PNL-10382
UC-400

FINAL REPORT AND RECOMMENDATIONS OF THE ESNET AUTHENTICATION PILOT PROJECT

G. R. Johnson
C. L. Athey[a]
D. E. Engert[b]
J. P. Moore
J. E. Ramus[c]

January 1995

Prepared for
the U.S. Department of Energy
under Contract DE-AC06-76RLO 1830

Pacific Northwest Laboratory
Richland, Washington 99352

(a) Lawrence Livermore National Laboratory
Livermore, California
(b) Argonne National Laboratory
Argonne, Illinois
(c) National Energy Research Supercomputer Center
Livermore, California

DISCLAIMER

SUMMARY

To conduct their work, U.S. Department of Energy (DOE) researchers require access to a wide range of computing systems and information resources outside of their respective laboratories. Electronically communicating with peers using the global Internet has become a necessity to effective collaboration with university, industrial, and other government partners. DOE's Energy Sciences Network (ESnet) needs to be engineered to facilitate this "collaboratory" while ensuring the protection of government computing resources from unauthorized use. Sensitive information and intellectual properties must be protected from unauthorized disclosure, modification, or destruction.

In August 1993, DOE funded four ESnet sites (Argonne National Laboratory, Lawrence Livermore National Laboratory, the National Energy Research Supercomputer Center, and Pacific Northwest Laboratory) to begin implementing and evaluating authenticated ESnet services using the advanced Kerberos Version 5. The purpose of this project was to identify, understand, and resolve the technical, procedural, cultural, and policy issues surrounding peer-to-peer authentication in an inter-organization internet.

The investigators have concluded that, with certain conditions, Kerberos Version 5 is a suitable technology to enable ESnet users to freely share resources and information without compromising the integrity of their systems and data. The pilot project has demonstrated that Kerberos Version 5 is capable of supporting trusted third-party authentication across an inter-organization internet and that Kerberos Version 5 would be practical to implement across the ESnet community within the U.S. The investigators made several modifications to the Kerberos Version 5 system that are necessary for operation in the current Internet environment and have documented other technical shortcomings that must be addressed before large-scale deployment is attempted. The investigators also identified a number of administrative and implementation issues with recommended solutions or suggestions for further action.

Although it is unclear which of several competing network authentication technologies will gain general long-term acceptance, it is recommended that ESnet sites begin to deploy Kerberos Version 5 upon release of production code by the Massachusetts Institute of Technology. Kerberos Version 5 can immediately and effectively reduce exposure to a well-known and serious vulnerability that results from the transmission of clear-text passwords across the network.

ACKNOWLEDGMENTS

The authors wish to express their appreciation to the following individuals and organizations for their contributions to the ESnet Authentication Pilot Project: Ted Ts'o and Jeff Schiller, Massachusetts Institute of Technology; John Linn, Digital Equipment Corporation;[a] Cliff Neuman, University of Southern California; and Glenn Machin and Bill Wrahe, Sandia National Laboratory.

(a) John Linn is currently employed by OpenVision Technologies.

GLOSSARY

Active Attack — An attempt to improperly modify data, gain authentication, or gain authorization by inserting false packets into the data stream or by modifying packets transiting the data stream. (See passive attacks and replay attacks.)

Authentication — Verifying the claimed identity of a principal.

Authentication Path — A sequence of intermediate realms that are transited in the authentication process when communicating from one realm to another.

Authenticator — A record containing information that can be shown to have been recently generated using the session key known only by the client and server.

Authorization — The process of determining whether a client may use a service, which objects the client is allowed to access, and the type of access allowed for each object.

Cipher-text — Encrypted text.

Clear-text — Unencrypted text.

Client — A process that makes use of a network service on behalf of a user. Note that in some cases a server may be a client of some other server (e.g., a print server may be a client of a file server).

Credentials — A ticket plus the secret session key necessary to successfully use that ticket in an authentication exchange.

Data Encryption Standard (DES) — A secret-key encryption algorithm.

Key Distribution Center (KDC) — A network service that supplies tickets and temporary session keys or an instance of that service or the host on which it runs. The KDC services both initial ticket and ticket-granting ticket requests. The initial ticket portion is sometimes referred to as the **Authentication Server** (or **Service**). The ticket-granting ticket portion is sometimes referred to as the **Ticket-Granting Server** (or **Service**).

Passive Attack — An attack on an authentication system that inserts no data into the stream but instead relies on being able to passively monitor information being sent between other parties. This information could be used at a later time in what appears to be a valid session. (See active attack and replay attack)

Principal — A basic entity that participates in network authentication exchanges. A principal usually represents a user or the instance of a network service on a particular host. Each principal is uniquely named by its **Principal Identifier**.

Public-Key Cryptography	An encryption system that uses different keys for encryption and decryption. The two keys have an intrinsic mathematical relationship to each other. Also called Asymmetric Cryptography. (See secret-key cryptography.)
Realm	An autonomously administered Key Distribution Center (KDC) and the logical group of clients and servers registered to that KDC. Each realm has a unique **Realm Name**.
Replay Attack	An attack on an authentication system by recording and replaying previously sent valid messages (or parts of messages). Any constant authentication information, such as a password or electronically transmitted biometric data, can be recorded and used later to forge messages that appear to be authentic.
RSA Algorithm	A public-key encryption system named for its inventors: Rivest, Shamir, and Adleman.
Secret-Key Cryptography	An encryption system that uses the same key for encryption and decryption. Also called Symmetric Cryptography. (See public-key cryptography.)
Server	A particular principal that provides a resource (service) to network clients.
Session Key	A temporary encryption key used between two principals, with a lifetime limited to the duration of a single login "session."
Ticket	A record that helps a client authenticate itself to a server. A Kerberos ticket contains the client's identity, a session key, a timestamp, and other information, all encrypted using the server's secret key. It only serves to authenticate a client when presented along with a fresh Authenticator.
Ticket-Granting Ticket	A ticket that a client uses to authenticate itself to a ticket-granting server. A Ticket-Granting Ticket (TGT) to the local Ticket-Granting Server (TGS) is issued to the client during the initial authentication to Kerberos. TGTs for remote Ticket-Granting Servers may be issued to the client during cross-realm authentication.

CONTENTS

FIGURES

TABLES

1.0 INTRODUCTION

To conduct their work, U.S. Department of Energy (DOE) researchers require access to a wide range of computing systems and information resources outside of their respective laboratories. Electronically communicating with peers using the global Internet has become a necessity to effective collaboration with university, industrial, and other government partners. DOE's Energy Sciences Network (ESnet) needs to be engineered to facilitate this "collaboratory" while ensuring the protection of government computing resources from unauthorized use. Sensitive information and intellectual properties must be protected from unauthorized disclosure, modification, or destruction.

The vulnerability of information that is accessible from or transmitted across the Internet is well-documented (Stoll 1990; Bellovin 1992, 1993; CERT 1994; Cheswick and Bellovin 1994). Perhaps the most significant threat is the capability of any networked workstation to eavesdrop on network traffic. This could lead to the capture and exploitation of user authentication information. For example, the rash of computer break-ins at some ESnet sites during December 1993 occurred when perpetrators used valid system passwords obtained by network eavesdropping.

Various mechanisms to reduce this type of threat have been developed, but the commonly used techniques of network segmentation and one-time passwords are user-obtrusive and become barriers not only to unauthorized users but also to authorized users.

One approach to unobtrusively protecting against eavesdropping is the use of a network authentication protocol, such as the Kerberos system developed at the Massachusetts Institute of Technology (MIT). This technology has been in use for 5 or more years by a variety of government, educational, and commercial institutions to authenticate user access to computer and information resources within the confines of a local area network managed by a single organizational entity. However, the problem of authentication between organizations on a wide area network, such as ESnet, is just now being explored.

In March 1992, the ESnet Site Coordinating Committee (ESCC) chartered a network authentication task force to "plan and conduct an Internet authentication pilot project" in order to "identify, understand and resolve the technical, procedural, cultural and policy issues surrounding peer-to-peer authentication in an inter-organization internet."

The task force first met in May 1992 at MIT to review alternatives, define the scope and goals of the pilot project, and outline the project plan. Jeff Schiller, a principal developer of the MIT Kerberos network authentication system, and John Linn from the Digital Equipment Corporation Secure Systems Group attended the meeting to brief the group on the status and direction of network authentication technology. Discussed were Kerberos Versions 4 and 5, the Open Software Foundation (OSF) Distributed Computing Environment (DCE) Security Service, the draft Internet Engineering Task Force (IETF) Distributed Authentication Security Service submitted by Digital Equipment, and IETF work in progress on a new Internet Authentication Service (IETF-IAS). As described by Jeff Schiller, the new IETF-IAS promised to resolve major deficiencies or weaknesses found in all of the other alternatives.

Based on the presentations by Schiller and Linn, the task force concluded that the technology was in a very rapid state of flux and that a decision to choose a specific technical solution for an inter-site authentication pilot would be premature. The task force recommended that sites should experiment individually with Kerberos Version 4 until early 1993 to gain experience. At that point, the group would meet again and review the situation.

By March 1993, the task force had concluded that Kerberos Version 4 was not a viable solution for internet authentication because of two deficiencies: problematic key management requiring each site to share a pair of secrets with each site and missing support for simultaneous authentication to multiple

sites. Furthermore, IETF work on the new authentication standard had been stalled because of the demise of Digital's Secure Systems Group, which had been a principal sponsor of the IETF work. However, developers at MIT had made considerable progress stabilizing Kerberos Version 5.

In August 1993, DOE, Office of Energy Research, funded four ESnet sites (Argonne National Laboratory, Lawrence Livermore National Laboratory, the National Energy Research Supercomputer Center, and Pacific Northwest Laboratory) to begin implementing and evaluating authenticated ESnet services using the advanced Kerberos Version 5 authentication system. The project was broken into two phases. Phase I was to implement the baseline authentication infrastructure to be used for further study. Specific activities planned for Phase I were the following.

- Implementing a Kerberos Version 5 infrastructure consisting of local Key Distribution Centers (KDCs) at each of the four participating sites, plus a "root" ESnet KDC to arbitrate inter-site authentication requests.

- Implementing Kerberos-authenticated versions of the more commonly used network services, such as telnet, FTP, and the Berkeley 'r' commands.

Phase II of the project was using this baseline infrastructure to determine if Kerberos Version 5 was a suitable technology to meet requirements of the ESnet community. Specific activities planned for Phase II were the following.

- Investigating and recommending a strategy for integrating Kerberos Version 5 with other commonly used distributed security environments, such as Kerberos Version 4, the Andrew File System security mechanism, and the OSF/DCE security service.

- Identifying and developing the procedures and tools necessary for administering user accounts and authentication services between ESnet sites.

- Evangelizing researchers at the participating sites and assisting in "Kerberizing" one or more distributed computing applications.

Implementing individual site and ESnet "root" KDCs and testing intra-site authentication were accomplished in late fiscal year 1993. However, it was not until March 1994, following considerable troubleshooting of the beta software distributed by MIT, that inter-site authentication was successfully demonstrated across ESnet.

Inter-site authentication tests continued through the remainder of the project with numerous "bug" fixes being reported to the MIT developers for incorporation into the publicly available release of the software. An extension to the standard Kerberos inter-realm path-finding mechanism was developed by members of the project team to allow predefined authentication paths between the ESnet KDC hierarchy and other KDC hierarchies and to provide greater flexibility in realm naming. MIT-distributed versions of Kerberized Berkeley 'r' commands and the telnet program have been ported to a variety of operating systems. Other applications not distributed by MIT, including FTP, NCSA Mosaic, and the LLNL XDIR network file management program, have been Kerberized as part of the pilot project.

This report presents the findings of the ESnet Authentication Pilot Project. Section 2 of the report contains a tutorial on network authentication. The tutorial is helpful in understanding the issues described later in the report. Sections 3 through 7 describe findings of the project in the following areas:

- Section 3 - Cross-Realm Authentication

- Section 4 - Applications

- Section 5 - Transition and Inter-Operability Issues

- Section 6 - Technical Problems and Limitations

- Section 7 - Administration Issues.

Finally, conclusions and recommendations are summarized in Section 8.

