Kerberos within Automatically Configured Systems

Simon Wilkinson,
School of Informatics
University of Edinburgh

Introduction

- ~ 1000 machines
- Central configuration database - LCFG
- Every machine reconfigures itself according to the contents of the database
- Reconfiguration happens on the fly, with no administrator intervention
- Installation occurs with minimal intervention
- What to do about service keys?

Machine client keys

- Every machine is given a 'client' Kerberos principal at installation time
- This principal is used to request services from the network that are required by the machine, rather than a user
- Use a different principal (**`hostclient`**) from the default **`host`** service key.

Creating hostclient principals

- Lots of thought went into how to do unattended installations
- Most installations occur on secure switched networks
- List of MAC addresses of machines being installed?
- Key material on installation disks?

Creating hostclient principals

- But ... all too complicated, and not needed in our environment
- Just prompt for an administrator principal and password on the console during installation
- Request a **`hostclient`** principal using the kadmin service, and store it on the machine

Services

- LCFG means that new services can come and go at will.
- No direct administrator involvement in starting, or stopping a service on a machine
- Keeping this was a core requirement of our Kerberos rollout

Service key acquisition (Mark 1)

- Use the **`hostclient`** principal to request service keys from the KDC
- ACLs on the KDC only allow a machine to request keys for itself.
- Local **`kdcregister`** utility gets key material using the kadmin protocol
- Key acquiry automated through configuration management system.

Service keys (problems)

- KDC access is worryingly liberal
- No way of deleting keys when a machine stops offering a service
- No way of removing all the keys for a machine when it is retired

Key acquisition (Mark 2)

- Use LCFG to produce list of service keys for each machine
- Syncronise KDC database against this list locally (but watch for users ...!)
- Machines can only retrieve key material for principals that match their hostname, and that have already been created.

X509 keys

- Solving the X509 key management problem very important
- Same issues as with Kerberos service keys
- Already have a central University CA, which is prepared to sign certificates for other CAs.

SIXKTS

- Locally developed X509 signing service
- Heavily inspired by UMICH's kx509 system
- Machine creates RSA key material
- Sends public key to SIXKTS service, signed with its Kerberos hostclient key
- SIXKTS checks request against configuration database
- Returns signed X509 certificate

SIXKTS

- Uses GSSAPI over a UDP connection
- Dangerous because it leverages a short term Kerberos credential into a long term X509 certificate.
- Could this be safer?
- Could do what kadmin does, but will this work with GSSAPI?

SSH Host Keys

- Best option is to just use GSS key exchange
- Can't do this everywhere. So, still need known hosts file.
- Interesting counterpoint to Kerberos and X509.

SSH Known Hosts

- Maintain known hosts list in LDAP
- Machines extract current list nightly
- Machines publish their own host keys into LDAP when they generate them.
- Publishing is secured using the Kerberos hostclient principals created earlier.

Summary

- Kerberos, X509 and SSH keys all automatically generated and managed
- Possible to extend these techniques to other types of key material
- Private keys which have to be shared between multiple hosts are hard (kca CA certificates, AFS key)