Building Global HEP Systems on Kerberos

Matt Crawford,* Fermilab, Batavia, IL, USA

Abstract

As an underpinning of AFS and Windows 2000, and as a formally proven security protocol [1] in its own right, Kerberos is ubiquitous among HEP sites. Fermilab and users from other sites have taken advantage of this and built a diversity of distributed applications over Kerberos v5. We present several projects in which this security infrastructure has been leveraged to meet the requirements of geographically dispersed collaborations. These range from straightforward "Kerberization" of applications such as database and batch services, to quick tricks like simulating a user-authenticated web service with AFS and the "file:" schema, to more complex systems. Examples of the latter include experiment control room operations and the Central Analysis Farm (CAF).

We present several use cases and their security models, and examine how they attempt to address some of the outstanding problems of secure distributed computing: delegation of the least necessary privilege; establishment of trust between a user and a remote processing facility; credentials for long-queued or long-running processes, and automated processes running without any user's presence; security of remotely-stored credentials; and ability to scale to the numbers of sites, machines and users expected in the collaborations of the coming decade.

INTRODUCTION

Approaching the building of secure distributed computing environments, several problems commonly arise. Some are particular to large or far-flung collaborations—the "Grid environment"—while others must be addressed even for localized groups of moderate size. This contribution to CHEP 2004 presents Fermilab's solutions to a selection of these common problems.

No claim is made that the solutions presented represent the last or best word in any of the areas explored. Indeed, the first example that will be shown is no longer in use at Fermilab, and has been superseded by X.509 certificate-based methods. Nevertheless, we hope that these case studies will prove useful. Areas for further development are occasionally pointed out along the way.

*Work supported by the U.S. Department of Energy under contract No. DE-AC02-76CH03000.

Environment

A few words are in order about the environment in which these security practices were developed. This all took place without reference to the concept of a Virtual Organization (VO). All computing activities discussed here were either localized at Fermilab or performed remotely on systems with a Fermilab-centered mission. Consequently, all users of the systems were "enrolled" with a single organization. (However, some of them were enrolled through trusted third parties' endorsements without visiting the lab in a physical sense.)

Kerberos and PKI

Although all solutions presented here are implemented using Kerberos, this paper is not to be construed as an argument for the adoption of Kerberos over other security protocols such as PK methods using X.509 certificates. Although no two security protocols are not entirely interchangeable, the significant components of Kerberos and PKI can be mapped into each other, as Table 1 attempts to show. The comparison makes it clear that some common generalizations about the differences between authentication systems based on symmetric-key cryptography and those based on public-key methods don't really hold. Deployment decisions are properly based on considerations other than the underlying cryptography. Those considerations include the support available in the required applications or the details of the infrastructure needed to build the complete security system.

In environments where the users of resources tend to have working relationships with the managers of those resources, Kerberos is more often the chosen authentication technology. Where the users and the resource managers are organizationally more distant,

Kerberos	PKI
Long-term secrets are symmetric keys (DES, AES)	Long-term secrets are asymmetric keys (RSA, ECC)
Symmetric session keys negotiated by communicating parties	Symmetric session keys negotiated by communicating parties
Principal holds secret key	End entity holds private key
KDC issues tickets asserting secret key possession	CA issues certificates asserting public key binding
KDC knows all parties' keys	CAs' public keys known securely to all parties
Ticket-granting tickets reduce the use of long-term client secrets	Proxy certificates reduce the use of long-term client secrets
KDC must be on-line with respect to clients	Fresh CRLs or an OCSP server must be on-line to clients & servers
Cross-realm trust configured by realm managers	Trusted CA list configured by system admins

Table 1: Kerberos to PKI correspondences

PKI systems are more often chosen. But it must be stressed that at a fundamental level, either system can be made to work in either environment. Credentials can even be "translated" (issued in one system based on possession of credentials in the other) [2] [3] [4] although repeated translations are generally undesirable due to information loss with each translation.

