Towards Trusted Grid Computing
Contents

1. My motivational example
2. Defining our terms
3. Grid security and trust challenges
4. Recent developments in trusted grid

Description draws on work of, and joint work with: Andy Cooper, David Wallom, Cornelius Namiluko, members of climateprediction.net, and others, including those whose work is cited.
1. My motivational example
1. My motivational example

climateprediction.net
climate modelling has generally been the
preserve of supercomputers
climate models have many uncertainties
previously: one PhD ≈ five model runs
(different parameters, start conditions)
new strategy:
recruit home PC users
each PC runs a complete climate model in
approx. six weeks (elapsed)
Monte Carlo simulation to find the most likely
outcomes
much in common with SETI@home,
distributed.net, etc. and many differences
climateprediction.net users

250 000+ model runs completed / 0.5m users
Important characteristics

heterogeneous hardware
unsophisticated users
  challenges for recruitment and retention
substantial performance 'stretch'
  CPU time
  storage requirements
  network usage
'legacy' software: 1m lines of Fortran, proprietary code
software must run 'unattended'
contrasting data synthesis with search

security?
Classical *dual* problem in distributed computing:

**trusted host, untrusted code**
the wise user will ask:
- does this software do what it claims?
- does it interfere (in installation, in operation) with the rest of my computing experience?

**trusted code, untrusted host**
did the remote user really run the intended software, with the intended parameters?
- are the results returned untainted, accurate, repeatable?
- was anything stolen/copied?
trusted host, untrusted code

code-signing can help
  I was nervous when *my* name was on the certificate!
  doesn't stop bad things from happening – just gives you chance to trust
  the provider
sandboxing *would* help – but tends to degrade performance
  no one wants to compile Fortran to bytecode!
operating system can give some isolation/protection
  but we did need a system which ran no matter who was logged-on
but:
  behaviour of the main executable is one thing; what can we say
  about the behaviour of the installation program?
  we installed extra trusted root certificates, silently, for example
trusted code, untrusted host

any integrity protection measures built into the client software can be spoofed by the user
one response is to duplicate the task
this is what SETI@home does: each work-unit is sent to up to twelve participants, and results are compared
would reduce substantially the number of model runs we could complete
the nature of the task means that we do not expect binary equivalence of results anyway
in general, this is a hard problem!
climateprediction.net security approach

Biggest risks:
well-meaning attempts to 'improve' code
trying to 'game' the leader board
politically-motivated results perturbation
loss of confidence by participants; abandon their model runs

This led us to:
try to avoid an 'arms race': don't make the security too interesting
protect integrity with a simple hash
use statistical measures to estimate the amount of cheating taking place; throw away (or repeat) anomalous results
use code-signing and our reputation to motivate participants

Outcomes: largely successful; one surprising incident...
2. Defining our terms
Grid

associated with “computing on demand”
a vision taken up now by “cloud”
for some, the term relates to uniform access to supercomputer and cluster resources (let's call this a “big grid”)
crossing domain boundaries
managing heterogeneity
co-scheduling CPU, network, and storage resources
supporting lightweight 'virtual organisations'
some of the greatest success has come from desktop grids
Condor platform: high throughput computing
BOINC platform: SETI@home, climateprediction.net
very often pragmatically developed:
performance-driven
reusing and re-purposing hardware
Grid vs Cloud

“over-grown batch system”
services for running jobs
explicit cross-domain working
services for authN and authZ
grid owners and users part of a
mutual enterprise

infrastructure for co-
scheduling, smart resource
handling

many applications will be constructed from a mix of “jobs” and
“services”: the line is quite blurred.
“cloud is a grid belonging to someone else”

rent-a-virtual-machine
servers for running services
create a (temporary) virtual
domain
commercial purchase of
services; disconnect,
disinterested
rent-a-virtual-machine
“Big grid” vs desktop grids

“Big grid” examples: EGEE, UK National Grid Service, TeraGrid characterized by no single point of control but small number of resource brokers, identity providers, authorization/virtual organization services, storage services large clusters, supercomputers, etc.: heterogeneous, but managed degree of variability able to exploit highly parallel and close-coupled systems typically has a mix of compute services and data services Desktop grids: within a department, across a campus, across the world inherent heterogeneity more lightweight services offered best suited to “embarrassingly parallel” problems contrast “high performance” computing with “high throughput”
Example: *Condor*

Condor runs as a *service* on Windows, equivalently on Linux or MacOS — usually offering a background service when the interactive user is idle. Client (worker node) publishes a *ClassAd* describing memory, CPU, operating system, etc.