2.0 NETWORK AUTHENTICATION TUTORIAL

This section presents a tutorial on network authentication using Kerberos as an example. The tutorial helps in understanding the work and issues that are described later in this report.

2.1 THE KERBEROS AUTHENTICATION SYSTEM

Kerberos is a protocol and system that uses electronic *tickets* to authenticate a client to a server without the risk of exposing authentication secrets (such as user passwords) to eavesdroppers. The Kerberos authentication protocol, illustrated in Figure 1, generally works as described below.

1. To obtain a ticket to a particular server, the client first sends a message to the Authentication Server (AS) requesting a ticket to the Ticket-Granting Server (TGS). The AS and TGS functions are typically collocated on a single system, which is popularly referred to as a Key Distribution Center (KDC).

2. The AS generates a temporary *session key*, which is encrypted using the client's secret key (a one-way hash of the client/user's password). The AS also generates a *ticket-granting ticket* for the client to present to the TGS. The ticket-granting ticket, which contains the client's identity and a copy of the session key, is encrypted using the TGS's secret key (which is known only to the AS and the TGS). The AS sends both of these encrypted messages back to the client.

3. Using its secret key (a one-way hash of the user's password which is computed by the client after prompting the user for a password), the client deciphers the session key sent by the AS. It then sends a message to the TGS requesting a ticket to the target server. This request contains the name of the target server, the ticket-granting ticket received from the AS (which is already encrypted in the TGS's secret key), and an *authenticator* encrypted using the session key generated and sent by the AS.

4. The TGS deciphers the ticket-granting ticket using its own secret key. The TGS then uses the session key contained in the ticket-granting ticket to decipher the authenticator. Information contained in the authenticator is used to validate the request and to protect against replay attacks. The TGS then generates a new session key for the client and target server and incorporates this into a ticket for the client to present to the target server. This ticket is encrypted using the target server's secret key, which is known only to the TGS and the target server. The TGS also encrypts the new client-target session key using the session key that it already shares with the client. Both of these encrypted messages are sent back to the client.

5. The client forwards the ticket to the target server, along with a new authenticator that is encrypted using the TGS-generated session key. The target server deciphers the ticket using its secret key, which is known only to itself and the TGS. The session key contained in the ticket, which is now shared by the client and target server, is used to authenticate the client and may optionally be used to authenticate the server. It may also be used to encrypt further communication between the two parties.

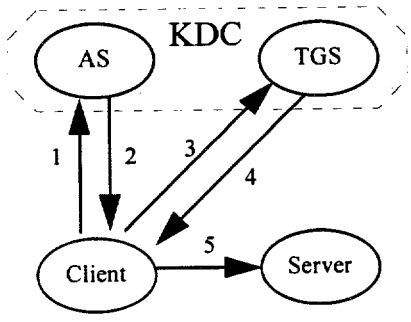

Figure 1. Kerberos Authentication Protocol

Once the client obtains a ticket-granting ticket (steps 1 and 2), it can use it to obtain a ticket for as many servers as needed without repeating this initial

authentication process and, conveniently, without the user having to re-enter a password. Similarly, once the client obtains a ticket for a particular server (steps 3 and 4), it can use it to authenticate its access to that server (step 5) as many times as necessary or desired until the ticket expires.

The Kerberos protocol is designed to operate across organizational boundaries. A client in one organization can be authenticated to a server in another. Each organization wanting to run a Kerberos server establishes its own *realm*. The name of the realm where a client is registered is part of the client's name and can be used by the end-service to decide whether to honor a request.

By establishing *inter-realm* keys, the administrators of two realms can allow a client that is authenticated in the local realm to obtain tickets for servers in the remote realm. The exchange of inter-realm keys (a separate key is typically used for each direction) registers the TGS of each realm as a principal in the other realm. A client is then able to obtain a ticket-granting ticket for the remote realm's TGS from its local realm. Cross-realm authentication, the primary interest of the ESnet authentication pilot, is described further in Section 3.

Kerberos was originally designed and implemented as a part of the MIT Project Athena (Miller et al. 1987; Steiner et al. 1988; Davis and Swick 1990). Versions 1 through 3 were experimental protocols used internal to the project. Kerberos Version 4 is publicly available from MIT and is widely used for authentication within a variety of organizations (Lunt 1990). Kerberos Version 5, described in RFC 1510 (Kohl and Neuman 1993), has evolved from Version 4 based on new requirements and known limitations of the previous version (Bellovin and Merritt 1990; Lunt 1990; Kohl 1991). A summary of the differences between Kerberos Versions 4 and 5 is shown in Table 1.

Table 1. Comparison of Kerberos Version 4 and Kerberos Version 5

Kerberos Version 4 – Limitations	Kerberos Version 5 – Improvements
Digital Encryption Standard (DES) dependence limits export.	Modular encryption code (DES independence).
IP dependence prohibits use with other network protocols.	Multiple network protocol/address support (IP independence).
"Receiver make it right" byte ordering.	ISO ASN.1 message encoding.
Principal naming restrictions.	ASN.1 "GeneralString" principal name field (compromise to resolving principal naming restrictions).
Limited (~21 hour) ticket lifetime. No "batch" processing support.	Unlimited ticket lifetimes. "Batch" support via "valid starting" time.
No authentication forwarding.	Supports authentication forwarding.
Flawed use of DES plain- and cipher-block-chaining (PCBC) mode.	Standard DES cipher-block-chaining (CBC) mode with embedded message digest.
Potentially flawed message digest.	Alternative (and presumably correct) message digest routines.
Potentially weak timestamp-based authenticators and replay detection.	Optional NONCE-based authenticators and replay detection.
Session key reuse creates replay vulnerability.	Subsession key negotiation mechanism reduces vulnerability.
Cannot simultaneously authenticate to multiple realms.	Supports simultaneous authentication to multiple realms.
$O(n^2)$ cross-realm key management problem.	KDC hierarchy reduces key management to $O(\log(n))$.
Vulnerable to offline password-guessing attack.	Ticket format changes reduce (but do not eliminate) vulnerability.

Not all previously documented problems and limitations of the Kerberos protocol have been resolved in the Version 5 specification. A few of the more significant limitations include the following.

- The initial exchange between a user/client and the AS is particularly subject to password-guessing attacks. (Recall that a portion of the information sent by the AS to the client is encrypted with a hash of the user's password.)

- Even with inter-realm authentication protocol improvements which greatly reduce the magnitude of the key management problem, secure distribution of secret keys between sites remains problematic.

- Inter-realm navigation issues are likewise unresolved. The specification is noncommittal, and the MIT reference implementation is based on an inflexible realm-naming convention that assumes the existence of a global KDC hierarchy having a single trusted "root" for ubiquitous cross-realm authentication.

These well-known limitations, along with additional problems and limitations with the MIT reference code used during the pilot (beta 5.4.2), are described further in Section 6.

2.2 ALTERNATIVE AUTHENTICATION SYSTEMS

Kerberos is by far the most well-known and widely deployed network authentication system, but it is not the only one. Two noteworthy alternatives to the Kerberos authentication system are discussed in this section.

2.2.1 OSF/DCE Security Service

The authentication component of the Open Software Foundation's DCE Security Service (OSF 1993) is based on an early alpha release of Kerberos Version 5. This service conforms to the basic Kerberos Version 5 protocol specification and thus shares the same weaknesses as Kerberos.

Although they share a common lineage, DCE authentication and Kerberos Version 5 do not support identical functionality, and so they do not inter-operate in all aspects. Furthermore, as the two systems have developed independently, minor incompatibilities have naturally resulted. Inter-operability of Kerberos Version 5 and the DCE Security Service is described further in Section 5.

2.2.2 Distributed Authentication Security Service

The Distributed Authentication Security Service (DASS) provides a comparable service to Kerberos, but it differs in its use of both public-key and secret-key cryptography whereas Kerberos uses secret-key exclusively.

DASS was developed by Digital Equipment Corporation and submitted as an Internet RFC (Kaufman 1993). A partial reference implementation known as SPX (Tardo and Alagappan 1991) is available in the public domain.

Like Kerberos Version 5, DASS incorporates hierarchically organized KDCs to reduce cross-realm key management to an O(log(n)) problem. As an improvement over Kerberos, though, DASS uses public-key cryptography to entirely eliminate the need to exchange secret keys between sites.

Another advantage of the using public-key cryptography in DASS is the reduced vulnerability to offline attack. And, unlike Kerberos, intrusion of the KDC by a perpetrator does not necessitate issuance of new user passwords.

Unfortunately, the benefits of using public-key cryptography come with a price. Public-key cryptography is 2 to 3 orders of magnitude more computational intensive than secret-key, resulting in considerably slower performance of DASS when compared to Kerberos.

Widespread acceptance of DASS as an alternative to Kerberos is highly unlikely because of the performance limitations of its extensively public-key-based protocol and its lack of inter-operability with either Kerberos or the DCE Security Service.

2.3 FUTURE OUTLOOK

The U.S. Department of Defense (DOD), Advanced Research Projects Agency (ARPA), has recently funded specification of a follow-on protocol to Kerberos Version 5 (presumably Version 6). Early reports are that the project will seek to resolve the realm navigation issue, possibly through an extension of the Internet Domain Name Service. The project will also seek a solution to the inter-realm key management problem, possibly using public-key cryptography as was done in the DASS. (The performance hit of public-key cryptography may be acceptable during the initial authentication of a client/user to the KDC or when first authenticating to another realm, given the advantages noted previously.)

Several OSF contributors are actively working to reduce the vulnerability of the DCE Security Service to password-guessing attacks. Methods proposed to date include third-party preauthentication (Pato 1993), and two-factor authentication using token cards (Kotanchik 1994) or smart cards (Merckling and Anderson 1994). Similar work to reduce the vulnerability of Kerberos is also in progress.

It is reasonable to expect that Kerberos and the OSF authentication protocols, given their common roots, could converge over time. Adopting Kerberos Version 5 as the near-term ESnet authentication standard is low risk and will actually provide a controlled migration path to the DCE Security Service if and when OSF/DCE becomes widely accepted.

3.0 CROSS-REALM AUTHENTICATION

Kerberos has been extensively deployed and used for authentication within a single realm. Its functionality and limitations for single-realm authentication have been documented elsewhere (Bellovin and Merritt 1990; Davis and Swick 1990; Lunt 1990). In contrast, the principal objective of the ESnet pilot was to demonstrate and assess cross-realm authentication in an inter-organization internet. This section describes the experiences and findings of the pilot in implementing and testing cross-realm authentication. It assumes that the reader is familiar with Kerberos or has read the protocol description in Section 2.1.

3.1 SIMPLE CROSS-REALM AUTHENTICATION

A client of a local realm can obtain a ticket for a server in a remote realm if the two realms have exchanged inter-realm keys. The client in the local realm uses its ticket-granting ticket (TGT) for the local realm to request a TGT for the remote realm. It then uses this new TGT to obtain a service ticket from the remote KDC.

Both KDCs define a principal of the form:

`krbtgt/<remote-realm>@<local-realm>`

having the same key and key version number. On the `<local-realm>` KDC, it is used much like a server principal to issue a service ticket. On the `<remote-realm>` KDC, it is used much like an entry in the service table; it holds the key to be used to authenticate the request.

For cross-realm authentication in the other direction, both KDCs define a principal of the form:

`krbtgt/<local-realm>@<remote-realm>`

having the same key. This key does not have to be the same as the one used in the other direction.

The administrators of the two realms need to coordinate the keys for these principals in their own KDC. The simplest method is to use the `kadmin` program (see Appendix A) to add the principal and generate a key using a password that has been exchanged through a secure channel.

3.2 CROSS-REALM AUTHENTICATION USING HIERARCHICAL KDCs

A realm can share keys with any number of other realms. Using the simple cross-realm authentication method used in Kerberos Version 4 and described above would require secure exchange and management of $O(n^2)$ key pairs between 'n' realms.[a] Key management using this method becomes problematic as the number of realms increases beyond a handful and becomes impossible in a national or global Internet. In the case of ESnet, nearly 1000 secret keys would need to be generated, securely transmitted, and periodically updated between the 32 current ESnet sites.

In Kerberos Version 5, just as a client can use a TGT for the local realm to obtain a TGT for a remote realm, it can use a remote TGT to get a TGT for yet another remote realm (see Figure 2).

