Middleware Security & The INFN Tier-1 – Some Observations

Davide Salomoni, INFN-CNAF

March 27, 2008

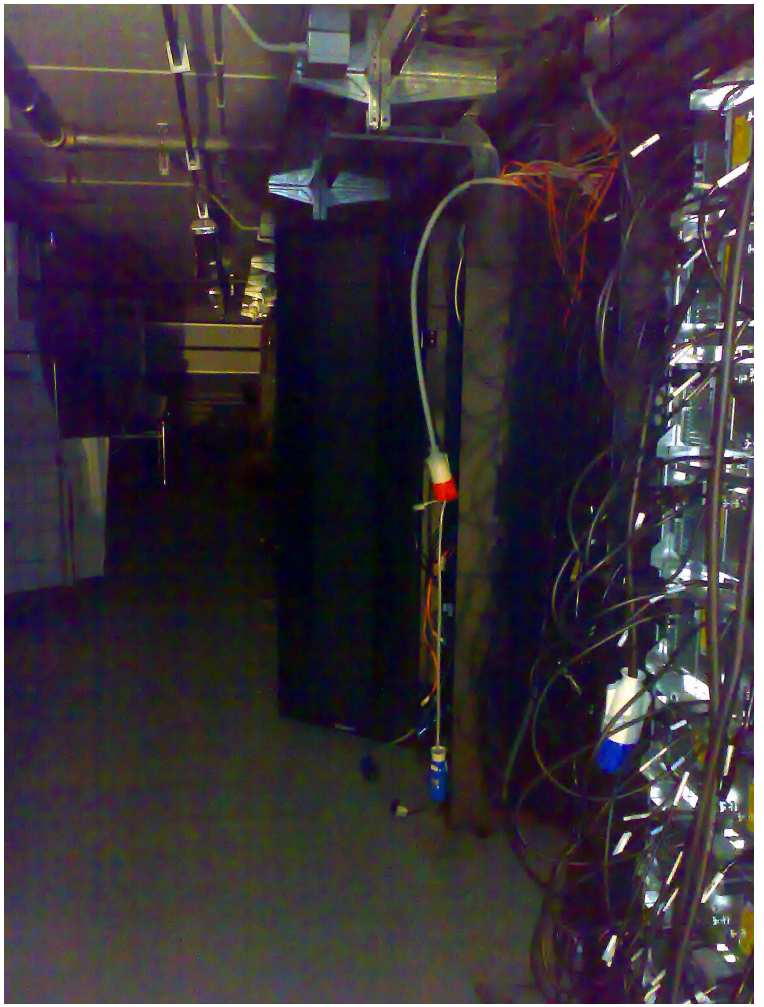
Outline, and a Disclaimer

- The INFN Tier-1 today
- Some middleware security comments, concerns, questions born out of the experience we have made running the INFN Tier-1

- **Disclaimer**: I manage the Tier-1 Farming group, and I am by no means a security expert. Please read this talk as a set of site-related questions.

The Tier-1 *Today*

- *Well, exactly today (March 27, 2008)*, the Tier-1 is a set of disassembled and powered-off racks, due to massive refurbishing of the power and air-conditioning subsystems (sorry for any inconvenience this may cause to your meetings)



The INFN Tier-1

- Location: INFN - CNAF, Bologna (Italy)
  - 1000 m^2 hall in the basement (floor -2)
- Multi-Experiment Tier-1 (~20 VOs, including LHC experiments, CDF, BABAR and others)
  - Resources are negotiated, assigned and enforced to customers – i.e. experiments – on a yearly basis

- One of the core nodes of the Italian Academic and Research Network (GARR)
- In a nutshell:
  - about 3 MSI2K with ~2000 CPU/cores, to be expanded to ~7MSI2K by May 2008
  - about 1 PB of disk space, to be expanded to ~2.6 PB by Q2 2008
  - 1 PB tape library (STK L5500), to be expanded with a new library (for an additional 10PB) by Q3-Q4 2008
  - Gigabit Ethernet network, w/ some (LAN & WAN) 10 Gb/s links

Farming & Storage: Some Configuration Highlights

- **Farm**: LSF-based [hierarchical] fairsharing extensively used, w/ per-VO queues and *dynamic* allocation of resources. High-availability and/or load-balancing provided by a number of solutions (e.g., multiple CEs, redundant KRB/LDAP, XEN-based servers et al.)
- **Storage**: supported storage classes are D1T0 w/ GPFS/StoRM (SRM interface to GPFS) and DxT1 w/ CASTOR. (Tests are ongoing to see how a combination of GPFS/TSM/StoRM could [replace/complement] CASTOR in DxT1 solutions)

