Kerberos/DCE, the Secure Shell, and Practical Internet Security

Wayne Schroeder, San Diego Supercomputer Center, San Diego, California, U.S.A.

ABSTRACT: *Continuing with work described at the Fairbanks (Fall 1995) CUG conference, SDSC now has an operational Kerberos environment for authentication/encryption within SDSC (Cray, Paragon, and workstations) and we are integrating it with DCE systems (DCE security daemon ("secd") and HPSS). However, instead of extending these services to our national (and international) user community, we are using more focused and concise software systems, the Secure Shell (SSH and F-SSH) and Secure Network Key (SNK), to meet our immediate Internet goals. This paper will discuss Kerberos/DCE, the Secure Shell, and SNK, how we are using them, and our longer-term Single-Sign-On goal.*

Introduction

Of necessity, computer security is a continuing SDSC focus. As Tom Perrine, manager of SDSC's Security Technologies team, has noted [1], cracking tools are now being shared, attacks are becoming more sophisticated, and the use of plain-text passwords is the most significant security problem in our distributed environment. For SDSC, as with most Internet sites, plain-text passwords, exploited via simple network "sniffing," is a key vulnerability.

SDSC is using DCE/Kerberos, the Secure Shell, and Secure Network Keys (one-time passwords) to mitigate this problem. In this paper, we will help introduce the Cray community to a valuable new tool, the Secure Shell, describe SDSC's current and planned use of Kerberos/DCE and other tools, briefly describe our HPSS plans (with particular emphasis on secure authentication), and outline some related SDSC initiatives.

The Secure Shell—A Valuable New Tool

The Secure Shell is a valuable new tool for improving Internet security, either in conjunction with Kerberos/DCE or independently deployed. In this author's opinion, it is the most practical currently available solution for some of the more serious Internet security problems. For sites without the resources for a full Kerberos/DCE installation, the Secure Shell offers a relatively simple and easily provided solution.

Both the Secure Shell (SSH) and Kerberos/DCE deal effectively with the plain-text password problem. Each also has additional features, with some overlapping general functionality.

The Secure Shell is like the PGP of interactive access. It is a relatively simple, freely available solution that solves a difficult problem today, while large organizations are still in the process of creating more complex solutions. It was created mostly by one person, Tatu Ylönen <ylo@cs.hut.fi>, at Helsinki University of Technology, Finland.

Kerberos, for example, has been under development for many years and is still in Beta (currently Kerberos version 5 Beta 7). DCE has also been under development for many years and still relies on Kerberos utilities for secure network authentication. Both are fairly large and complex systems. The current Kerberos 5 Beta 7 and Beta 6 are much cleaner and easier to install than previous Kerberos systems, but they are still under development.

We currently have SSH running on our Cray C90, Paragon, IBM SP2, and most workstations: Suns (SunOS and Solaris), SGIs (R4000s, R8000s, and an R10000), DEC Alphas, IBM RS6000s, and PC's running NeXTStep and Windows NT. SSH runs on a large number of Unix systems, and there is a commercial version (F-SSH) available for PCs (and, soon, Macintoshes) from Data Fellows Ltd. [2] Data Fellows also markets and supports a commercial Unix version.

We strongly recommend the use of SSH to the SDSC user community. It is relatively easy to install, can be run independently (i.e., individual workstations can run SSH without site-wide coordination), provides strong password protection, encrypts X-window sessions, and provides additional security and convenience.

Source, documentation, and configure/make scripts are freely available for Unix systems via the SSH home page [3]. Also see the SDSC remote site SSH suggestions.

As described on the SSH home page, SSH (Secure Shell) is a program to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another. It provides strong authentication and secure communications over insecure channels. Its features include the following:

- Strong authentication. Closes several security holes (e.g., IP routing, DNS spoofing, and listening for passwords from the network). New authentication methods: .rhosts together with RSA-based host authentication, and pure RSA authentication.
- All communications are automatically and transparently encrypted. Encryption is also used to protect against spoofed packets and hijacked connections.
- X11 connection forwarding provides secure X11 sessions. This is normally provided automatically for the user.
- Arbitrary TCP/IP ports can be redirected over the encrypted channel in both directions.
- The client RSA-authenticates the server machine in the beginning of every connection to prevent Trojan horses (by routing or DNS spoofing) and man-in-the-middle attacks. The server RSA-authenticates the client machine before accepting .rhosts or /etc/hosts.equiv authentication (to prevent DNS, routing, or IP spoofing).
- An authentication agent, running in the user's local workstation or laptop, can be used to hold the user's RSA authentication keys.
- Multiple convenience features fix annoying problems with rlogin and rsh.

