RXGK
why?

AFS & Kerberos Best Practices Workshop 2007

Love Hörnquist Åstrand
Stockholm University

Problem statement

- Need better then fcrypt
- Need to protect the cache manager data
- Not keep re-using same key
- Authentication independent of transport
- Possible to migrate to newer authentication mechanisms

Overview

- Security problems
- The protocol
- Migration
- Status

Security problem

- Symmetric authentication protocols, like Kerberos and NTLM security problem
- The user knows the secret key the client uses to secure the traffic to the file server, and thus can inject/modify data on the fly.
- The attack is an active MITM attack.

Attacks

- Fool the client to allow reading files that exists in the buffer cache

Attacks

- By making the client think the attacker is allowed to modify files, insert data into buffer cache and then wait until the unsuspecting legitimate user flushes the data to the file server

Attacks

- By pre-fetching data from the file server and corrupting it before its first stored in the buffer cache, the when the legitimate user fetches the data, its not the same as on the file server.

What is the problem ?

- user knows the key
- cache manager can’t verify it

Token combing

- Allow the cache manager get a key
- Combine the key with the user key
- Have the rights as the user, but the key as the cache manager knows

Key for cache manager

- A keytab for the machine (machine account)
- Public key for the cell

The Protocol

- RX authentication can't handle large packets
- RX authentication can't multi roundtrips
- Want to have different keys for each connection
- Client chooses the key
- Want to use GSS-API
- Reuse existing cryptographic framework

The Protocol
RXGK service

- rxgk service runs on same port as the service
- Runs in clear text
- Uses GSS-API to secure a rxgk token
  - gss_pseudo_random() to get a key
  - server parameters

Server parameters

- Selected encryption types
- Time and byte limits for connections
- Security level

Connection key

- Derives the connection key based on
  - rxgk key
  - connection id
  - service id
  - start time
  - epoch

GSS-API

- rxgk can use all GSS-API mechanisms that supports gss_pseudo_random()
- In GSS-API all names are not born equal
- Kerberos have not same names as X.509
- afsgk@_afs.cell.name

Names

- Name: type + data
- GSS-API names

Migration

- Will support partly migrated cell
- Transition allows downgrade attacks
- Allows running old file servers and DB software w/o modification

Migration

- New cache manager
- New aklog/afslog
- New fileservers
- New rxgk service on the DB servers
- New PTserver
- New VLDB server that replace rxgk server

Client Flow

- aklog/afslog get list of supported CM mechs
- Is cell is a migrated cell ? Remove rxkad
- Gets tokens and install them

Cache manager flow

- When starting, get a rxgk token
- Before RX challenge/response, check if FS supports RXGK on the same service
- Combine RXGK token
- Use combined RXGK token to fileserver

File server flow

- Extract the name from RXGK token
- Talk to the ptserver to convert to AFS ID
  - If missing ptserver support, convert locally to Kerberos 4 and use that

Status

- test client working
- migration strategies worked out

Left to do

- code cleanup and verification
- aklog/afslog/libkafs (have sample client)
- RXGK service (have sample service)
- OpenAFS file server file server integration
- integration into OpenAFS client
  - arla done

Questions?