Automated Keytab Installation

Thoughts about designing secure applications

The Problem

- Batch systems need automated install
- New ssh requires krb5 keytabs for GSSAPI authentication
- System must scale to hundreds of machines

Trust

- Trust can only be extended by software, not created.
- What do you trust already?
- How to extend that trust securely?

Ideas that didn't work

- Ip based one time auth
- Client pull schemes
- Temporary trust
- Creating a key hash from ssh keys (this one came closest.)

DOH!

- The automated install process already installs a public ssh key to allow access to the root account.
- Install a separate one to use only for keytab "catching".

Minimum Privilege

- Split the code into parts that can easily be restricted.
- Named pipes can be very useful to use the filesystem for authorization.

On the Client

- catchkeytab : run out of root .ssh/authorized_keys on local machine, expects keytab on STDIN.
- from="afsdb1.slac.stanford.edu",command="/opt/openssh/sbin/catchkeytab"

On the KDC

- listenkeytab : inetd server that gets ip address of connection, writes ip addr to named pipe read by
- instkeytab : A stand alone daemon that runs as root and uses local version of kadmin to create host principals. It then does
- cat keytab | ssh -l root host.slac.stanford.edu

Trust but verify

- instkeytab checks:
  - that the ip addr is slac.stanford.edu
  - that a request hasn't arrived in the last 5 minutes.
  - that the machine has "taylor" installed (ie. is centrally administrated).

Trust but Verify Part II

- catchkeytab checks:
  - If input is larger than X drop on floor.
  - If machine already has a valid keytab, drop on floor.
  - If the keytab is valid, install from temp file to /etc/krb5.keytab.

Denial of Service

- Hundreds of processes all trying to get a lock on the KDC at once is a bad thing.
- Throttle where ever possible via a request queue and make what you can't throttle as lightweight as possible.
- When in doubt throw it out! Never trust your own code, whenever data passes an privilege boundary it must be verified again.