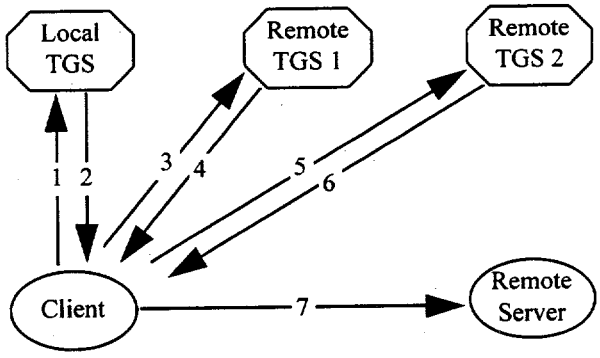

Figure 2. Cross-Realm Authentication

(a) Correctly, $n*(n-1)/2$ key pairs; $\sim n^2$ for large values of 'n'.

The Kerberos Version 5 specification, RFC 1510 (Kohl and Neuman 1993), states, "Realms are typically organized hierarchically. Each realm shares a key with its parent and a different key with each child." Having a single central, or *root*, KDC that shares keys with all the other KDCs reduces the number of keys from $O(n^2)$ key pairs to 'n' key pairs. Hierarchies of greater depth result in O(log(n)) key pairs. For ESnet, the number of secret keys that must be generated and securely exchanged is reduced from nearly 1000 to a mere 64.

The sequence of realms used in the authentication process is referred to as the *authentication path.* But how does a client determine the authentication path to a service in a remote realm? The Kerberos Version 5 specification states that "inter-operability across realm boundaries requires agreement on how realm names are to be assigned, and what information they imply." The current MIT Kerberos implementation uses a realm-naming convention patterned after the Internet Domain Name Service (DNS) to define a hierarchical authentication path.

For the ESnet pilot, the researchers initially chose realm names that would allow use of the MIT Kerberos method for defining the authentication path based on this hierarchical-naming convention (Figure 3). ES.NET was chosen as the top level, and it shared keys with four children: ANL.ES.NET, LLNL.ES.NET, NERSC.ES.NET, and PNL.ES.NET. At Argonne National Laboratory (ANL) two additional realms were defined for testing: ONE.ANL.ES.NET and TWO.ANL.ES.NET. Each KDC and realm were administered by its own organization.

The Kerberos `krb5_walk_realm_tree` routine returns a list of realms to be used for the authentication path that is based on the client and server realm names. For example, a client in the realm LLNL.ES.NET wants to connect to a server in TWO.ANL.ES.NET. Calling the `krb5_walk_realm_tree` routine with the client and server realm names returns the following list, which represents the credentials needed to get a ticket-granting ticket for the server's realm:

```
krbtgt/LLNL.ES.NET@LLNL.ES.NET
krbtgt/ES.NET@LLNL.ES.NET
krbtgt/ANL.ES.NET@ES.NET
krbtgt/TWO.ANL.ES.NET@ANL.ES.NET
```

The client then checks its credentials cache to see if it has any of these. It should at least have the first, which was obtained during its initial exchange with the AS and TGS. It may also have others because of previous requests being cached. The client then contacts the KDC of the realm "closest" to the server and for which it has a ticket-granting ticket. The contacted KDC generates the same list and will check its database to see if it can return either the next ticket-granting ticket requested by the client or one "closer" to the server. The client will then use the returned ticket-granting ticket to get the next ticket-granting ticket from the next KDC.

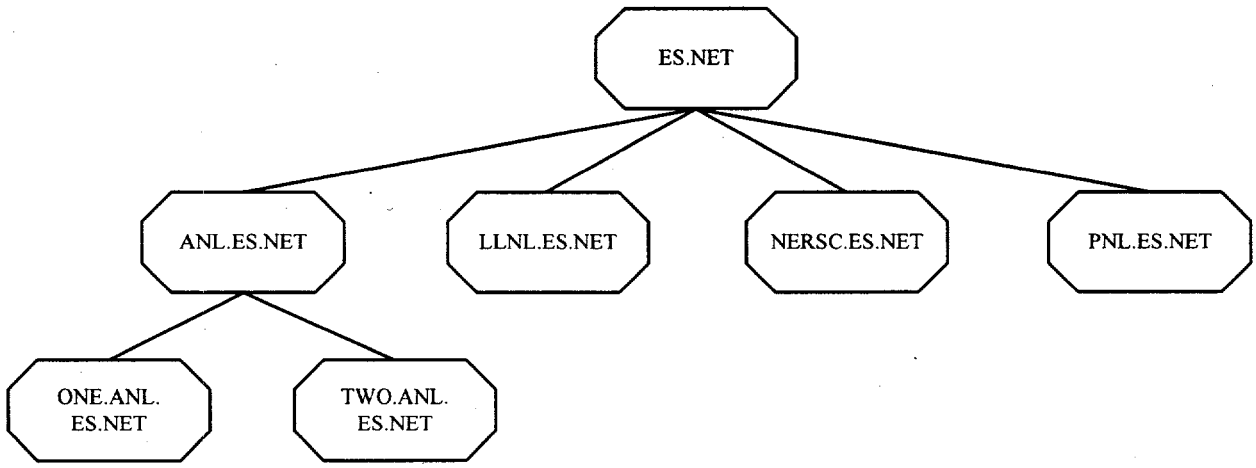

Figure 3. Initial ESnet Pilot Realm Configuration

Note that not all of the realms on the returned authentication path need to be involved. If a KDC along the path shares a key with a realm "closer" to the server, it can return a ticket for that realm. Each KDC uses the `krb5_walk_realm_tree` routine to generate the hierarchical authentication path and to check if it shares a key with a realm on the list that is closer to the server's realm. Needless to say, its realm must be on the list, and it must share a key with the next realm on the list if cross-realm authentication is to succeed. The *transited path* is defined as the list of the realms that were *actually* used to obtain the current ticket.

In one of the tests, a key was shared between ANL.ES.NET and PNL.ES.NET. A client at ONE.ANL.ES.NET wanting to connect to a server at PNL.ES.NET would generate the following list of credentials:

```
krbtgt/ONE.ANL.ES.NET@ONE.ANL.ES.NET
krbtgt/ANL.ES.NET@ONE.ANL.ES.NET
krbtgt/ES.NET@ANL.ES.NET
krbtgt/PNL.ES.NET@ES.NET
```

The client would then start to contact the KDCs and the ANL.ES.NET KDC would return the following:

```
krbtgt/PNL.ES.NET@ANL.ES.NET
```

because it is "closer" to the server. This would avoid the ES.NET KDC.

3.3 CONFIGURABLE AUTHENTICATION PATHS

Using realm names to define the authentication path allows for each client, KDC, and server to generate the same path. However, it also imposes an unrealistic constraint on the choice of realm names. All of the member organizations in the project are members of ESnet, but they are also members of other organizations. It is not realistic to impose the ES.NET name on each of their realm names.

A more acceptable choice for realm names is to use the well-established Internet domain names. Argonne National Laboratory wants to use a realm of ANL.GOV and the National Energy Research Supercomputer Center wants to use a realm of NERSC.GOV. But, to cross-realm authenticate between these two realms requires either shared keys between the two realms or shared keys with a parent realm of GOV. In the first case, as the number of organizations grows, it falls back to the $O(n^2)$ key management problem, which is what the researchers are trying to avoid. In the second case, who would run the GOV realm? What about PNL.GOV contacting the BATTELLE.ORG realm? Who would run the GOV and ORG realms? How would these two realms be connected (see Figure 4)?

This problem was resolved by modifying the standard MIT `krb5_walk_realm_tree` routine. The modifications allow parts of the hierarchical authentication path to be changed based on information stored in an additional configuration file.

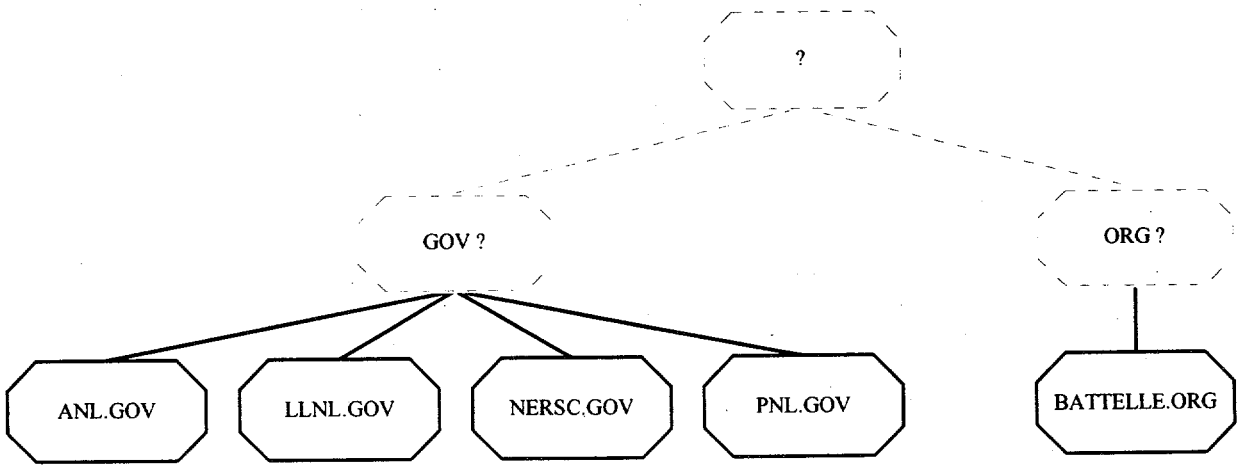

Figure 4. Problems of Realm-Name-Based Authentication Paths

The Kerberos specification (RFC 1510) allows for this by stating "If a hierarchical organization is not used, it may be necessary to consult some database in order to construct an authentication path between realms." The principal intent of the modification was to allow organizations to use their domain names for realms and contact parent KDCs, such as ES.NET, when authenticating with other ES.NET member organizations. The modified routine still allows hierarchical naming of subrealms within the organizations and falls back to the hierarchical authentication path if entries are not found in the configuration file.

The `krb5_walk_realm_tree` routine converts the realm hierarchy into an authentication path. It is called by both the client and the KDC when they need to get the authentication path. Modifications were made to this routine to allow parts of the hierarchical authentication path to be changed based on information stored in a configuration file. The configuration file, called `krb.capaths`, contains pairs of realm names, optionally followed by a list of up to five other realm names defining an authentication path between the pair of realms. Each pair of realm names and its list of other realms are on a separate line in the file.

The modified `krb5_walk_realm_tree` routine first generates a candidate authentication path list based on the hierarchical realm names. It then begins to scan the `krb.capaths` configuration file for alternate paths. If the pair of realm names contained on a line in the configuration file is found in the candidate authentication path list, the scan stops and the candidate list is modified by 1) deleting any realms that appear between the two matched realms and 2) inserting any realms listed in the configuration file for the matched pair. New realms are inserted in the correct order depending on which of the pair of realm names is "closer" to the server. Finally, the candidate authentication path list is converted into the list of principals and returned. Only one pass is made through the configuration file.

For example, ANL.GOV, LLNL.GOV, NERSC.GOV, and PNL.GOV all cross-authenticate by sharing keys with ES.NET. ANL.GOV also cross-authenticates directly with UCHICAGO.EDU, and with FNAL.GOV via HEP.NET. The `krb.capath` files for two of the realms, ES.NET and ANL.GOV, would look like the following:

ES.NET

```
ANL.GOV      LLNL.GOV      ES.NET
ANL.GOV      NERSC.GOV     ES.NET
ANL.GOV      PNL.GOV       ES.NET
ANL.GOV      ES.NET
LLNL.GOV     NERSC.GOV     ES.NET
LLNL.GOV     PNL.GOV       ES.NET
LLNL.GOV     ES.NET
NERSC.GOV    PNL.GOV       ES.NET
NERSC.GOV    ES.NET
PNL.GOV      ES.NET
```

ANL.GOV

```
ANL.GOV      LLNL.GOV      ES.NET
ANL.GOV      NERSC.GOV     ES.NET
ANL.GOV      PNL.GOV       ES.NET
ANL.GOV      ES.NET
ANL.GOV      UCHICAGO.EDU
ANL.GOV      FNAL.GOV      HEP.NET
```

Although these files may seem to add a greater degree of complexity, realms only need to define the entries that they need. They also need the realm's KDCs listed in the `krb.conf` files. Subrealms of these realms may use the same `krb.capaths` files.