Storage Classes

- Disk1tape0 (D1T0)
  - Space managed by the Virtual Organization (VO)
  - No tape copy, data just on disk
  - Already in production at CNAF with GPFS 3.1 and StoRM 1.3
- Disk1tape1 (D1T1)
  - Space managed by the VO (i.e. if disk is full, write fails)
  - Large buffer of disk with tape back-end and no garbage collector
    - 1 to 1 permanent correspondence between files on disk and on tape
- Disk0Tape1 (D0T1)
  - Space managed by system
  - Data migrated to tapes and deleted from disk when staging area is full
  - Active recalls when required data is on tape but no longer on disk

glexec

- Basic requirement: a site needs to know who's using its resources (“incontrovertible evidence”, perhaps for forensic reasons)
  - Since *forcing* a job/all jobs to use glexec does not seem trivial, the site pattern is/should be:
    - ask VO to subscribe to the site AUP (“sign the contract”);
    - if VO derogates from AUP, warn; if warning is ineffective, then potentially **ban** offending user if known (e.g. via glexec logging), and/or the VO proxying actor (typical of pilot jobs who do not present per-user valid delegations), and/or the entire VO.
    - This pattern is not fully in place here yet. We'd like to have evidence of how glexec-like mechanisms (shall/should) integrate with existing logging and accounting, what should be checked by sites, etc. A concern is to have to rely on non-validated, VO-developed accounting systems to get a picture of what's going on.

(glexec and) pilot jobs

- Site says: what happens if a VO task queue used by pilots hasn't got any job to offer, and the VO wants to retain its pilot jobs already running at a site anyway? (or perhaps there *are* jobs to offer, but the pilot jobs task queue is buggy)
  - not much (who cares how the VO uses its allocated resources after all), **if** accounting is done with wall-clock time, rather than with CPU time. Two related issues, though:
    - pledged resources are more often than not expressed in terms of KSI2K
    - pilot job behavior should be accounted for in LRMS configurations e.g. when defining a fairshare dynamic priority formula. Traditional/default LRMS configurations might expect “normal” jobs, rather than pilots eventually forking “normal” jobs.

(glexec and) pilot jobs

- Pilots “wasting resources”: is this a MW security issue?
  - If glexec helps in understanding what's going on with pilot vs. real jobs (for instance, to track efficiency of real jobs – this is normally fairly useful to pinpoint intra-site issues), and if glexec is a MW security component, then yes, it is a MW security issue. Point is, we need clear directions on how to effectively use solutions a' la glexec, and integrated auditing / accounting.
  - If an LRMS only marginally takes into account wall-clock time in its fairshare calculations when dealing with pilots, then it is not unconceivable to think of (maybe even unintentional) cross-VO DoS attacks: I currently have no real jobs to do, but I'll send $O(10^6)$ pilots to a site just to occupy jobs slots – they won't be used by my competitors, nor shall I be penalized too much wrt my fairshare, because I shall use very little CPU time.

(glexec and) pilot jobs

- Is there a way to avoid that a pilot running as belonging to VO X forks jobs belonging to (users of the) VO Y?
  - The reason why they might want to do that: VO Y has used up all of its share already. They will then ask a “shadow VO” – regularly supported at a site – to lend them shares. This subverts standard site accounting and share allocations.
  - A glexec-like mechanism with good logging/integrated accounting might help here (as long as forked jobs present per-user valid delegations).

job priorities

- The motto:
  - PERL: “there is more than one way to do it”
  - Python: “there should be one – and preferably only one – obvious way to do it”
- I like Python.
- Since managing job priorities well is a difficult problem, with potentially high security implications, we hope that *any* solution offered to / required of the community in the future is first fully validated and tested in realistic environments (e.g. avoiding time- and resource- wasting iterations typical of the first VOviews implementations) – where “realistic” should not be confused with “production”.

job priorities

- Basic requirement: at a site, given N shares to which VOs can submit jobs using M roles/classifications (with N not necessarily = M, and perhaps N<<M), it should always be possible for site to be in full control of policies and possibly overrule externally defined policies.
  - (Will skip all technicalities re e.g. mapping mechanisms, and discussions re end-to-end share access)
- For example, as long as site honors contracts signed with its customers (e.g. minimum KSI2K/year for a given VO/sub-VO), we wish to retain full control to (not exhaustive list):
  - prioritize a site-defined, or multi-site-defined set of users
  - prioritize a given funding agency
  - prioritize a VO federation

storage security policies

- Given the supported storage classes here, there are some concerns on authorization mechanisms:
  - GPFS/StoRM uses GPFS ACLs to control access to files; SRM access control is typically performed at the VO group level (e.g. CMS files are readable/writable by any CMS user) – although pattern-level control should also be possible.
  - The CASTOR situation seems to be evolving – need to have more details on this one, waiting for Akos' presentation on Friday.
- In general, what is the status of recommendations regarding Authz on Storage Elements? Do you know about implementation time frames?
  - Is DPM the reference model? Will it require exact matching? Will it then be possible to provide “role-based” access to data?
  - Have you considered / are you considering working out a framework, and policy criteria *similar* to those being discussed for job priorities? (appreciating differences between the two problems) Previous site considerations re overruling external policies might apply here as well.

SOSB (some other security business)

- Are you going to revise the Information System wrt security?
  - what about man in the middle attacks? For example, DNS spoofing applied to top-BDII connecting to site-BDII, thus creating numerous hacking possibilities (job DoS with fake CEs, site DoS pretending a legitimate CE has all its queues in closed state [or has +INF ERT], user proxy stealing, etc.)
  - would https mitigate the problem? (would you be able at least to trace the attacker?)

Thanks, and have a fruitful meeting