Secure Filesystems

Simon Wilkinson & Craig Strachan
school of Informatics
University of Edinburgh
simon@sxw.org.uk

Previously at UKUUG

- A sequel to last year's "Kerberizing Our Network" talk
- Talked about how we secured a wide range of application protocols
- But left the issue of filesystems dangling ...

Overview

- What is a secure filesystem?
- Our evaluation process
- More details on our selected system
- Deployment Experience
- Questions!

What is a Secure Filesystem?

- ... not NFS v3 (or v2 or v1)
- Unless you use RPCSEC_GSS

Defining A Secure Filesystem

- For today's purposes a "Secure Fileystem" is
  - One that does not rely on network trust
  - One that insulates users of multi-user machines from each other

Different methods of security

- There are (at least) three different ways of providing filesystem security
  - Checking the host identity at mount time
  - Checking the user identity at mount time
  - Checking the user identity at access time

Background on Informatics

- ~ 2000 active users, ~1500 hosts
- 20 Tb of centrally managed filestore
- Deployed Kerberos and LDAP infrastructure

Our Existing Filesystem

- NFS v3 based with Sun fileservers and predominantly Linux clients
- AMD automounter providing identical filesystem on every machine
- Locally developed mechanisms to populate AMD filesystem maps, manage quotas, and do nightly mirroring
- Developed incrementally over many years.

Weaknesses

- Lack of security
  - Can't allow access from unmanaged machines
  - Can't allow access from beyond the firewall

Weaknesses

- Lack of portability
  - AMD infrastructure required significant modifications to off-the-shelf machines
  - Lack of client availability for some systems

Weaknesses

- Lack of maintainability
  - Local glue required lots of effort just to keep running
  - Dealing with partition filling, and the resultant home directory moves
  - Fileserver failure leads to hung mounts, and lots of rebooting

Criteria

- Secure enough to permit access from foreign machines, and across firewalls
- Flexible ACL model
- Better performance
- Stability
- Linux and Solaris support required, Windows and Mac OS X desirable
- Easily scale to our client & data requirements
- No per-client licensing fees
- Preferably be a self-contained solution

Candidates

- AFS
- CIFS
- Coda
- DFS
- NFSv4

AFS

- Originally developed by Carnegie Mellon University as part of Project Andrew
- Commercialised by Transarc, later acquired by IBM
- Became open source (and free!) in 2000
- Strong development community since then

NFSv4

- Next generation of the NFS workhorse
- Developed under the auspices of the IETF
- Takes ideas from most of the other filesystems available, including AFS

Feature Comparison

- On paper, most AFS features are present in NFSv4
- Critical absence is volume location independence
- Can't move filespace between servers without the user noticing
- No concept of a global namespace - still needs automounter glue!

Evaluation

- AFS and NFSv4 feature sets very similar on paper, with NFSv4 leading the way
- However, NFSv4 “not quite ready yet” - few implementations of complete feature set
- Linux NFSv4 only did machine based authentication at mount time
- Bugs in implementation caused benchmarks to hang

Benchmarks

- Three benchmarks selected
  - iozone
  - blogbench
  - The Andrew Benchmark
- Only iozone and blogbench eventually used

Benchmarking Results

- NFSv4 won the iozone one every time - by a small margin for files smaller than the AFS cache size
- Much more evenly matched with blogbench
- “Lies, damn lies, and statistics”

Evaluation Results

- NFSv4 just wasn't ready, and would still have required automounter madness.
- "Don't want our data to be their learning experience"
- OpenAFS met the majority of our criteria, with stability as an added bonus!

AFS

- Units of file manipulation
  - file
  - directory
  - volume
  - partition
- The volume is the key unit for management

Volumes

- Basic volume is a read-write copy
- Multiple read-only replicas for redundancy and load sharing
- Single ‘backup’ volumes provide a snapshot for backup & recovery
- Volume replication is a manual process

Backup Volumes

- Backup volumes are maintained as deltas
- No protection against disk failure
- Provide means for users to access yesterday's data

More on Volumes

- AFS allows volumes to be transparently migrated between servers
- Volumes are stitched together through mountpoints to produce the filesystem
- Filesystem is typically identical on **every** host running AFS

A Global Filesystem

- Standard mountpoint on all clients - /afs
- Next level is 'cell' - your site - /afs/inf.ed.ac.uk
  Derived from DNS or global config file
- Below this, is up to the individual site
- All AFS sites can access /afs/inf.ed.ac.uk
- Our clients can access all AFS cells

Cells and Databases

- A cell is the AFS organisational unit
- Each cell will have a number of database servers providing
  - Volume Location - which fileserver a given volume is on
  - Protection database - group membership and permissions for all users in a
- AFS has powerful multimaster replication for all databases - you want more than one!

Fileservers

- Each cell may contain any number of fileservers
- Fileservers do not store data on disk in human readable form - all access must come through AFS client
- Possible to completely bypass the native filesystem and use the raw disk

Clients

- Clients are comprised of a kernel module, plus a user space daemon - the cache manager
- Cache manager deals with fitting volumes together into the filesystem
- Also handles powerful local caching system

Callbacks

- AFS protects cache integrity using Callbacks
- When a client opens a file it registers a callback with the fileserver
- Any changes to that file will result in the fileserver notifiying the client

Authentication

- Originally AFS used Kerberos v4
- Can now use Kerberos v5 natively
- Only supports DES encryption
- On-the-wire encryption is even weaker
- Better encryption on the way...

ACLs And Groups

- ACLs available to control access on a per directory basis
- ACLs can set permissions by user, system wide group, or by user defined group
- Permissions can be both position and negative
- Special groups exist for ‘any authenticated user’ and ‘any user’

Platform Support

- Linux client works, although lack of a stable kernel API can be a hinderance
- Solaris client very good
- Mac OS X client now works well (some issues with Finder)
- Windows client has improved immensely, although some implementation issues remain

Deployment Experiences

- Softly, softly ...
- Initially offered additional filespace, rather than homedirectories, to the adventurous
- Gradually shifted computing staff home directories over
- Now creating all new users in AFS
- Starting to bulk move existing users

AFS specific Issues

- AFS ACLs aren't as powerful as they could be - only available on a per directory basis
- No support for 'special' files such as devices or named pipes.

Security Hurts!

- Requirement to gain credentials before accessing files causes problems
  - Cron
  - Web servers

Security Still Hurts

- Having to renew credentials is not popular
  - Long running jobs
  - Processes left running overnight (Thunderbird, gnome-screensaver!)
- Unix applications aren't good at dealing with unexpected FS failure

Reduce The Pain

- Get your filesystem credentials at login
- Renew them whenever you can (screensavers &c.)
- Don't have credentials expiring in the middle of the day

Long Running Jobs

- Provide a mechanism for stashing credentials with a subset of permissions on the local disk
- Encourage people to use this to provide credentials for long running jobs

Conclusions

- Going well so far
- The crunch point is just around the corner!
- Softly, softly has perhaps been too soft
- Ensuring reliability before moving users, and responding rapidly to their concerns has been key

QUESTIONS?