SSH is intended as a complete replacement for rlogin, rsh, rcp, and rdist. It can also replace telnet in many cases.

The basic operation is very simple, much like telnet. On a workstation, a user enters ssh and the name of a host. For example,

ssh *hostname*

will create a login session (as with telnet or rlogin), but the entire session will be encrypted, including the password and any passwords used with 'su'. For many users, that is all they need to learn.

SSH also provides a remote copy (scp) capability which is preferable to rcp, as it provides stronger authentication and can encrypt the transmitted data.

Users who use SSH frequently can use alternative SSH authentication mechanisms instead of passwords. By running 'ssh-keygen' they can create a pair of RSA public/private keys. They then copy the individual public key to the systems they want to log into, into the file ~/.ssh/authorized_keys. At SDSC, since this file is NFS mounted on most of our workstations, users will be able to SSH to most workstations without a password. An example and details are available from an SDSC SSH example page (see the WWW version of this paper for the URL).

When users run ssh-keygen, by default, the individual public and private key files are stored in their home directory (in ~/.ssh). One's private RSA key is like a password and should be protected, so it shouldn't be stored in an NFS file system (the data would traverse the network in the clear). Instead, users can store it on the local workstation, for example, in /tmp/id, encrypting it with a short password, and/or using the ssh-agent to hold the decrypted key. SSH man pages explain how to use these facilities.

SDSC developed a number of extensions for the SSH daemon to run properly on the Cray C90 under Unicos 8 and 9.0, and these are available in the current SSH release. This includes PTY handling, setting up the session account (via acctid), setting up the job (via setjob), and setting limits and permissions (via setlimits). The foundation routines, including encryption, had previously been ported to the Cray, but SDSC was the first to completely install it (i.e., run it as root).

The Cray version of the SSH daemon does not, however, prompt for account selection as Cray Research's login does, when users have multiple accounts available on a particular user id. For the time being, this is acceptable.

The SSH protocols and algorithms are fairly efficient, and so the encryption, while not free, is not excessively costly either. An SSH session to a Paragon node has significantly better interactivity (quicker echoing of keystrokes) than an encrypted Kerberos session (krlogin -x), at least with our port of the krlogin daemon (basically 5 beta 5 code).

Even with DCE/Kerberos available at SDSC, we are using SSH as an option for users, both internally and remotely. Attempting to support Kerberos for our nationally distributed user community on a wide variety of platforms would have been prohibitively difficult. SSH, although not trivial, can often be compiled and installed by users themselves.

(It should be noted that the current Kerberos release (5 Beta 7) is much more solid than previous versions. On many workstations, it can now be configured, compiled, and installed with few difficulties. However, it is still substantially larger and more difficult to work with than SSH.)

SSH uses RSA to exchange a private key for each session, and then uses conventional encryption (including triple DES and IDEA) for the session. Each host has an RSA public/private key set and the SSH daemon (SSHD) maintains an RSA public/private key set for itself (which it changes periodically). When an SSH client connects to the SSH daemon, the SSH daemon sends its host and daemon public keys. The SSH client randomly picks a session key, RSA encrypts it in both keys, and then sends it to SSH daemon. Only the SSH daemon will be able to decrypt the packet. The SSH client and server then have a shared secret key for the session.

The licensing and legality issues for SSH are a little complicated due to the use of RSA, IDEA, and encryption technology in general, but are becoming less so since Data Fellows now has

RSA and other licenses. Reportedly, SSH can be used legally anywhere (except in France and a few other countries where all encryption is forbidden). The commercial version can be used legally almost anywhere, since it includes the required patent licenses. The non-commercial version can also be used legally for non-commercial purposes, as the RSA patent is not valid outside of the U.S., and non-commercial use of IDEA is allowed without a license. Also, the U.S. government can use RSA without a license because it was invented at MIT with partial government funding. So it appears that the non-commercial version is suitable for educational and non-commercial use (i.e., where use of the SSH/RSA/etc package is not being sold). Additional information is available with the distribution and on the web pages.