Problem Space

The following representative security problems have been selected as examples for this presentation.

- Authentication and authorization of web clients.
- Authentication of a user's unattended processes.
- Delegation of a subset of a user's rights.
- Authentication of long-queued or long-running batch processes.
- Authentication of an agent acting on a user's behalf, when that agent acts for many users.

Methods for scaling the solutions to some of these problems to handle large and widely dispersed collaborations will be discussed at the end.

CASE STUDIES

Web Authentication

An old adage in the field holds that "when all you have is a hammer, everything looks like a nail." That's the sort of thinking led to the first (and perhaps the least interesting) example of building an authenticated distributed application on Kerberos. It was observed that a mature authentication system and authorization structure already existed in the laboratory's AFS cell and could be exploited for fine-grained web access control. Linking to `file:///afs/...` URLs brings the AFS ACLs to bear without having to replicate and maintain user information in `.htpasswd` and `.htaccess` files.

In the negative column, however, is the fact that many useful features of the HTTP protocol are lost in the process – features such as state maintenance, query strings, server-side scripting and the POST method. Better solutions exist now, such as kct [2] and GridSite [5].

Unattended Processes

Even the moderately sophisticated computer user will make use of system facilities like `cron` for starting processes without the user's attention or presence.

In some cases, these processes require authenticated access to other resources over the network. When these processes are part of an existing service, the service's credentials are generally suitable to use for this purpose, but for the majority of users this is not the case and some distinct mechanism is needed.

Any scheme to authenticate an unattended process involves some stored secret available to that process. Hence the security of the operation of that scheme depends on the integrity of the storage of the secret. For the user to store his password on the computer is completely unacceptable. And any alternative mechanism that allows the process to obtain the user's credentials hides an element of risk from security and resource managers: namely, that the trustworthiness of this user's credentials depends on the integrity of a particular computer.

Our solution starts with an extension to the permission structure of the Kerberos administrative server, `kadmind`. This extension allows any principal *user@REALM* to create and destroy principals named *user*/**cron**/*hostname@REALM*. This is combined with a setuid program that helps the user create a stored-key file in a protected area of local disk on *hostname*.

At this point the user is able to add a single command to his job and have it obtain the credentials of *user*/**cron**/*hostname@REALM* whenever it runs. This identity, however, has no rights to any resources until the user places its name into an access control list.

Since the principal's name includes the host where the its key lives, the user, the security manager and, service managers can revoke or suspend the principal's key or its access rights if a suspected security incident requires such action.

Limited Rights Delegation

When authorizing an agent to act on her behalf, a prudent user might wish to delegate a restricted subset of her own rights. This would afford some protection against theft of credentials and malfeasance by the agent, and against errors in the user's instructions to the agent. Unfortunately this is a measure seldom taken, for a variety of reasons. Foremost is that it would be some considerable bother for the user to work out in advance the minimal set of privileges needed to carry out a task. Also, operating systems have rather little support for the concept of some of a user's processes having reduced privileges compared to others. Some storage systems have a bit more flexibility, supporting credentials which specify a set of access rights to a set of files.

The Kerberos "5-to-4 translator" service is able to rewrite principal names in tickets according to configured rules, as it reformats a Kerberos v5 ticket for a Kerberos v4 service. Fermilab users may request principals with names of the form *user*/**afs**/*hostname@REALM*. These principals obtain AFS credentials as if they were simply *user@REALM*, thus obtaining *user*'s AFS access but no other access (unless the user chooses to grant more, through a `.k5login` file for example).

Both Kerberos tickets and X.509 certificates can carry extension fields that could convey limited authorization to a service. Where these extensions are defined, there is either no provision for carrying a subset of the user's rights (Windows 2000), or only a single bit's worth of limitation (Globus "limited proxy" certificates). Both protocols allow the user to request or insert specific authorization data, so there are opportunities for development in this area.

