WP6 Security Issues

(some personal observations from a WP6 and sysadmin perspective)

Andrew McNab, University of Manchester

mcnab@hep.man.ac.uk

Outline

- Sysadmin hitlist
- Existing VO vs CAS
- Pool accounts
- SlashGrid
- “UID domains”: NFS, PBS etc.
- Need for Grid ACLs
- XML Grid ACL’s
- GACL library
- Certfs as native “container” hosting environment

Sysadmin hitlist

- Subjective list of things to eliminate, from my experience and admins I've talked to:
  - Administrative work creating / maintaining user accounts.
  - Files/processes left over/created in unwanted places by jobs - bad enough when local "students" do this: don't want Student X from University of Z doing this to our kit via the Grid.
  - NFS - "No File Security" - difficult/impossible to secure unless physical components of the LAN are secure (ie in a locked room) - makes it easier to compromise more machines once have root on one.
- I think we now either have foreseeable solutions to all of these...

Existing VO vs CAS

- Have already about VO authorisation servers in use: centrally provided authorisation listings.
- Provides a list of DN 's for a given group: eg an experiment, or a group within an experiment.
- Groups have to be defined by an admin of the VO
  - so an experiment can define the Tau Working Group
  - but I can 't define “my friends in the Tau Working Group” myself
- However, current system gives the functionality running experiments like BaBar cope with, so ok.
- Globus CAS would allow finer grained authorisation.
  - Do we also need a way for users to define new resources and associate authorisation groups with them? In CAS or locally?

Pool accounts

- The other half of removing account creation burden from admins
- Widely used by TB1 sites.
- Auditing possible since all DN=>UID mappings recorded in log files.
- Same pool mappings can be shared across a farm by sharing gridmapdir with NFS (file ops are suitably atomic - but NFS still!)
- Existing system works ok for CPU+tmpfile only jobs.
- But not really appropriate if users creating long lived files at the site in question.
- Limitations are because files are still owned by Unix UID: can't recycle UID until all files created have been removed.

SlashGrid / certfs

- Framework for creating "Grid-aware" filesystems
  - different types of filesystem provided by dynamically loaded (and potentially third-party) plugins.
  - Source, binaries and API notes: http://www.gridpp.ac.uk/slashgrid/
- certfs.so plugin provides local storage governed by Access Control Lists based on DN's.
- Since most ACL's would have just one entry, this is equivalent to file ownership by DN rather than UID.
  - solves admin worries about long lived files owned by pool accounts.
  - if pool accounts are prevented from writing to normal disks, then no chance they will write something unpleasant somewhere unexpected.
- (Also, a GridFTP plugin could provide secure replacement for NFS.)

“UID domains”

- Each testbed site currently constitutes a “UID domain” in which DN=>UID mappings must be consistent on all machines.
  - Currently achieved by sharing grid-mapfile or gridmapdir by NFS (or replicating with LCFG)
- This arises from two major components:
  - NFS sharing of disks.
  - Local batch (usually PBS) by default assumes same UID on front and backend machines.
- Would simplify recycling of pool accounts on gatekeeper if didn’t need to maintain this consistency:
  - gatekeeper would just allocate a pool UID which had no processes already running
  - if use “gsiftpfs” instead of NFS, then DN=>UID mappings done dynamically on SE etc too
  - but, would need to configure / modify PBS etc to dynamically allocate a UID on backend node and copy proxy?

Need for “Grid ACL’s”

- Initial idea of SlashGrid/certfs was to replace ownership by UID to ownership by DN via an ACL.
- For simplicity, would want to use same ACL format for gsiftpfs etc.
- Current prototype is plain text, per-directory ACL in .grid-acl
- As a file, this can be stored in directories, copied via unmodified http, gsiftp channels and easily manipulated by scripts and applications.
- Implementing ACL’s could also solve some other issues to emerge with TB1:
  - eg per-UID tape storage: could store all tape files with one UID but associate ACL with the file and use that.
- Sysadmins want disk filesystem ACL’s on same physical disk as files if possible.

