerberos version 5 for user authentication. And several vendors, including DEC, Hewlett-Packard Co. (HP, Palo Alto, Calif.), and IBM, now offer Kerberos implementations as part of their standard middleware offerings on commercial Unix and midrange server platforms.

The public-domain version of Kerberos, which can be obtained via anonymous FTP from a server at MIT, also enjoys widespread use. Although it is more convenient to purchase Kerberos from a vendor that will provide customer service and support, do-it-yourself types, untroubled by the absence of formal technical support, have found it easy to obtain and install. For specific instructions on how to access and download the latest version of Kerberos, use telnet to reach athena-dist.mit.edu, then switch to pub/kerberos and get the file readme.krb4 or readme.krb5. ■

REQUEST FOR COMMENT
If you would like to see more articles on this subject please circle **464** on the Reader Service Card.