Currently a `krb.capaths` file is needed by both the clients and the KDCs. Each only needs a subset of the possible authentication paths in which it is involved. The client's `krb.capaths` file only needs to define paths to the realms listed in the `krb.realms` file that cannot be derived using hierarchial names. The KDC's `krb.capaths` file only needs to define paths to the realms its clients might want to access and that cannot be derived from the hierarchical names.

The Configurable Authorization Path enhancement developed as part of the pilot has been submitted to MIT for possible inclusion into the publicly available distribution of Kerberos Version 5.

3.4 SCALABILITY OF CONFIGURABLE AUTHENTICATION PATHS

The Configurable Authentication Path modification to Kerberos that was described in the previous section can easily accommodate the current ESnet configuration of 32 sites, even if those organizations have subrealms of their own. It can even accommodate, for instance, the addition of a DOE.GOV "root" realm serving a few other DOE realms and sharing inter-realm keys with ES.NET. But, as the number of realms grows, the need for each intermediate KDC to have an entry in the `krb.capaths` for every authentication path increases as $O(n^2)$.

With additional changes to the client routines, the need for any `krb.capaths` file for the intermediate KDC's could be eliminated. These changes would be in `krb5_get_cred_from_kdc` and would eliminate much of the other "short cut" code that attempts to determine whether the KDC has a direct path to the server's realm. The `krb5_get_cred_from_kdc` routine could be modified to query each KDC along the way for a ticket-granting ticket to the next realm, rather than query each KDC for a ticket-granting ticket to the server's realm. Thus, only the client needs to know the authentication path to the server.

Consider, for example, a DOE.GOV realm acting as the parent to the ES.NET and other groups of realms. Under the present Configurable Authentication Path implementation, if a new realm is added under ES.NET, the `krb.capaths` files would need updating on the DOE.GOV and ES.NET KDCs and all site KDCs that intend to authenticate with the new realm. In the revised implementation described above, these KDCs would not need updating. Most client `krb.realms` and `krb.capaths` files would not need updating either. Only the files of those clients that need access to the new realm would need updating.

Improved scalability using the techniques described here is achievable today. The ideal, long-term solution to the Kerberos inter-realm key and table management scalability problem is using public keys for cross-realm authentication. Each client's KDC would use a name service to find the server's KDC's public key and address and could contact the server's KDC directly. It is understood that this is a design goal for Kerberos Version 6.

3.5 UNIVERSAL USER IDENTIFICATION

The ability to cross-authenticate defines by its nature the universal user identification, `<user>@<realm>`. Because the user name is unique within the realm and the realm names are derived from the unique DNS domain names, the combination is also unique. This user identification is currently usable in the `.k5login` file and can be used in AFS access control lists.

With the universal user identification in place, users only need a Kerberos principal in their home realm. Users do not need to be registered as a principal in other Kerberos realms but will still need authorization to access servers in other realms (i.e., login accounts must be established for the user).

4.0 APPLICATIONS

Applications (clients) and the programs providing service to them (servers) can be modified to use Kerberos for the initial authentication of users and to ensure the integrity and privacy of ongoing communications. The Kerberos libraries provide an API for each of these functions.

To do initial user authentication, a client must obtain the appropriate ticket and send it to the server. After establishing a network connection to a server, the client initializes the Kerberos package, obtains an appropriate ticket from the TGS, and sends the ticket to the server. This is accomplished with three library calls to get and check the ticket and two library calls to actually send the ticket. The client may optionally verify that a valid ticket is available using one other library call. All of this requires less than 100 lines of 'C' code and also requires that the client be loaded with the Kerberos libraries.

The server must also perform the appropriate Kerberos initialization and then receive the ticket. The ticket is then validated and the principal name mapped to a local user name (if necessary). Access to the local user's account is also checked. This also requires less than 100 lines of code and requires that the server be loaded with the Kerberos libraries.

The Kerberos libraries also provide routines for ensuring message integrity and/or privacy. With the use of appropriate calls, data may be transferred between the client and server with a cryptographic checksum or may be transferred encrypted.

Examples of "Kerberized" applications are described in the following sections. Appendix B describes in greater detail the modifications made to the File Transfer Protocol (FTP) and NCSA Mosaic.

4.1 BERKELEY 'r' COMMANDS

Kerberized versions of the Berkeley 'r' commands are distributed with the current MIT release of Kerberos Version 5. The client and server versions of these commands attempt to authenticate the user either within the local realm or by using the cross-realm mechanism.

The Berkeley 'r' commands, which come bundled with nearly all UNIX-based operating systems, have well-known vulnerabilities. The Kerberized versions of the 'r' commands eliminate the need of a `.rhosts` file, providing instead an authenticated login session by using the `.k5login` principal configuration file. A feature of the Kerberized 'r' commands is to fall back to the non-Kerberized version of the command when Kerberos authentication fails. This feature can be disabled by specifying appropriate configuration options to the daemon's reference within the `/etc/inetd.conf` file or to the service reference in the `/etc/services` file.

4.2 TELNET

Kerberized versions of the `telnet` client and server programs are distributed with the current release of MIT Kerberos Version 5. Unlike the Kerberized 'r' commands, which fall back on the non-Kerberized version when Kerberos authentication fails, `telnet` attempts to negotiate an authentication method (and possibly an encryption method) during the initial handshake with the server. Work is in progress to standardize how Kerberos and other authentication and encryption options should be handled by `telnet` (Borman 1993a, 1993b).

4.3 FILE TRANSFER PROTOCOL

The current MIT release of Kerberos Version 5 does not include a Kerberized FTP client program or server daemon. After a number of FTP implementations were reviewed, it was determined that adding Kerberos functionality to the Berkeley FTP would provide the greatest flexibility.

On the surface, it appears that adding telnet-like option negotiation to allow Kerberos authentication would be the preferred implementation path. Work is in progress to define how Kerberos and other

authentication and encryption options should be handled by FTP (Lunt 1994), but the draft protocol has not been implemented to our knowledge.

While reviewing FTP implementations, the researchers discovered that telnet-like option negotiation commands are often ignored even though they are accepted by most FTP daemons. Because one of the main goals was to maintain as much inter-operability as possible with existing FTP clients and servers, the researchers implemented Kerberos authentication in FTP using a new command, `TKT`. Details on the modifications made are included in Appendix B.

4.4 NCSA MOSAIC

The NCSA Mosaic client and `httpd` server daemon were both modified during the pilot to allow HTTP requests to use Kerberos Version 5 authentication. The Mosaic client was also modified to use Kerberos authentication when transferring files using FTP. See Appendix B for details.

4.5 XDIR

The LLNL XDIR network file manager uses FTP daemons to transfer files. A Kerberized version of XDIR has been implemented and made available to scientists working with the pilot for multi-realm data transfers. A description of XDIR can be found in Appendix C.

5.0 TRANSITION AND INTER-OPERABILITY ISSUES

One of the goals of the ESnet authentication pilot project was to investigate and recommend a strategy for integrating Kerberos Version 5 with other commonly used distributed security environments. The following sections describe the issues, options, and solutions that were explored for inter-operability with Kerberos Version 4, the Andrew File System (AFS), and the OSF/DCE Security Service.

5.1 KERBEROS VERSIONS 5 AND 4

When inter-operability between Kerberos Versions 5 and 4 is desired, three separate areas must be considered: the KDC, the server, and the client.

The Kerberos Version 5 KDC is capable of issuing both Version 5 and Version 4 tickets. In this mode, there is a single realm and each user and service has a single principal in the database. User entries must be added with Version 4 keys, and the `PREAUTH` flag must be off for any user or service principal that may authenticate using Version 4. (Because all Kerberos Version 4 requests for ticket-granting tickets are made in clear-text, they will fail if `PREAUTH` is on for the principal.)

Alternatively two separate KDCs can operate and appear as a single realm. An example of this is using the AFS `kaserver` as a Kerberos Version 4 KDC for current AFS users and having a Version 5 KDC that has a principal for the `afs@<realm>`. This allows Kerberos Version 5 users to obtain a Version 4 credential for AFS, converting the credentials into AFS tokens. The `afs@<realm>` entries in both KDCs must have the same key and key version number for this to work.

Servers may offer both Kerberos Version 5 and 4 services. This can be done by installing separate daemons for the different versions, each having a unique port number. Alternatively a single daemon can be designed to service requests for both versions. An example of the later method is the `rlogind` daemon provided with the MIT release. When compiled with the Kerberos Version 4 compatibility option, the daemon examines the version number of the request and takes the correct action.

In either case, the Kerberos Version 4 and Version 5 routines use their own `krb.conf` and `krb.realms` files. Since the format of these files is the same, it may be possible to share the same files using symbolic links. On the other hand, the `srvtab` files have different formats and cannot be shared. For example, the principal used for `rlogin` in Version 4 is `rcmd.<hostname>@<realm>` where `<hostname>` is not qualified. In Version 5, the principal is `host/<hostname>@<realm>` where `<hostname>` is the fully qualified domain name of the host.

At the client end, separate programs for each version of Kerberos may be built, or a single client program may be designed to negotiate with the server which version of Kerberos to use. An example of the former is `rlogin`. The Kerberized `telnet` program, on the other hand, has the capability of negotiating with the `telnetd` on the server which version of Kerberos to use, if any, and which encryption method to use for the session. Thus, one version of `telnet` can be used with any type of server. In either case, the appropriate `krb.conf` and `krb.realms` files are consulted.

At times, it may be necessary for a client to convert a Kerberos Version 5 credential into a Version 4 credential. The AFS `aklog` program is an example of this. Conversion of a credential requires that the ticket be deciphered, converted, and re-encrypted. This cannot be done by the client because it requires knowledge of the secret key of some principal other than itself. Instead, the client program calls the `krb524` library routines to contact a server daemon (normally running on the Kerberos Version 5 KDC), which completes the conversion on behalf of the client.

During the pilot, the Kerberos Version 4 clients bundled with Solaris 2.x were successfully used with a Kerberos Version 5 server, which had KDC

backward compatibility turned on. The krb524 routines and server daemon were used to obtain Kerberos Version 4 tickets that were given an initial Kerberos Version 5 ticket-granting ticket.

A problem was discovered with the Kerberos Version 5 replay detection mechanism while using NFS on a 64-node Meiko running Sol2. All requests to the KDC are placed in a replay cache along with the response. If a duplicate request is received *within* 1 second, the previous response is sent with no other processing taking place. A Kerberos Version 5 request contains the network addresses that may use the resultant tickets. For a Version 4 request, these network addresses are not in the request; they are determined by the address that the request was sent from. Thus, if a Kerberos Version 4 request arrives from one address for a ticket for service/<machine>@<realm> and another Version 4 request for the same ticket from a different address arrives within 1 second, it is determined to be a replay and the previous response is sent. Because the response has an incorrect address in the ticket for the second requestor, the ticket does not work. This can be solved either by disabling replay detection for Kerberos Version 4 requests or by keeping Version 4 requestor addresses in the replay cache.

When migrating from Kerberos Version 4 to Kerberos Version 5, administrators may want to convert the existing Version 4 database rather than create a new database. Sandia National Laboratory has included a utility in its Kerberos Version 5 (beta 2) release that attempts to accomplish this task. Unfortunately, this utility is not complete. With minor customizing, it can be used so long as the realm name being used with Kerberos Version 5 is the same as that used by Kerberos Version 4. If the realm name is changed, it is not possible to convert a Kerberos Version 4 database to a Kerberos Version 5 database. When a database is converted, clients must use the Kerberos Version 4 key-to-string translation to obtain the DES key used to decipher the ticket-granting ticket.

5.2 KERBEROS VERSION 5 AND AFS

AFS uses Kerberos Version 4 for its authentication. The klog command is used to authenticate and obtain a token. This token is then used with every file request to the AFS servers. The token is essentially a Kerberos Version 4 ticket. AFS also has a cross-realm authentication capability that allows foreign principals to be used on access control lists as <user>@<cell>.

MIT wrote a program called aklog, which uses a Kerberos Version 4 protocol to obtain a credential for afs@<realm>. It then converts this credential to a token and uses the AFS routines to store it.