The following table summarizes the key features and attributes of SSH and Kerberos.

Security Function or Attribute	Secure Shell	Kerberos/DCE Security
Encrypts/Conceals Interactive Passwords	Yes	Yes
Encrypts Entire Interactive Session	Yes, default	Yes, optional
Secure Passwordless Logins	Yes, user managed, developing Single Sign-On	Yes, possible Single Sign-On
Authentication Forwardable	Yes, currently difficult	Yes
Centralized Password Administration	No	Yes
Each Computer Independent (i.e. no site-wide coordination needed)	Yes	No
File Transfer (rcp-like, without plain-text passwords)	Yes	Yes
Encrypts File Data	Yes, default	Yes, optional
Encrypts/Conceals FTP Passwords	Planned	Yes, if using Kerberized ftp and ftpd
Encrypts X-Windows Sessions	Yes	No
Compresses Terminal and/or X-Windows Data	Yes, often 3-5 X	No
Encrypts Arbitrary Sockets	Yes	No
Requires Synchronized Clocks	No	Yes, currently
Application Programming Interface	Planned	Yes
Available For Many Unix Systems	Yes	Yes
Available For Macs	Very soon	V4, V5 Probably
Available For PCs	Yes, Prerelease/Commercial	V4, Good V5s are rare
Source code size (SSH 1.2.14, Kerberos 5B7)	About 44,000 lines (203 *.c files)	About 250,000 lines, 975 *.c files
Available for Crays	Yes, as distributed	Yes, Cray DCE 1.1, or from SDSC, or Sandia
Can it be used legally in the US	Probably	Probably
Can it be used legally outside the US	Yes (as understood)	Maybe

Kerberos/DCE

Although SSH provides much, we are continuing to make use of Kerberos/DCE security and expect to make greater use of it in the future. DCE has strong support from many vendors, and may be growing in popularity. Kerberos 5, after years of development and Beta releases, is nearing final release and provides remote access functionality in a DCE environment. HPSS (High Performance Storage System), the new archival storage system being developed by IBM Government Systems and four DOE laboratories, makes use of DCE. DCE authentication may eventually make Single Sign-On a realistic goal.

DCE, the Distributed Computing Environment, is being developed by the Open Group which was formed in early 1996 by the consolidation of two open systems consortia, X/Open Company Ltd (X/Open) and the Open Software Foundation (OSF). The Open Group includes a large number of computer vendors including IBM, DEC, Microsoft, and SGI/CR (and, until its merger with SGI, CRI).

DCE consists of multiple components that work closely together, including the DCE Remote Procedure Call (RPC), Cell and Global Directory Services (CDS and GDS), Security Service, DCE Threads, Distributed Time Service (DTS), and Distributed File Service (DFS). The Threads, RPC, CDS, Security, and DTS components are commonly referred to as the "secure core" and are the required components of any DCE installation.

As reported at the Fairbanks (Fall 1995) Cray User Group conference in "SDSC's Installation and Development of Kerberos" [4], SDSC was in the process of installing and porting MIT's Kerberos 5 Beta 4 network security software to SDSC systems, including the C90, Intel Paragon, SUNs, DEC Alphas, SGIs and RS6000s. Since then we have converted from a test to a production system, and are now installing 5 Beta 7.

Converting to production required the regeneration and secure distribution of the v5srvtab files (identification files) to each host, and changing the realm name. We also ported and installed Kerberized daemons on the Paragon, installed TCP wrappers on the various Kerberos daemons, corrected some additional bugs, and encouraged staff use.

We now consider this Kerberos system "production" in the sense that it is available for general use and supported by systems staff. Using it and SSH, we have significantly reduced the number of plain-text passwords on our local network. We still provide less secure services, though, such as telnet and rlogin, although we discourage their use.