Batch Processing and Shared Agents

It is a common characteristic of batch jobs to be queued or running longer than the lifetime of any delegated credentials the submitting user can provide. The cautious user may hesitate to delegate his full credentials to the batch system, and may even be unaware which batch facility will ultimately run his job. This leads us again to the idea of a distinct authentication entity to represent the user's processes on the batch facility.

The example of unattended process brought out the dependence of one user's special credentials on the integrity of one system. A batch processing cluster multiplies this concern in two ways – many users and the many computers that the system comprises. But the integrity of the computers in such a cluster is tightly coupled and is generally considered as a unit. So rather than multiply the number of principal identities by the number of machines, a single principal per user per cluster is created.

To reduce the risk of one job's identity being stolen by another job, user processes do not have access to the secret keys which obtain their credentials. Instead, the batch system obtains credentials before and/or during job execution, using the recorded identity of the submitting user to select the corresponding principal to identify the job. (Delegated credentials from the user are not required, only authentication of the user.) This removes any requirement for each user to have a

separate system account for execution.

SCALING CONSIDERATIONS

We have taken the security managers and the helpdesk out of the loop for creation and management of users' "cron principals" for unattended process authentication. We had not done the same for the principals used for similar purposes on the batch farms. The great increase in off-site analysis computing by Run II experiments made that a necessity. Once again, we introduced principal naming structure to control the authority and reflect the trust implications of the new principals.

The CAF [6], or Central Analysis Facility (originally CDF Analysis Farm), is a batch cluster design developed by and for the CDF experiment. It developed further into DCAF (DeCentralized Analysis Facility), of which roughly 25 instances exist around the world.

Each CAF (and by that term I include DCAF) has a headnode, to which the lab security team issues one special Kerberos principal called the "headnode principal." Its name is of the form *cafname*/**cdfcaf**/*headnode-name*@FNAL.GOV, where *cafname* uniquely identifies a particular cluster, and *headnode-name* is the fully-qualified domain name of the headnode. The kerberos administrative server for the realm grants authority to the headnode principal to create, destroy, and manage the keys for "CAF user principals" of the form *username*/**cdf**/*cafname*@FNAL.GOV. When a user registers to use a given CAF, the headnode will create such a principal and add its key to a keytab file inaccessible to the user. (The headnode also impersonates each CAF user principal once every few months for the purpose of changing its long-term secret key.) In this way, central support personnel are relieved of the task of managing CAF user principals (which number over 5,000 at this time), while the lines of trust and responsibility for the privileges of those principals are clearly delineated.

When a job starts, the batch monitor obtains a ticket for the job's identity, it renews the ticket during job execution, and it destroys the ticket upon exit. As in previous examples, this CAF user principal has no access rights except what the user explicitly grants. This grant of rights will typically cover access to some file server for delivery of results.

SUMMARY

Fermilab has built a number of custom security solutions on Kerberos, which is a security framework widely used in the HENP community. The solutions are conceptually portable to public-key-based security systems. The special authorization features are all based on minimally structured names of entities, not on newly-defined extensions to credentials. The latter would have more expressive power, but would require development of new client and server software. The schemes presented here involve no protocol changes and work with most vendors' standard Kerberos v5 clients. Some extensions were made to the syntax of the administrative server's ACL file (`kadm5.acl`), but those extensions are now part of the MIT Kerberos distribution [7].

REFERENCES

[1] G. Bella and E. Riccobene, "Formal Analysis of the Kerberos Authentication System". Journal of Universal Computer Science, vol. 3, no. 12 (1997), 1337-1381.

[2] http://www.citi.umich.edu/projects/kerb_pki/

[3] http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-init-20.txt

[4] ftp://achilles.ctd.anl.gov/pub/kerberos.v5/README.sslk5

[5] A McNab and S. Kaushall, "The GridSite Authorization System". Proceedings of Computing in High Energy and Nuclear Physics (CHEP) 2004.

[6] http://cdfcaf.fnal.gov/doc/cafDesign/cafDesign.html

[7] http://web.mit.edu/kerberos/