Rxk5: The Next Generation

Marcus Watts <mdw@umich.edu>

Matt Benjamin <matt@linuxbox.com>

Rxk5

Kerberos 5 security class for rx/OpenAFS. Provides authentication and encryption on behalf of openafs users in a manner similar to rxkad, but using native Kerberos 5 functions.

Works today

Design Ideas

Keep It Simple

Small Footprint

Avoid complex dependencies

Portability

XDR by default

Use Kerberos 5 APIs

Noteworthy for:

XDR wire and token formats

Portable krb5 kernel interfaces

Rxgk

Comparable mechanism designed earlier

GSSAPI

Intending to address larger feature deficit in OpenAFS than Rxk5

Rxk5 TNG borrows ideas and design requirements

What Rxk5 Doesn't Do

PKI for Initial Authentication (authentication using certificates, etc)

Protected Anonymous Access (integrity checking and optional privacy)

Callbacks

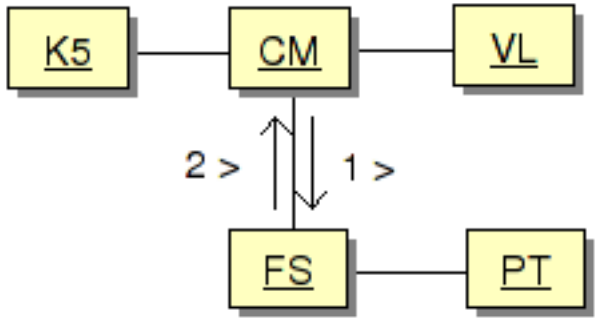
(and callback protection)

Prevent Cache Poisioning

Departmental Fileservers (whee!)

Better Crypto Streaming Support

RPC Conversations



K5

klog/aklog interface K5 and CM, acquire a K5 ticket and convert it to a token within the openafs cache mgr (happens once at thebeginning of an authenticated session)

CM/VL

CM communication with VL required frequently when accessing new volumes, there are other interesting things that might happen here

CM/FS

FS makes an unauthenticated connection to CM for callback processing (and miscellaneous host-tracking and cache-control operations)

FS/PT

on file accesses, FS connects to PT to execute GetCPS calls

Crytographic Mechanisms

Symmetric cryptography not sufficient for anonymous connections, unless clients are keyed

Diffie-Hellman key exchange would be the obvious mechanism to support it

Key Derivation

For us, a mechanism to take a secret and a value which may be secret or non-secret, and generate a new secret irreversably as output

(See below for cases)

Ticket Wrapping

Need a way for a process to provide a credential to a second service

sufficient to allow it to request a third service to perform specific operations on its behalf

eg, allow CM to delegate GetCPS capability to a FileServer for a specific user

PKI for Initial Authentication

For Rxk5, PKINIT is sufficient to solve this problem

Ie, need the capability to map a certificate to a principal in a kerberos 5 database

Streaming Crypto

K5ssl implements k5crypto, but also additional functions.

Requested change is to provide an efficient, scatter-gather encrytion interface

should in future include CMS cryptographic suite

native crypto interfaces where available (OS acceleration).

Anonymous Server Access

This is an unauthenticated access from CM to FS or CM to VL. DH mechanisms will be used to establish a shared secret with a cell.

CM to VL

CM to FS

FS to CM

Cache Poisoning

With Rxkad security, a hostile user sharing a host with legitimate users can impersonate a fileserver to local clients, because he knows his o wn session key

When CM talks to FS, it needs to employ a secret not known to users.

To produce the required secret to protect authenticated file accesses, we need to encapsulate the user tickets in an another ticket encrypted witha key derived from the CM private key, cell public key, and the session key from the user's ticket

FS needs to be able to construct the same session key from the cell private key, the CM public key, and the user's ticket

and encrypted session key. This provides a shared session key different from the user's kerberos session key, and not deducible by the user

Wrapped ticket must contain CM public key and the user's AFS service ticket

Departmental File Servers

Until now, all servers are assumed to be run by a single group, an can share a single key. Now, we will assume the opposite.

When we talk to the fileserver, whatever we give that server must be insufficient to impersonate the user to any other fileserver, or to VLserver, or to Ptserver. In short, the departmental fileserver must not know or receive the key of AFS.

The cache manager must recognize a departmental file server, and when communicating with it, needs to supply it with a service ticket that does 3 things

1/ establish a shared secret

2/ have an expiration date

3/ have the user's name^H^H^H^H^H^H^U

3/ be convertible to something that can be used for 'getcps'. For that user. For a limited time

Fileserver/Ptserver Considerations

For this to work, PT and FS must have a similar capability as that required for CM and VL.

VLserver must inform cache managers when they encounter a dept. fileserver and provide a mechanism to acquire the fileserver's public key

We propose a new GetCPS method that takes a credential supplied by the cache manager and passes it to a ptserver, indicating the operation to be performed (similar to a kerberos AP_REQ)

Ptserver will validate the operation, returning a list of viceids which includes the user's viceid and the expiration of the user's connection

Vos/Vlserver Considerations

The foregoing solves general file operations, but does not address volume operations. Must provide the ability to manage groups of servers as a seamless whole, without conferring the ability to manage an entire cell.

vos command (must set up all machinery)

fileserver a

fileserver b

vlserver

Vlserver has to have a notion of dept. fileserver administrators, and subsets of volume database they are allowed to administrate.

Departments need a way to publish valid administrators. The userlist on a dept fileserver may contain the list of fileserver administrators. When a vos command runs, it must acquire from the dept fileserver something which it can present to the vlserver that the user is an administrator on that server. It must acquire this for each server involved. Presuming user actually is an administrator on each fileserver involved, the operation will succeed.

The case of moving volumes between dept and ordinary fileserver may be a bit ugly.

At some stage it might be nice if userlist magic could be handled by policy mechanisms of some sort...

Vosserver

The rather complicated machinery described above for the vos command could be streamlined by locating all of those operations within a dedicated volume management service

a new server process

would be desireable for other reasons

such as to coordinate long-running operations

Roadmap

Rxk5 for Update 1.5.53 (or very soon)

expect announcement on openafs-devel in a couple of weeks

Rxk5 TNG protocol document

expect on AFS-3 Standardization relatively soon

Rxk5 TNG patches/test releases

expect announcements on openafs-devel later in fall

phase 1: secure unauthenticated access

phase 2: cache poisoning

phase 3: departmental fileservers, vlserver changes

Q & A

What did we miss?