Although we had originally hoped to distribute Kerberos clients to our user community, both legal and technical issues prevented it. The legal issues involve the distribution of encryption technologies, and particularly a few months ago, it was not clear what protections an organization must take to distribute such software. We also felt that supporting a Kerberos distribution for a large number of remote sites, on a large number of architectures, would be excessively problematic and time-consuming.

We experimented with 5 Beta 5 version of the Kerberos server (KDC) and its backwards compatibility with Kerberos Version 4, but found that moving forward with the DCE security daemon ("secd") would offer us much more. With secd, we can support a Kerberos 5 environment and DCE for HPSS. Rather than supporting Kerberos 4, which is weaker than Kerberos 5, we are finding other solutions for Macintoshes and PCs. These include APOP for email, and soon SSH and/or Kerberos 5 for Macintoshes, and SSH for PCs.

To deal with another major source of plain-text passwords on our local network, the Macintosh Eudora email package, we are installing Eudora with APOP support (see RFC 1939 Post Office Protocol). Eudora periodically logs in to the email (Post Office Protocol) server to access user email. Via APOP, this is handled via a challenge/response authentication method.

The following table summarizes the SDSC Kerberos environment.

Architecture	**Operating System or Function**	**Current**	**Planned**
Cray	Unicos 9.0	SDSC 5B4	Cray 5B5
Intel Paragon	OSF/1	SDSC 5B5	5B7
Sun	SunOS 4.1.x	5B7	5B7
Sun	Solaris 2.5	5B7	5B7
SGI	Irix 5.3	5B4	5B7
SGI	Irix 6.2	5B7	5B7
Digital	Digital Unix 3.x	5B4	5B7
IBM	AIX 4.1.4	5B6	5B7
IBM	AIX 3.2.5	5B4	5B7
Sun/IBM	Security Server	5B5 KDC	DCE 1.1 secd

We are now installing the current version of Kerberos on our workstations. This version, 5 Beta 7 (and 5 Beta 6 last summer), is significantly improved from 5 Beta 5 and earlier versions and can now be configured, compiled, and installed with few difficulties. We have compiled Kerberos Version 5 Beta 7 on SunOS 4.1.4, Solaris 2.5, IRIX 5.3 and 6.2, and Digital Unix OSF/1 2.0. KRB 5 Beta 7 was first installed on Solaris (as we had no previous version of Kerberos there), and on IRIX (as there are known bugs with the previous Kerberos installation there).

We are also integrating our DCE (HPSS) and Kerberos environments. With changes and additions contributed by a DOE working group, led by ANL (Douglas Engert), Kerberos 5 Beta 6 (and 5 Beta 7) can be combined with OSF/DCE by using the DCE security server (secd) instead of the Kerberos KDC (Key Distribution Center). This allows for single sign-on using the DCE userid and password and provides the best features of both DCE and Kerberos, e.g., DCE applications such as DFS and kerberized clients such as rlogin, telnet, or FTP. These kerberized clients can be run on machines with or without DCE, and can be used to establish encrypted terminal sessions.

We plan to run a DCE secd to support HPSS DCE authentication (both internal to HPSS and for user authentication), and to support our Kerberos environment. This will also provide support for additional DCE services if needed. In the future, for example, we may want to make use of the DCE Distributed File System.

Cray Research now supports Kerberos 5 Beta 5 utilities and daemons (ktelnet, klogin, krsh, kcp) as part of their DCE 1.1 product. This is a welcome and appreciated move. We plan to purchase their product to replace our own port of 5 Beta 4. Our current 5 Beta 4 ktelnet daemon is not functioning under Unicos 9.0, apparently due to Unicos 9.0 differences in the setsid and related system calls.

Secure Network Key (SNK)

We are currently using Secure Network Key (SNK) for interactive logins through a gateway, and for remote UniTree access.

The purpose of our SNK gateway host is to provide secure interactive access for staff and for a few general users. Staff can use this while on travel so that the use of plain-text passwords can be avoided (passwords used at conferences are frequently compromised). A user telnet's or rlogin's to the SNK gateway host and then rlogin's to another SDSC host.

Similarly, SNK support in UniTree protects user passwords during remote FTP access.