The commonly used method of combining Kerberos Version 5 with AFS is using the Kerberos Version 5 KDC to return a Version 4 ticket-granting ticket and then using aklog to get the Version 4 credential for afs@<realm>. This has some drawbacks, including having to use the Kerberos Version 4 kinit and having to cache the Version 4 credentials. More important, it will not work with forwarded Kerberos Version 5 credentials or as part of the kadmin client and server programs.

For the pilot project, modifications were made to the aklog program to use the Kerberos Version 5 protocols, including using forwarded credentials to obtain a Version 5 credential. The krb524 routine is used to convert the Version 5 credential for afs@<realm> to a Version 4 credential. This is then converted to an AFS token.

The krb524 routines require a krb524d daemon running on a machine that has access to the key of the server that generated the Kerberos Version 5 credential. The daemon is usually run on the machine with the KDC and has access to the Kerberos database.

The modifications to aklog were done using the krb425 routines and library, which present the Kerberos Version 4 API but use the Version 5 protocols. krb425 was designed as a conversion aid. Having demonstrated that it is possible to use forwarded credentials to obtain an AFS token, aklog should be remodified to use the Kerberos

Version 5 API directly, thus eliminating the `krb425` routines and producing clearer error messages.

A modification was also made to the `kdb5_edit` program to allow a key and key version number to be set directly in the Kerberos Version 5 database. This is used to set the key of the `afs@<realm>` principal to be the same key and key version number that are used by the AFS servers. This modification should be rewritten as a separate subcommand of the `kdb5_edit` and `kadmin` programs.

Inter-operability with AFS requires the Kerberos Version 5 realm name to match the AFS cell name. AFS converts the upper-case realm name to a lower-case cell name. Because AFS cells typically follow the domain name convention, this is another reason for using the domain name as the realm name as suggested in Section 3.3.

In summary, the modified `aklog` program is used to obtain a Kerberos Version 5 credential for `afs@<realm>` using the Version 5 protocols, including cross-realm authentication. The Version 5 credential is then converted to a Version 4 credential with the assistance of the `krb524d` daemon. The Version 4 credential is then converted to an AFS token and stored using AFS routines.

5.3 KERBEROS VERSION 5 AND OSF/DCE SECURITY SERVICE

The authentication component of the DCE Security Service typically communicates with clients using DCE remote procedure calls (RPCs), but it is also capable of handling standard Kerberos requests and responding appropriately. Clients of Kerberos Version 5 can authenticate and get service tickets using a DCE Security Server in the same manner they would with a Kerberos Version 5 server, provided that the DCE authentication database contains the appropriate principals with the correct keys. This concept was successfully tested during the pilot with a reference version of the DCE Security Server and several different Kerberos Version 5 clients. No modifications were made to the clients for this to work.

Conversely, a Kerberos Version 5 server cannot support a DCE client because it does not communicate using DCE RPCs. Also, the DCE Security Service supports authorization using Privilege Attribute Certificates (PACs). The Kerberos Version 5 server does not support authorization using PACs. The Kerberos Version 5 server could communicate with an adjunct server to handle authorization using PACs, but no such server currently exists.

Another aspect limiting inter-operability is the incompatibility of the DCE Security Service and Kerberos Version 5 server administration tools. The DCE Security Service mechanism for remote administration uses DCE RPCs while Kerberos Version 5 administration uses Sun RPCs.

Finally, the MIT Kerberos Version 5 API is not officially supported in the DCE. DCE exports its own DCE Security-specific API. DCE support for the Generic Security Service API (Linn 1993a, 1993b, and 1994; Wray 1993) would be necessary if compile-time portability is a requirement.

6.0 TECHNICAL PROBLEMS AND LIMITATIONS

The following sections document a number of technical problems and limitations that still exist in Kerberos Version 5. Potential solutions to most of the shortcomings are also identified.

Some of the problems, such as the vulnerability of Kerberos to off-line password-guessing attacks, are well-known and previously documented. They are mentioned again in this report because of their importance in making an informed decision concerning the acceptability of Kerberos as an ESnet network authentication solution. Other problems, such as missing validation of the authentication path, are serious vulnerabilities that have been introduced as a result of new Kerberos functionality added in Version 5. As far as the researchers know, they have not been formally documented previously. Finally, this section discusses a number of items, such as deficient KDC administration tools, which are implementation issues specific to the MIT reference code used during the pilot (beta 5.4.2) and not to the basic Kerberos Version 5 protocols.

6.1 VALIDATION OF THE AUTHENTICATION PATH

The list of all realms that were involved in the authentication process is called the *transited path.* As each KDC grants a ticket-granting ticket based on a cross-realm request, it adds the realm name of the remote realm to the transited path. The client's and server's realms are not included in the transited path field because they are implied. The transited field for authentication within a single realm or across two realms will thus be null. Only when there is a third realm involved is anything listed.

The Kerberos specification (RFC 1510) states "It is important for the end-service to know which realms were transited when deciding how much faith to place in the authentication process." The release of Kerberos used during the pilot project (beta 5.4.2) did not check the transited field in any of the applications. The implication of this is that unscrupulous KDC administrators could obtain and exploit tickets disguised as principals in *any* realm, not just their own.

For example, realms A, B, and C all share inter-realm keys with D. Seeking to access a service in realm C, the unscrupulous administrator of realm A could create a fraudulent ticket request that contains the identifier of a principal in realm B. This request is processed in turn by the administrator's own KDC (which has been configured to accept the request from the client masquerading as a principal in realm B); the KDC in realm D (which trusts A and thus provides a ticket-granting ticket to realm C); and the KDC in realm C (which accepts the ticket-granting ticket generated by D and returns the requested ticket to the masquerading client). Because the service in C is not checking the transited path field, it will accept the ticket, even though the transited path field would unexpectedly contain realm A.

It would appear that if the end-service or the end-service's KDC used the `krb5_walk_realm_tree` routine to obtain the expected authentication path, it could use this to validate the transited path. This means that only realms listed in the authentication path should appear in the transited path. In the preceding example, realm A would appear in the transited path, having been placed there by D. If the client was really in B and authenticating to C, then A should not be in the authentication path, and the request should be refused.[a]

6.2 PASSWORD GUESSING

The Kerberos specification (RFC 1510) admits that the protocol is subject to offline password-guessing attacks. Good password management is the only effective protection against this threat. The Kerberos KDC should be enhanced to enforce site-specific policies for password expiration, length, and

(a) Since the conclusion of the pilot, MIT has added code to beta release 5.4.3 that implements transit path validation using the method suggested here.

composition. Composition policy could include pattern checks, dictionary checks, or machine generation.

6.3 MANAGEMENT OF PASSWORD COMPROMISE

If a user's password has been compromised (i.e., become known to some other party either by exposure of the password or by being guessed by some password-cracking program), the password can be easily changed in the KDC so that any future authentication requests using the compromised password are disallowed. However, it is also essential to invalidate any requests based on a ticket (or ticket-granting ticket) that may have been previously granted using the compromised password.

One of the features of Kerberos Version 5 is the ability to issue tickets that must be revalidated by the KDC before they can be used at a later time. This allows the KDC to check a "hot-list" of stolen or suspect tickets so that their use can be disallowed. The "hot-list" mechanism has not yet been implemented in the MIT beta release of Kerberos Version 5.

Cross-realm authentication further complicates the password compromise problem. For example, if a user obtains a ticket-granting ticket for a remote realm, changing the password on the local realm or adding an entry to a hot-list on the local realm does not stop the use of the remote ticket-granting ticket in the remote realm. A mechanism is needed to notify other KDCs to add these tickets to their own hot-lists. There are two possible choices: either notify all other KDCs or notify only those KDCs that have been issued the suspect ticket-granting tickets. The former may become unmanageable as the number of realms grows. The latter requires logging of cross-realm ticket-granting ticket requests and processing of the log to find KDCs that need to be contacted.

Without support for hot-lists and without cross-realm updating of hot-lists, advanced Kerberos Version 5 features, such as renewable tickets, are rendered useless. Tickets will always be valid for the full lifetime, because even if the KDC is contacted, it will renew the ticket because it has no reason not to.

6.4 LOG AND ERROR MESSAGES

The messages produced by the KDC and server applications are inadequate for auditing and, in particular, searching for usage patterns that could help identify possible stolen tickets or compromised passwords.

Users tend to follow selected patterns, logging on from the same machines and using the same sets of servers. If passwords are compromised or tickets stolen, the usage pattern will most likely change. Requests for new ticket-granting tickets will come from different locations. Requests for tickets to servers that are not typically accessed may appear.

The messages produced by the KDC and servers must be generated with the above usage in mind and must be logged so that the information is easily processed in order to find unauthorized use patterns. One such logging scheme is having all the appropriate messages logged in the `syslog` of the KDC.

Cross-realm authentication complicates this, because remote KDCs will be issuing tickets and logging the requests in their own `syslog`. The remote KDC should send audit trail information to the user's local KDC for all requests it makes on behalf of the remote user. This would allow the local administrator to review the logs on the local KDC for all suspicious activity.

Kerberos Version 5 can function without these changes, but it is very difficult to identify and trace unauthorized use because many separate logs must be reviewed.

6.5 USER- VERSUS SESSION-BASED CREDENTIAL CACHE

Credentials are cached by default in `/tmp/krb5_<uid>` where `<uid>` is the user's

identification number. The file is owned by the user. The environment variable KRB5CCNAME can be set to point to an alternate location.

As applications start to take advantage of the forwardable, proxiable, and postdated ticket features of Kerberos Version 5, they must be able to use the KRB5CCNAME environment variable to set the location of the cache on a session-by-session basis. Both the Berkeley 'r' commands and `telnet` can forward credentials to be cached on the server. However, they currently do not set the environment variable and thus share the cache with all the user's processes, leading to problems if the user inadvertently destroys the cache in one session while it is still being used (or needed) by a different session. It is particularly important for batch-processing systems to use session caching because the user cannot predict when processing will start and when it is safe to destroy the cache.

Processes that start sessions for users, including `rlogind`, `telnetd`, and batch-processing systems, should define a session-cache based on a unique property of the session (such as the process identifier) and should set the KRB5CCNAME environment variable. They could then destroy the session-cache upon completion of the session. Users would still own the caches and could change the environment variable to point at any of their caches at their own risk.

6.6 SYNCHRONIZATION OF KEY VERSION NUMBERS

When a principal is first defined, the key version number is initialized to 1 and then incremented whenever the key is changed. This can cause a problem if one administrator makes a mistake when changing an inter-realm key and then corrects it by changing it again. Each change will increment the key version number. This will cause the key version number to differ by one between the two KDCs, and cross-realm authentication will fail.

The `kdb5_edit` and `kadmin` programs currently distributed by MIT do not allow the key version number to be set independently of the key itself. If key version numbers become unsynchronized, as in the preceding example, either the administrator on the "lagging" side must change the key again to increment the key version number or both administrators must delete and add the principal to reset the key version numbers. A better solution would be modifying the `kdb5_edit` and `kadmin` programs to allow the administrator to directly set the key version number.

6.7 INTER-REALM KEY MANAGEMENT

Even with the Version 5 improvements to the inter-realm authentication protocols greatly reducing the magnitude of the key-management problem, secure distribution of secret keys between sites remains problematic.

A manual method for exchanging inter-realm keys was used during the pilot. During a face-to-face meeting in Livermore, California, the ESnet KDC administrator generated and personally delivered a list of passwords for each of the site KDC administrators. Numbered entries on this list were then referenced via electronic mail or telephone to change the keys as needed.

Although this seems clumsy, only the site KDC administrators, not the users, were impacted. For a modest ESnet authentication hierarchy of 32 sites or less, this may be a workable near-term solution until more automated methods can be implemented.

DOD/ARPA has recently funded specification of a follow-on protocol to Kerberos Version 5 (presumably Version 6). Early reports are that the project will seek to resolve the inter-realm key management problem, possibly using public-key cryptography.

6.8 RENEWING EXPIRED TICKETS

Kerberos Version 5 allows for ticket lifetimes to be renewed. However, renewal requests are denied if the ticket has expired, even if the renewable lifetime has not expired. Because there is not an easy way to be told that a ticket is about to expire, a

mechanism to renew an expired but still renewable ticket is highly desirable. This is especially needed for batch-oriented systems that want to use Kerberos.