Grid ACL vs CAS (or fine-grained VO)

- CAS provides ACL-like feature of specifying what action (eg write) is permissible on an object (eg tau-wg-montecarlo).
- (If using lots of subgroups within a VO, could achieve much the same thing: eg define a group of people in tau-wg-montecarlo-write)
- In some cases, this could be used to provide ACL functionality.
- However, it is too coarse grained and too heavyweight for all contexts
    - eg if my job creates a temporary, working directory in /grid/tmp, I don't want to setup a new entry on the central CAS machine to control this.
- The two systems should be seen as complementary
    - when you create some tau Monte Carlo, put it somewhere the ACL gives write access for people with "tau-wg-montecarlo write.")
    - when you just create a temporary directory, the ACL defaults to just the creator having admin access.

XML Grid ACL?

- Several variations of XML Grid Access Control Lists have been suggested.
- XML-based format an obvious choice, since:
  - (a) have XML parsers around already for other things
  - (b) many protocols and metadata formats going to XML so could easily include a Grid ACL
  - (c) XML is extensible so we don't need to predict the future so much.
- For files, most seem to be based on about 4 levels: read, list, write and admin (cf AFS.)
- Then associate these with combinations of personal DN's, CAS objects and LDAP VO groups.

Just one example XML Grid ACL format...

```
<gacl version="0.0.1">
<entry>
        <ldap-group><server>ldap://ldap.abc.ac.uk/</server>
                <group>ou=xyz,dc=abc,dc=ac,dc=uk</group>
                </ldap-group>
        <cas-object><dn>/O=Grid/OU=abc.ac.uk/DN=AbcCAS</dn>
            <object>Can-read-http://www.abc.ac.uk/bigfiles/</object>
            </cas-object>
        <allow><read/></allow>
</entry>

<entry>
        <person><dn>/O=Grid/DN=Andrew</dn>
            </person>
        <allow><read/><list/><write/></allow>
        <deny><admin/></deny>
</entry>
</gacl>
```

GACL library

- XML ACL format not decided but want to write code that needs it now (GridSite in production for GridPP; SlashGrid to be in EDG 1.3.)
- ACL may change again in the future; may need to understand different (ugh!) ACL's from other Grid projects.
- Insulate G-S and S-G from this by moving existing ACL handling functions into a standalone library, and make this understand XML.
- Handles ACL's in a reasonably general way, packs C structs with their contents and provides access functions to manipulate the structs as new types:
  - GACLlevel - read, list, write, admin...
  - GACLcred - a DN, VO group or CAS object.
  - GACLentry - several credentials, plus Allow and Deny for Levels.
  - GACLacl - several entries.

GACL library (2)

- Currently uses libxml to do basic XML parsing
  - can read from files or from strings in memory.
- Functions like GACLnewCred(int type, char *issuer, char *name) provided to build up new ACL's in memory, and manipulate or evaluate existing ones.
- Working version of GridSite using GACL exists; SlashGrid next.
- Intend to provide file and directory utility functions:
  - "read in the ACL for file /dir1/dir2/xyz" looks in /dir1/dir2/.gacl-xyz for a file ACL, then /dir1/dir2/.gacl, /dir1/.gacl …
  - but don't limit functionality to files (ACL's on metadata? queues? RB's?)
- Currently, implements XML format from earlier slide.
- See http://www.gridpp.ac.uk/gacl/ for source and API description of 0.0.1 version.

Certfs as container hosting environment

- Some of the OGSA discussions make distinction between simple (eg native Linux) and container (eg Java or .NET) hosting environments.
- The original motivation for “in a box” environments is security.
- OGSA interest is in creating new services dynamically: this is easier if services are “in a box” to start with.
- Certfs is motivated by desire to keep users from making long lived UID-owned files.
- However, it is also a step towards the kind of dynamic environments OGSA talks about.
- Is the answer to our concerns about security and our desire for flexible, dynamic services, to make Unix UID’s as transitory as Process Group ID’s?

Summary

- Most of the concerns of admins are being addressed to some extent.
- Current VO system is probably sufficient, but CAS would be more flexible.
- Pool accounts are useful but limited by UID file ownership issues.
- SlashGrid / certfs intended to provide solution to this.
- Defining a Grid ACL format deals with other issues too.
- Do this in XML: what format?
- GACL library provides API for handling whatever is decided.
- How far can we go towards make UID's purely transitory?