The SNK system works as follows. When a user makes connection to a SNK enabled server, a challenge number is displayed back. The user enters his Personal ID, and then this challenge number on the SNK key pad, and types in the response it provides. The server than compares that response with the calculated appropriate response to verify authenticity. The SNK card is about the size of a small calculator.

Thus it is a little cumbersome and inconvenient, but does provide very secure access for a small number of people at a reasonable cost. Of course, this wouldn't scale well to our national user community, where easy access for thousands of users is a necessity.

High Performance Storage System

The High Performance Storage System (HPSS) is a new archival storage system being developed by IBM Government Systems, LANL, LLNL, ORNL, and Sandia. Other collaboration partners and early deployment sites include NASA Langley Research Center, Maui High Performance Computing Center, the San Diego Supercomputer Center, Cornell Theory Center, Fermi National Accelerator Laboratory, Caltech/Jet Propulsion Laboratory, and the University of Washington. The HPSS architecture is based on the IEEE Mass Storage Reference Model: version 5 and is network-centered. The control network uses the DCE's Remote Procedure Call technology. In implementation, the control and data transfer networks may be physically separate or shared.

An important feature of HPSS is its support for both parallel and sequential input/output (I/O) and standard interfaces for communication between processors (parallel or otherwise) and storage devices. In typical use, clients direct a request for data to an HPSS server. The HPSS server directs the network-attached storage devices to transfer data directly, sequentially, or in parallel to the client node(s) through the high-speed data transfer network.

SDSC is running HPSS in test now on an IBM SP-2. The SP-2 system will include 23 nodes (20 thin-nodes, 1 wide-node, and 2 high-nodes), 1 TB of SSA (Serial Storage Architecture) Disk, 60 GB of HIPPI attached RAID, and two 3494 tape libraries. Some of this equipment is partitioned for specific data-intensive applications. The HIPPI-attached disk will be used for IPI-3 third-party transfer between HPSS and the Cray C90, Intel Paragon, and SGI hosts. Network interfaces include HIPPI, FDDI, ATM, and Ethernet.

We plan to migrate from NSL UniTree to HPSS as our primary archival storage system in early 1997. We will migrate metadata from our existing archive systems as we convert from NSL UniTree to HPSS. We are also using HPSS as the foundation for a number of major SDSC data-intensive initiatives (see next section).

In addition to DCE RPC, HPSS makes use of DCE Encina as a transaction processing layer. Encina provides journaling and recovery mechanisms, and HPSS uses these to safeguard the Metadata (directory and index information).

The HPSS Parallel FTP (PFTP) utility is a high performance data transfer user interface, similar to FTP but with extensions to allow users to transfer to and from HPSS across parallel communication interfaces.

Other user interface options will include the HSI utility which is under development by SDSC (Mike Gleicher) and will be shared with the HPSS community. HSI will include features of its predecessor, UTI—an NSL UniTree Interface utility—with extensions for HPSS including parallel I/O, class of service, multi-threading, and scheduling. HSI uses the current HPSS API which is DCE RPC- and Encina-based and so limited to DCE platforms for which DCE is available. Since SDSC is not currently running DCE on our major platforms, including the Cray, we can not make use of this immediately. The HPSS team (and/or SDSC) may develop a non-DCE/Encina API, which would then be used by HSI.

For production and batch use, we are adding features from UFTP (our current Unitree FTP user interface program) to PFTP. Not only will this provide us with a simple, standard user interface, but will also make the conversion process much simpler for our users, since the basic user interface will be nearly identical. Our DataTree to UniTree conversion, in contrast, included a completely new user interface.

The UFTP features being incorporated into PFTP include reliable exit status information, execute line stacking (multiple FTP commands on a command line, separated by ";"), automatic retry of recoverable errors (including distinguishing transitory and permanent errors), and password-less secure authentication.

The authentication system for this PFTP/UFTP utility will allow its use from batch jobs on the Cray without either placing passwords into scripts or transmitting plain-text passwords across the network. There are two main choices, a method similar to our current Unitree UFTP mechanism or Kerberos/DCE.