6.9 ENCRYPTION ALGORITHMS

Kerberos Version 5 comes from MIT with a DES encryption library. DES, introduced in 1975, is the current U.S. government standard for secret-key encryption (NBS 1977). Although it was the subject of controversy because its keys were characterized as too small and other weaknesses were suspected, DES has proved resistant to public attacks. Nevertheless, DES is coming to the end of its useful life with its key size and complexity being overtaken by improvements in speed and cost of computers (Landau 1994). With the arrival of more powerful processors and massively parallel machines, it is desirable to have other, more formidable encryption libraries available, such as triple-DES.

The Kerberos Version 5 specification (RFC 1510) is designed to allow the encryption algorithms to be changed, but it is still difficult to accomplish this using the MIT reference code. Modifications to Kerberos Version 5 to handle multiple or different encryption libraries in a manner similar to that used for credential caches is highly desirable.

Because strong cryptography for confidentiality purposes has the potential to interfere with foreign intelligence gathering, the U.S. government restricts export of encryption software. These restrictions apply to the DES libraries included in the MIT release of Kerberos Version 5. It is paradoxical that U.S. government restrictions on the export of cryptographic software hampers its ability to protect its own information and resources.

Alternatives to DES that might be considered for use by ESnet are the RC2 and RC4 variable-key-size encryption algorithms. Through an agreement between the Software Publishers Association and the U.S. government, export approval is simpler and quicker for products using RC2 and RC4. However, to qualify for quick approval, the product must limit key sizes to 40 bits. Limiting the key length effectively weakens the strength of the cipher by a factor of 2^{16} relative to DES, but techniques are available to help confound attacks. RC2 and RC4 are proprietary algorithms of RSA Data Security, Inc., and would need to be licensed for ESnet use. A waiver to use an encryption algorithm other than the U.S. government standard DES may also need to be obtained.

6.10 ADMINISTRATION TOOLS

Administering service tables using the tools provided with the standard MIT distribution is difficult, because either the server system administrators require access to the KDC or the site KDC administrator must assume this burden. Also, maintenance of service tables using the KDC introduces the potential for exposing service keys when the table is transferred from the KDC to the server machine.

The `ksrvutil` program developed at Sandia National Laboratory for Kerberos Version 5 (beta release 2) allows a system administrator to maintain the service table locally. Functions provided by `ksrvutil` include addition, modification and deletion of service keys, display of service table entries, and translation of service tables from Kerberos Version 4 to Version 5 and vice versa. Because the program operates on the local system, there is no need for access to the KDC to maintain the service table, and the potential for exposing service keys when transferring tables from the KDC to the server is eliminated.

The pilot project will submit an updated version of the `ksrvutil` utility to MIT for inclusion in their Kerberos Version 5 distribution. The updated source will also be placed on the ESnet NIC so that it will be available to ESnet sites before MIT includes it in their distribution.

6.11 KDC DATABASE PROPAGATION

The current MIT implementation for propagating the KDC database to slave servers is brute-force database copy. This should be redesigned to support the enterprise-wide and high-availability

environments that exist within the ESnet. Furthermore, updates to slave servers should happen in a timely manner to provide adequate service to users of the system.

6.12 X-TERMINALS

X-Terminals present another problem in that they currently do not have built-in Kerberos support. When users initiate a `telnet` session using an X-Terminal, their passwords are sent over the network in clear-text. This can be remedied by the X-Terminal vendors, because most can download the software to the terminal and they could add Kerberos support to their software.

This problem also applies to most X-Terminal emulators, which run on personal computers (e.g., IBM/compatible PCs and Macintosh). Users may not be aware of how their passwords are handled by these devices and/or systems. This can lead to compromised passwords.

6.13 DOCUMENTATION

The documentation included in the MIT Kerberos Version 5 software distribution is incomplete or inadequate in some areas. The build procedure is adequately documented, and there are man pages for many of the tools. Documentation is still needed for the following:

- KDC creation and initial operation
- configuration file formats
- Kerberos Version 5 API Library.

The man page files are also missing for the following executable files, which are installed by the build procedure:

```
gss-client          rlogin
gss-server          rsh
krlogind            sim_client
krshd               sim_server
login.krb5          uuclient
movemail            uuserver
rcp                 v4kadmind
```

7.0 ADMINISTRATIVE ISSUES

Solving the technical aspects of cross-realm authentication is only half the battle. Trust relationships between site KDCs can be programmed, but inter-site authentication will still rely on establishing trust relationships between the organizations and people who manage those KDCs.

Cross-realm authentication could potentially weaken security at some sites. For example, Pacific Northwest Laboratory (PNL) protects all dial-up access using the Security Dynamics SecurID token-card system. Such physical port protection of dial-up access to network-connected systems is specifically required in PNL's computer security policy. If PNL allows cross-realm authentication access from a site that does not have a similar dial-up protection system in place, PNL increases its exposure and violates its own policy. This paradox may be true for other differences in computer security policies that may exist between sites, such as differences in host password expiration, length, and composition.

Minimum computer security standards that are based on DOE policy, Internet site security guidelines (Holbrook and Reynolds 1991), and good management practices should be established for sites participating in cross-realm authentication. Particular attention must be given to the protection of KDCs. A KDC, if compromised, could be used to exploit not only the local site but other sites within the cross-realm environment of the ESnet.

Periodic security audits of site KDCs should be conducted, preferably with the assistance of an independent reviewer, to minimize risk and build trust between sites. Computer Incident Advisory Capability (CIAC) staff, for instance, could be contracted by the site KDC administrator to assist with an audit of the local KDC. CIAC staff located at Lawrence Livermore National Laboratory could also assist with review of the ESnet root KDC(s). Audits are intended to identify any potential security holes within the KDC. The audits should ensure the following.

- Remote login access and file transfer access to the KDC are disabled.
- The number of persons having logical or physical access to the KDC is minimized.
- The KDC master password is a minimum of 11 characters in length, is machine-generated, and is changed monthly.
- Principal passwords are a minimum of 8 characters in length and are changed at least quarterly.
- Log reports are generated and reviewed daily by the site KDC administrator.
- A KDC disaster recovery plan is in place.
- A KDC break-in recovery plan is in place.

Additional work is needed to establish written standards for managing site KDCs. Additional work is also needed to develop a "Memorandum of Understanding" template to document mutual agreements between sites to manage KDCs to the established standards. These standards should be developed cooperatively by ESnet and site management, affirmed by the ESnet Site Coordinating Committee, and approved by the ESnet Steering Committee.

8.0 CONCLUSIONS AND RECOMMENDATIONS

The investigators have concluded that, with certain conditions, Kerberos Version 5 is a suitable technology to enable ESnet users to freely share resources and information without compromising the integrity of their systems and data. The pilot project demonstrated that Kerberos Version 5 is capable of supporting trusted third-party authentication across an inter-organization internet and that Version 5 would be practical to implement across the ESnet community within the U.S. The investigators have made necessary modifications to the Kerberos Version 5 system in order to operate in the current Internet environment and have documented other technical shortcomings that must be addressed before attempting large-scale deployment. A number of administrative and implementation issues have been identified, along with recommended solutions or suggestions for further action.

Although it is unclear which of several competing network authentication technologies will gain ubiquitous long-term acceptance, it is recommended that ESnet sites begin to deploy Kerberos Version 5 upon release of production code by MIT. Kerberos Version 5 can immediately and effectively reduce exposure to well-known and serious vulnerabilities that result from the transmission of clear-text passwords across the network. The system can be implemented with very little impact to users. The system is inter-operable with currently used network applications, such as the Berkeley 'r' commands, telnet, and FTP. Long-term risks appear to be minimal because the principal competing technologies are also based on Kerberos Version 5.

Finally, it is recommended that DOE support the following work.

- Assist MIT researchers in completing development and testing of Kerberos Version 5, ensuring that modifications necessary for large-scale deployment across ESnet are incorporated.

- Establish minimum site security requirements for participation in an ESnet authentication service, including standards for managing individual site Kerberos services.

- Provide consultation to other ESnet sites that implement Kerberos, documenting lessons learned as new sites are added, and coordinating ESnet cross-realm configuration.

- Initiate a consortium of interested educational, commercial, and government organizations to promote the inclusion of Kerberos authentication in distributed applications and operating systems. Microsoft, Novell, and Oracle have expressed interest in participating in such a consortium but are unwilling to initiate its formation. Organizing an open meeting in cooperation with MIT may be sufficient to "break the ice."

9.0 REFERENCES

Bellovin, S. M. 1993. "Packets Found on an Internet." *ACM Computer Communications Review*, 23(3):26-31.

Bellovin, S. M., and Michael Merritt. 1990. "Limitations of the Kerberos Authentication System." *ACM Computer Communications Review*, 20(5):119-132.

Bellovin, S. M. 1992. "There Be Dragons." In *Proceedings of the 3rd Usenix UNIX Security Symposium*, Baltimore, Maryland.

Borman, D. 1993a. "Telnet Authentication Option." RFC-1409, DDN Network Information Center.

Borman, D. 1993b. "Telnet Authentication: Kerberos Version 4." RFC-1411, DDN Network Information Center.

Cheswick, W. R., and S. M. Bellovin. 1994. "An Evening with Berferd." Chapter 10 in *Firewalls & Internet Security*. Addison-Wesley, Reading, Massachusetts.

Computer Emergency Response Team (CERT). 1994. *Ongoing Network Monitoring Attacks*. CERT Advisory CA-94:01. Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Davis, D., and R. Swick. 1990. *Workstation Services and Kerberos Authentication at Project Athena*. Technical Memorandum TM-424, MIT Laboratory for Computer Science, Cambridge, Massachusetts.

Holbrook, P., and J. Reynolds. 1991. "Site Security Handbook." RFC-1244, DDN Network Information Center.

Kaufman, C. 1993. "Distributed Authentication Security Service (DASS)." RFC-1507, DDN Network Information Center.

Kohl, J. T. 1991. "The Evolution of the Kerberos Authentication Service." Presented at the Spring 1991 EurOpen Conference, Tromsø, Norway.

Kohl, J. T., and B. C. Neuman. 1993. "The Kerberos Network Authentication Service (V5)." RFC-1510, DDN Network Information Center.

Kotanchik, J. 1994. "Kerberos and Two-Factor Authentication." DCE RFC-59.0, Open Software Foundation.

Landau, S. 1994. "Crypto Policy Perspectives." *Communications of the ACM*, 37(8):115

Linn, J. 1993a. "Common Authentication Technology Overview." RFC-1511, DDN Network Information Center.

Linn, J. 1993b. "Generic Security Service Application Program Interface." RFC-1508, DDN Network Information Center.

Linn, J. 1994. "The Kerberos Version 5 GSS-API Mechanism." Internet Draft[a] available via anonymous FTP from: `nic.es.net:/pub/internet-drafts/draft-ietf-cat-kerb5gss-01.txt`

Lunt, S. J. 1990. "Experiences with Kerberos." In *Proceedings of the 2nd Usenix UNIX Security Symposium*, Portland, Oregon.

Lunt, S. J. 1994. "FTP Security Extensions." Internet Draft[a] available via anonymous FTP from: `nic.es.net:/pub/internet-drafts/draft-ietf-cat-ftpsec-05.txt`

Merckling, R., and A. Anderson. 1994. "DCE Smart Card Integration." DCE RFC-57.1, Open Software Foundation.

Miller, S. P., B. C. Neuman, J. I. Schiller, and J. H. Saltzer. 1987. *Section E.2.1: Kerberos Authentication and Authorization System*. MIT Project Athena, Cambridge, Massachusetts.

National Bureau of Standards (NBS). 1977. "Data Encryption Standard." Federal Information Processing Standards Publication 46. Government Printing Office, Washington, D.C.

Open Software Foundation (OSF). 1993. "Authentication." Chapter 40 in *OSF DCE Application Development Guide, Part 6: DCE Security Service*. Prentice Hall, Englewood Cliffs, New Jersey.

Pato, J. 1993. "Using Pre-Authentication to Avoid Password Guessing Attacks." DCE RFC-26.0, Open Software Foundation.

Steiner, J., C. Neuman, and J. I. Schiller. 1988. "Kerberos: An Authentication Service for Open Network Systems." In *Usenix Conference Proceedings*, pp. 191-202. February 1988, Dallas, Texas.

Stoll, C. 1990. *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*, Pocket Books, New York, New York.

Tardo, J. J., and K. Alagappan. 1991. "SPX: Global Authentication Using Public Key Certificates." In *Proceedings of the 1991 Symposium on Research in Security & Privacy*, pp. 232-244. IEEE Computer Society. Los Amitos, California.

Wray, J. 1993. "Generic Security Service Application Program Interface: C-bindings." RFC-1509, DDN Network Information Center.

(a) Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Internet Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time.

APPENDIX A

ESNET IMPLEMENTATION GUIDELINES FOR KERBEROS VERSION 5

APPENDIX A

ESNET IMPLEMENTATION GUIDELINES FOR KERBEROS VERSION 5

A.1 HARDWARE AND SOFTWARE REQUIREMENTS

The capacity requirements for a site KDC system are, of course, dependent upon the size and activity of the site's distributed computing environment. A typical KDC machine within the ESnet, however, should be on the order of a SPARCstation 5 with 32MB memory. 1GB of available disk space is adequate for several thousand principals in the KDC database. Slave servers should have similar configurations. The KDC should include local file backup facilities.

Slave KDC servers can be used to increase reliability and availability of authentication services. However, KDC database propagation in the current MIT release of Kerberos Version 5 is a brute-force, database copy approach that can be cumbersome to administer.

Intrusion of a KDC is catastrophic; all services that trust the KDC are subject to misuse until the intrusion is detected and tickets revoked, and all user passwords must be reissued. Hence, the KDC machine must be set up and configured in a very secure manner (this also applies to any KDC slave servers). Some guidelines to follow are listed below.

- Place the machine in a secure area, such as a locked cabinet within a computer center that has limited physical access.
- Disable network access other than what is required for Kerberos. This includes disabling `telnet`, `rlogin`, `rsh`, `rcp`, and `rcmd`.
- Limit user account access to the machine.
- Where possible, use machine-generated passwords at least 11 characters long for KDC user accounts and the KDC master passwords.
- Limit the functionality of the machine to serving only Kerberos and similar sensitive applications.

A.2 INSTALLING THE KERBEROS SOFTWARE

The ESnet Kerberos Version 5 distribution, which the pilot group has been working with, is the same as the MIT distribution with only a few modifications. These modifications have been supplied to MIT for inclusion in their distribution. To install the ESnet distribution, obtain the `README` file via anonymous FTP from `nic.es.net:/pub/public-domain/kerberos5/README`. The `README` file explains the steps required to download, install, and configure the ESnet Kerberos distribution.

A.3 CONFIGURING THE SITE KDC

This section presents some basic guidelines and steps for ESnet site KDC administrators to follow when configuring their KDCs. Administrators should also refer to the documentation included in the Kerberos Version 5 software distribution.

A.3.1 Choosing a Realm Name

The choice of realm names can have an impact on cross-realm authentication and inter-operability with AFS and previous versions of Kerberos. ESnet sites should use their DNS domain name(s) for their realm name(s) where possible. This is consistent with previous ESnet recommendations for naming of AFS cells and ensures that realm names are unique. With the proper choice of realm names, your Universal Login ID is your Kerberos principal, `<user>@<realm>`. This can be used with the `.k5login` file and with AFS cross-realm authentication.

Realm names should be registered with the ESnet coordinator by sending electronic mail to `k5admin@es.net`. Include the following information in the body of the message:

- site name
- realm name
- KDC host name
- KDC administrator contact (full name and telephone number).

Example:

```
Argonne National Laboratory
ANL.GOV
douglas.ctd.anl.gov
Doug Engert, (708)252-5444
```

Registering realm names with the ESnet coordinator will ensure that your site is included in the global ESnet configuration files.

A.3.2 Creating the KDC

To create the KDC for your site, perform the following steps.

1. After installing the Kerberos Version 5 software, change directory into the root directory where Kerberos Version 5 was installed.

2. Obtain all the configuration files for ESnet via anonymous FTP from the `nic.es.net:/pub/public-domain/kerberos5/config-files` directory. These configuration files are required to create the initial KDC database file.

3. Once the configuration files are transferred, your realm name must be inserted as the first entry in the `krb.conf` file above all other entries.

4. Change directory into the `admin` directory under the root directory where Kerberos Version 5 was installed.

5. Execute the `kdb5_create` command and provide the master password. The master password should be machine-generated and at least 11 characters in length. This password must be remembered in order to modify the database.

6. Add the following lines to the `/etc/services` file:

```
kerberos5         88/udp   kerberosV      krb5_kdc      # Kerberos V5 server
kerberos5         88/tcp   kerberosV      krb5_kdc      # Kerberos V5 server
kerberos_admin    89/udp   kerberos-adm   krb5_kadmind  # Kerberos V5 admin
kerberos_admin    89/tcp   kerberos-adm   krb5_kadmind  # Kerberos V5 admin
kerberos          750/udp  kdc            kerberos4     # Kerberos V4 server
kerberos          750/tcp  kdc            kerberos4     # Kerberos V4 server
kerberos_master   751/udp                               # Kerberos V5 master
kerberos_master   751/tcp                               # Kerberos V5 master
kerberos-sec      758/udp                               # Kerberos V5
kerberos-sec      758/tcp                               # Kerberos V5
```

Note: The exact contents of this file are site-dependent.

7. Start the Kerberos Version 5 server by executing `krb5kdc &` command located in the `sbin` directory of the root directory where Kerberos Version 5 was installed.

A.3.3 Stashing the KDC Password

If you want the Kerberos KDC server to automatically start when the KDC machine reboots, the KDC master password must be stored (stashed) on the system in a protected file. This can be done by executing the `kdb5_stash` command located in the root directory where Kerberos Version 5 was installed. The master password will be stored in the `/.k5.REALM-NAME` file.

A.3.4 Modifying the KDC Database

Maintaining the KDC database (adding, modifying, and deleting principals) can be done directly on the KDC server using the `kdb5_edit` utility. However, using `kdb5_edit` is difficult, because login access to the KDC machine *must* be done only from the system console to preserve security and KDC integrity.

An alternative to using `kdb5_edit` is the `kadmin` tool. `kadmin` communicates with the `kadmind` daemon, which runs on the KDC machine and can be used from any of the workstations within the KDC's realm to modify the KDC database.

For administrators to use `kadmin`, they must add a principal of the form `<user>/admin@<realm>` into the KDC database, where `<user>` is the user name to use from the remote workstation and `<realm>` is the realm name of the KDC to update (the local realm is the default). Also, the `kadmind` process must be running on the KDC to interface with the `kadmin` utility.

To initiate the connection from a workstation, type `kadmin -n <user>/admin` and provide the password for the `<user>/admin` principal. For example:

```
# /krb5/bin/kadmin -n fred/admin
Password for fred/admin@pnl.gov:
```

The `kadmin` tool accepts the following commands (among others):

`add <principal>`	Add a principal to the KDC.
`del <principal>`	Remove a principal from the KDC.
`inq <principal>`	Display the configurable attributes assigned to the principal.
`mod <principal>`	Modify one or more of the attributes assigned to the principal.
`cpw <principal>`	Change the password (key) for a principal.

Some common forms of `<principal>` include the following:

`<user>`	A user principal.
`host/<machine-name>`	A server principal.
`krbtgt/<realm-a>@<realm-b>`	An inter-realm principal.

The following example shows how to use `kadmin` to grant the "joe" principal the capability to forward tickets.

1. Initiate `kadmin` for administrator "fred."

```
# /krb5/bin/kadmin -n fred/admin
Password for fred/admin@pnl.gov:
```

2. Inquire the current attribute settings for "joe." Note the NOFOR attribute, which indicates that "joe" cannot forward tickets.

```
Command (add, cpw, del, inq, mod, addrnd, cpwrnd, addv4, cpwv4, q): inq joe

Principal: joe@pnl.gov

Maximum Ticket Lifetime (MTL) = 86400 (seconds)
Maximum Renewal Lifetime (MRL) = 604800 (seconds)
Principal Key Version (PKV) = 3
Principal Expiration Date (PED): 2037/12/30:16:00:00
Last Modification Date (LMD): 1994/11/17:21:19:15
Principal Attributes (PA): POST NOFOR TGT REN PROXY DUPSKEY UNLOCKED SVR
Principal Salt Type (PST) = Version 5 Normal
```

3. Use the `mod` command to change the `NOFOR` attribute to `FOR`. Typing a "?" at the `Attribute:` prompt will display the configurable attributes.

```
Command (add, cpw, del, inq, mod, addrnd, cpwrnd, addv4, cpwv4, q): mod joe
Parameter Type to be Modified (vno, attr, or q): attr

Attribute: ?

Valid Responses are:
post/nopost - Allow/Disallow postdating
forward/noforward - Allow/Disallow forwarding
tgt/notgt - Allow/Disallow initial tickets
ren/noren - Allow/Disallow renewable tickets
proxy/noproxy - Allow/Disallow proxiable tickets
dskey/nodskey - Allow/Disallow Duplicate Session Keys
lock/unlock - Lock/Unlock client
svr/nosvr - Allow/Disallow Use of Principal as Server
q - Quit from setting attributes.

Attribute: forward
```

4. An inquire of the current attribute settings for "joe" now shows the FOR attribute indicating that "joe" can forward tickets.

```
Command (add, cpw, del, inq, mod, addrnd, cpwrnd, addv4, cpwv4, q): inq joe

Principal: joe@pnl.gov

Maximum Ticket Lifetime (MTL) = 86400 (seconds)
Maximum Renewal Lifetime (MRL) = 604800 (seconds)
Principal Key Version (PKV) = 3
Principal Expiration Date (PED): 2037/12/30:16:00:00
Last Modification Date (LMD): 1994/11/17:21:19:33
Principal Attributes (PA): POST FOR TGT REN PROXY DUPSKEY UNLOCKED SVR
Principal Salt Type (PST) = Version 5 Normal
```

APPENDIX B

EXAMPLES OF "KERBERIZING" APPLICATIONS

APPENDIX B

EXAMPLES OF "KERBERIZING" APPLICATIONS

The following sections describe the modifications made by the researchers to add Kerberos Version 5 authentication to File Transfer Protocol (FTP) and NCSA Mosaic.

B.1 File Transfer Protocol

Kerberos authentication in FTP was implemented using a new protocol command, `TKT`. The Berkeley `ftp` client program and `ftpd` server daemon were modified to process the new command.

B.1.1 Changes to `ftp`

The changes involve only two routines: the `user()` command routine in `cmds.c` and the `login()` routine in `ftp.c`. Two new routines were added to `ftp.c`: `get_k5cred()`, and `send_k5cred()`.

Code was inserted into `user()` and `login()` so that, following a successful FTP `USER` command, either a Kerberos ticket or a password can be sent. The code first invokes `get_k5cred()`, which checks to see if client credentials exist. If client credentials exist, a `TKT` command is sent to the server daemon. If no client credentials exist, no `TKT` command is issued and the normal login proceeds with an appropriate `PASS` command. Similarly, if the `TKT` command does not result in a `CONTINUE` reply from the server, it is assumed that the `ftpd` daemon is not Kerberized and the normal login proceeds with an appropriate `PASS` command. If, on the other hand, the `TKT` command is accepted by the server, `send_k5cred()` is invoked. `send_k5cred()` invokes the Kerberos `sendauth` library routine and returns the reply message received from the server using the Kerberos `getreply()` routine. If `send_k5cred()` returns `COMPLETE`, the `user()` or `login()` routine returns a success status of 1. Otherwise, an error message is issued and a failure status of 0 is returned.

Anonymous FTP access is allowed using the current convention of specifying `ftp` or `anonymous` in the `USER` command.

B.1.2 Changes to `ftpd`

The changes involve only two source files: `ftpcmd.y` (a `TKT` command was added) and `ftpd.c` (the `tkt()` routine was added).