Our current UFTP mechanism uses a 'verify' daemon and support software from LLNL. The UniTree FTP daemon is able to connect to the verify daemon on the Cray and, via a three-way handshake that includes the use of Unix file permissions, confirm the identity of the user. This is similar to the 'ident' protocol (see RFC 1413 Identification Protocol), and we could perhaps make use of it, either directly or with some modification.

If we use this approach, we would have to DCE authenticate on behalf of the user on the HPSS server. To do this, we may use a program developed by Cornell Theory Center (Jeff Deutsch) and Sandia (Bill Rahe) to access DCE credentials from a keytab file.

Alternatively, we could use Kerberos or DCE authentication on the Cray. This would be more in-line with the design of HPSS but would require some support for long-duration and cron jobs, since authentication credentials are normally set to expire after a day or so. Unicos includes a mechanism that "borrows" the restarting processes credentials. After that, the tickets (in the NQE case) are refreshed on a periodic basis before they expire. We expect that this mechanism would be workable for our HPSS interface.

We are currently evaluating both approaches.

Related SDSC Initiatives

Meta-computing, the use of a set of computers as an integrated whole, is a major goal of current and near future SDSC projects. Key components of this include security and a Single Sign-On system.

Single Sign-On (SSO) is a system where users could authenticate (log in) once and repeatedly access the multiple hosts of the meta-computing environment. If or when the DCE/Kerberos environment becomes ubiquitous, Single Sign-On will become a realistic goal.

Data-intensive computing is a major research initiative of SDSC, and we are using HPSS as a foundation for that development. Our goal is to provide the software infrastructure to support the immense datasets of biology, geophysics, and astronomy, managing data through object-relational databases integrated with archival storage systems. SDSC is leading a number of major collaborative projects in this field, including the Massive Data Analysis Project (MDAS) and Distributed Object Computation Testbed (DOCT).

The Massive Data Analysis Project involves defining and prototyping an architecture for accessing very large data archives. The proposed architecture will support data-intensive applications that manipulate very large data sets by building upon object-relational database technology and archival storage technology. We will develop interfaces to integrate these two technologies. Another goal is to research the software infrastructure needed to replace the Unix file system paradigm with a data analysis and Application Programming Interface (API) system capable of accessing terabyte data sets.

The Distributed Object Computation Testbed will create an environment for handling complex documents on geographically distributed data archives and computing platforms. A persistent object representation based on the Legion computation environment will be used to integrate text search systems, document handling systems, and intelligent agents that support electronic submission of documents. The documents will be stored on distributed databases systems that are integrated with archival storage systems. Research activities include development of a distributed scheduling system, and development of the requirements for supporting electronic filing of documents.

Conclusion

Security is a key component of modern computing, particularly as we move to more network-centric and meta-computing systems. Two key technologies are emerging as important components of this.

Today, the Secure Shell (SSH) can be used to easily and efficiently provide secure authentication and privacy on a wide variety of computer systems. It is a practical interim solution and also a useful supplement to Kerberos and DCE security. This author highly recommends SSH to the Cray community and hopes to see it become widely deployed in the HPC and general computing communities.

Kerberos/DCE is continuing development, and is becoming a more practical solution. Kerberos 5 Beta 7 can now be configured, compiled, and installed with few difficulties on many workstations, and Cray Research now provides Kerberos 5 Beta 5 as part of its DCE 1.1 product. We expect that Kerberos/DCE will become a widespread and important technology, especially as Kerberos 5 moves into the final release stage (Version 5 Release 1.0), which is expected in the next few months.

References

[1] Perrine, Tom, Safely Connecting to the Internet, transcript of presentation at NetExpo/Internet Business Conference and other meetings.

[2] Data Fellows Ltd. home page: http://www.europe.datafellows.com/

[3] SSH home page:

http://www.cs.hut.fi/ssh/

[4] Schroeder, Wayne, *SDSC's Installation and Development of Kerberos*, Proceedings, Thirty-sixth Semiannual Cray User Group Meeting, Fairbanks, Alaska (September 1995).
http://www.sdsc.edu/~schroede/kerberos_cug.html

Acknowledgments

This work was funded in part by National Science Foundation Cooperative Agreement ASC-8902825.

All brand and product names are trademarks or registered trademarks of their respective holders.