The changes for the daemon are encapsulated in the new routine `tkt()`, which is invoked when a `TKT` command is received. The `tkt()` routine attempts to do a `recvauth` and returns an appropriate error reply if this fails. Next, it does the normal processing for setting the group identifier and user identifier for the indicated user, fixing the login directory and guest privileges as needed. If any errors are encountered in this process, an error return results (`pw` returned is NULL and effective `uid` is set to 0). Otherwise, an appropriate (guest or user) reply is sent to the client, and the login is completed.

B.2 NCSA Mosaic

The NCSA Mosaic client program and `httpd` server daemon were both modified to allow HTTP requests to use Kerberos Version 5 authentication. The Mosaic client was also modified to use Kerberos authentication when transferring files using FTP.

B.2.1 Modifications for Authenticated FTP

The changes made to the Mosaic sources to interface with the Kerberized FTP daemon were minor. Besides configuration changes encapsulated in the `makefile`, only one source file (`HTFTP.c`) was modified. The changes are essentially the same as to the FTP client described previously, but they have the necessary adjustments for the Mosaic structure.

In the `get_connection` routine, the parsing of the `arg` that contains the host URL string was extended to keep a copy of the parsed host portion of the string (the string following "@") and to examine this string for a port-specifier delimited by ":" from the rest of the host string. If a port specified is found, it is used in place of the default `IPPORT_FTP` constant.

After a successful `USER` command, either a Kerberos ticket or a password can be sent. The modified code first checks to see if there are Kerberos credentials for the client. If so, a `TKT` command is issued, and if accepted by the server, a `sendauth` is executed. If no credentials exist, no `TKT` command is issued and the normal login proceeds with an appropriate `PASS` command. If a `sendauth` is executed, the reply received must indicate the authorization was `COMPLETE` or an error return is generated.

Anonymous FTP access is allowed using the current convention of specifying `anonymous` in the `USER` command if no user name was provided in the URL or if the user name is `ftp` or `anonymous`.

B.2.2 Modifications for Authenticated HTTP

Changes were made to the NCSA `httpd` daemon program and Mosaic client to allow HTTP requests to use Kerberos Version 5 authentication.

The HTTP client (Mosaic, in this case) sends a request to `httpd`. This request contains a `GET` or `PUT` command on the first line. This may be followed by additional lines containing optional directives. Examples of optional directives include the `Accept:` directive, which tells the server what forms of documents the client can accept, and the `Authorization:` directive, which tells the server what authentication to use. The Mosaic client does not initially send an authorization directive, because it is only needed if the document to be accessed is restricted through a `.htaccess` file. An initial request is sent without the authorization. Then, if necessary, the server responds with a message that states authorization is required and what form of authorization is expected. (Presumably, if the client already knew what type of authorization directive was expected, it could send that information with the initial request. However, the NCSA Mosaic client does not do this.)

When the HTTP server (`httpd`, in this case) receives the command, it first checks out what kind of HTTP authorization is required for the requested document and only looks for an authorization directive if it is needed. If one is needed but not found, the server returns a reply indicating to the client what kind of authentication is required, as specified by the `AuthType` directive in the `.htaccess` file.

The client then continues the dialog by resending the original request, but this time including the required authorization directive. The client uses the first reply from the server in order to determine what kind of authorization directive to send.

Although one of the HTTP documents indicated that a Kerberos ticket might be sent as a parameter of the `Authorization:` directive, it was instead decided to simply send the string `KerberosV5`. The sending of the ticket is then managed by a call to the Kerberos Version 5 library routine `sendauth()`.

The Mosaic and `httpd` sources already had provisions for authentication methods other than basic user name/password pairs, so the changes needed to implement Kerberos were relatively minor.

Changes to `httpd`

Two new routines were added and two routines were modified to allow Kerberos Version 5 authentication in `httpd`. They are described below.

`authenticate_krb5_user(need_local_user)`

This routine executes a `recvauth()` to determine if the client/peer has a valid Kerberos ticket. If so, it checks `need_local_user` to determine whether or not a local user name is required, calling `get_local_krb5_http_user()` as needed. The routine returns 0 if no errors were detected and a local user name, if needed, it is returned through the global variable `user`. Otherwise, an error message is written to the `error_log` file and a -1 is returned.

`get_local_krb5_http_user(client,local_user)`

This routine tries to convert the given client to a local user name using `krb5_aname_to_localname()`. If this is successful, it also calls `krb5_kuserok()` to verify that the returned `local_user` has granted access to the client principal. The routine returns the `local_user` name and 0 if successful. Otherwise, it prints an error message to the `error_log` file and a -1 is returned.

The `auth_bong()` routine was modified to issue an appropriate reply (an `AUTH_REQUIRED` message) to the client when `auth_type` is `KerberosV5`.

The `check_auth()` routine was modified to call the modified `auth_bong()` routine when authorization is required but not received and to call the `authenticate_krb5_user()` routine when the authorization was provided in the client's request. If the authorization was received but invalid, `auth_bong()` is called to tell the client that Kerberos authentication failed. Otherwise, if a local user name is required, the returned user is processed as normal to validate that it occurs as one of the required users in the `.htaccess` file.

Changes to Mosaic

A new routine, `send_k5_cred()`, was added in `HTTP.c`. This routine uses the socket and host name passed as arguments to contact `httpd`. If the host argument is bad, it outputs an error message and returns -1. Otherwise, it invokes a sequence of Kerberos Version 5 library routines to obtain the client's credentials and to call `sendauth()` to send those credentials to the server. If no errors occur, it returns 0. Otherwise, it outputs an error message and returns the status of the failed Kerberos library routine.

`send_k5_cred()` is called at the appropriate place in `HTLoadHTTP()` after this routine has sent a request containing an `Authorization: KerberosV5` directive to the server, but before it tries to get the server's reply. Because of the code structure of `HTLoadHTTP()`, it was necessary to introduce a new state variable, `trying_with_auth`, to determine if the logic for sending the HTTP request contains an authorization directive. This state variable is used to ensure that the `send_k5_cred()` routine is only called when a Kerberos Version 5 authorization directive was included in the request.

In `HTAABrow.c`, the `HTAA_composeAuth()` routine was modified to return the appropriate (empty) authorization string for Kerberos Version 5. This routine is called for all authentication schemes and constructs the part of the `Authorization:` directive that follows the "type" of authorization to use (e.g., for basic authentication, it contains the encoded user name and password). This routine was also modified to set a new global state variable, `current_scheme`, which indicates what authentication scheme is being used and makes it available to the `HTLoadHTTP()` routine. This was consistent with how other state information is communicated among these routines.

APPENDIX C

LLNL XDIR: A NETWORK-ORIENTED FILE MANAGER

APPENDIX C

LLNL XDIR: A NETWORK-ORIENTED FILE MANAGER

Technology

LLNL XDIR answers the need to manage files in a heterogeneous network. It provides a graphical user interface for file transfer and for direct manipulation of local and remote directories on UNIX (and a number of non-UNIX) platforms. LLNL XDIR offers the ability to view information in four different formats, ranging from long lists to tree structures. LLNL XDIR is based on UNIX, the C programming language, OSF/Motif, and FTP; hence, LLNL XDIR is highly portable.

Applications

With LLNL XDIR, users can manage all of their files in a network. Specifically LLNL XDIR can be used to browse directory structures, transfer files and directories, view local and remote files, delete files and directories, rename files and directories, and search directories on one or more hosts for entries matching a specified pattern.

Project Description

LLNL XDIR was developed to manage files in a distributed heterogeneous environment. The program uses the familiar folder-icon metaphor to simultaneously display any number of directories of any number of (local or remote) hosts, with each directory being displayed in its own window. Several existing products are able to manage files on a single system, but LLNL XDIR extends this capability to manage files on an entire network.

LLNL XDIR does not have a "main" window, as such, but instead consists of a number of windows, which display the contents of a directory of the local or a remote host. Each of these "directory" windows has a complete set of controls for setting modes and invoking operations. Several other windows are provided for setting user preferences and displaying diagnostic information.

Features

LLNL XDIR offers elaborate directory-browsing functionality. Each directory can be viewed in several different formats, ranging from long lists to tree structures. A number of mechanisms are provided to traverse directory structures, including 1) double-click to enter a directory, 2) a "go-to-parent directory" button, and 3) a sophisticated history mechanism for easily re-entering previously visited directories.

LLNL XDIR provides a powerful file transfer capability using drag-and-drop. For the user, it is just as easy to copy a file between two remote systems as it is to copy a file from the local host to itself. Files and directories may be transferred either individually or in groups with just a few movements of the mouse.

One of LLNL XDIR's most powerful features is its ability to search directory structures for entry names that match a specified pattern. The user is able to specify the range and depth of such searches. The search can be restricted to a single subdirectory or can extend across machine boundaries.

With LLNL XDIR, the user is able to view local or remote files, using either the built-in viewer or any combination of external viewers of the user's choosing (e.g., "emacs" or "xv").

LLNL XDIR's history mechanism makes it simple to establish a connection with a remote host. This and other features make it especially painless to connect to an anonymous FTP site.

A number of directory-manipulation features are provided for operating on selected entries. There are several ways to select a group of directory entries to operate on, including toggling entries, sweeping out rectangular areas, and selecting wildcards. Once entries are selected, the user is able to rename entries, delete entries (including entire subdirectories), or move entries between directories that are in the same host. The user is also able to create directories, both on local and remote hosts.

LLNL XDIR uses a history mechanism for automatically caching 1) wildcard expressions, 2) previously visited directories, and 3) host names and user names used in connecting to remote hosts. The purpose of each cache is to reduce the amount of typing and mouse movement. For example, when the user wants to apply a wildcard expression to select directory entries to operate on, a list of recently referenced wildcard expressions for that host will be presented to the user; the user can then double-click on an existing wildcard expression to apply it. The caches are automatically preserved across LLNL XDIR sessions.

LLNL XDIR presents users with a graphical user interface for easily tailoring the program's behavior. Users can specify options, such as whether they would like the delete operation to first pop up a verification dialog that lists the entries to be deleted.

Finally, an extensive online help facility, much of which is contextual, is provided. However, experience has shown that the user interface is so intuitive that the user rarely needs to access the help package.

Availability

Portability was a major concern during design and implementation. So that LLNL XDIR would be widely available on a variety of platforms, industry standards were chosen for the operating system (UNIX), the programming language (C), the window system (X Window), the file transfer protocol (FTP), and the "look and feel" (OSF/Motif). As a result, LLNL XDIR can be ported to virtually any UNIX platform with minimal effort. Just a few of the many systems LLNL XDIR is expected to run on include the following:

- Sun SPARCstation under SunOS and Solaris
- DECsystem 5830 under Ultrix
- DEC 4000/710 under OSF/1
- SGI workstation
- IBM RS6000
- HP9000/730
- Meiko.

For more information about LLNL XDIR, contact Neale Smith, 510-422-0822, *llnlxdir@llnl.gov*. To obtain a license to commercialize LLNL XDIR, contact Lawrence Livermore National Laboratory's Technology Transfer Office, P.O. Box 808, L-795, CA 94550.

DISTRIBUTION

PNL-10382
UC-900

No. of Copies

OFFSITE

12 DOE/Office of Scientific and Technical Information

J. C. Cavallini
Office of Scientific Computing
ER-30 (GTN)
U. S. Department of Energy
Washington, DC 20585

5 R. J. Aiken
Office of Scientific Computing
ER-30 (GTN)
U. S. Department of Energy
Washington, DC 20585

C. L. Athey
Lawrence Livermore National Laboratory
MSIN L-073
7000 East Avenue
Livermore, CA 94550

D. E. Engert
Argonne National Laboratory
9700 South Cass Ave.
Argonne, IL 60439

J. F. Leighton
National Energy Research Supercomputer Center
MSIN L-561
P.O. Box 5509
Livermore, CA 94551

S. C. Loken
Lawrence Berkeley Laboratory
MSIN 50B-2239
1 Cyclotron Road
Berkeley, CA 94720

No. of Copies

J. E. Ramus
National Energy Research Supercomputer Center
MSIN L-560
P.O. Box 5509
Livermore, CA 94551

R. R. Whitney
Continuous Electron Beam Accelerator Facility
MSIN 12H
12000 Jefferson Ave.
Newport News, VA 23606

ONSITE

10 Pacific Northwest Laboratory

J. D. Fluckiger
G. R. Johnson
J. P. Moore
S. C. Tollbom
Publishing Coordination
Technical Report Files (5)