*Central Manager* receives job requests with associated requirements on memory, CPU, operating system, etc. Jobs originate at a *submit host*

*universe* may be Java container, general environment for native executable, specially-linked executable...
Example: Condor
Example: Condor ClassAd

MyType = "Machine"
TargetType = "Job"
Machine = "froth.cs.wisc.edu"
Arch = "INTEL"
OpSys = "SOLARIS251"
Disk = 35882
Memory = 128
KeyboardIdle = 173
LoadAvg = 0.1000
Requirements = TARGET.Owner=="smith" ||
                 LoadAvg<=0.3 && KeyboardIdle>15*60
Example: Condor Flock
providers may be offering commercial service – or may also be consumers of services
Implementation concept

directory
scheduling
replication
workload management

upper middleware
(collective services)

lower middleware
(resource services)

computation
storage
network
sensors
instruments

application

middleware

operating system

hardware

application
Trust

enormous topic; computer science, psychology, law, ...

TCG:

An entity can be trusted if it always behaves in the expected manner for the intended purpose

“(TCG 2004)”

Trusted to implement my security policy?
Trusted to return accurate results?
Trusted to maintain confidentiality?

“Trusted Grid” ？ “Does what it is supposed to do”
Trusted Platform Capabilities

Trusted Platform Module (TPM) and associated hardware enable:

- **trusted measurement**
  - securely compute a cryptographic hash of software before it is loaded and run

- **secure storage**
  - smartcard-like functionality: hardware protects keys from compromise
  - link with trusted measurement to “seal” keys so that they are available only in a particular software state – i.e. accessible only to designated program

- **trustworthy reporting**
  - give a signed measurement to a third party: show remotely which software is running
  - this a means of “remote attestation”
3. Grid security and trust challenges
Risk-based approach

consider risks according to application domain and system architecture

data with low criticality
proprietary data
  confidentiality
  licensing
proprietary software
  confidentiality
  licensing
personal data
  financial data
  medical records

“big grid” and cloud
  distributed hierarchy
  contractual arrangements – in principle
  must trust system administrators?
desktop grid
  shifting membership
  rely on technological controls
  must trust desktop owners?

embarrassment → scientific accuracy → legal/regulatory requirements
Security Functionality

interoperable identity and authentication
  successful international PKI federation
  certificate usability issues
  other federated identity solutions: *Shibboleth*, etc.
  or use traceable identity mapping, at various levels

delegation
  remote jobs need owner's credentials
  short-lived identity tokens

authorization
  relying party may use identity; may use attributes from various sources

accounting and incident management
  need traceability through identity mappings
  controls have to be available to the right party
Cross-domain security

Users should be strongly authenticated.
But who knows who they are?

Hosts should be strongly authenticated.
But the grid abstraction means I should not identify the host

Users and resources may organise themselves into lightweight, dynamic virtual organisations.
Sometimes involving members of competing organisations.

Must my security depend on the trustworthiness of your system manager?
The Middleware Problem

in the sense previously given, we have to trust the middleware
  to manage credentials correctly
  to mediate delegated access safely
  to enforce appropriate isolation of jobs/data
  to ensure integrity
  to maintain confidentiality

this is probably a bad idea!
Vulnerabilities

Every non-trivial piece of software has security problems.

BOINC (SETI@home, climateprediction.net)
  154MB source distribution; significant vulnerabilities found by Cooper
Condor
  214MB source distribution
Globus
  583 MB source distribution, “23 vulnerabilities listed in v4”
  security component is the most complex

Compare:
  Linux Kernel, 276MB source distribution
  Apache HTTP server, 30MB source distribution
  PHP, 62 MB source distribution
Globus components

(Cooper)
Isolation

isolation and
  data/software integrity
  data/software confidentiality
involves separating one task/process/job/service from another in order to pre-empt theft of data, alteration of results, etc. even by exploiting unforeseen vulnerabilities

separating grid jobs from one another
isolating grid jobs from desktop owner, and vice versa
temporal isolation
  does the job leave data lying in temporary files?
isolating jobs from cluster system manager
Provenance

A special instance of data integrity:

in many disciplines, ethics and scientific good conduct mean that we must know which software, and which inputs, produced a particular result

must be able to repeat a computation exactly

may need to track provenance through workflows, etc.

many disciplines now require data to be deposited alongside journal paper submissions

we may imagine expectations in future that these data will be digitally signed
Pragmatics: a genuine systems issue

these are large, interconnected distributed systems
they will not be deployed as a single purchase
they will probably not all run identical software
a migration path, with interoperability, is crucial
“best available” security may be very different from “perfection”
4. Recent developments in trusted grid
Credential Management

The Problem:

job executes independently from user, but needs user's credentials
  to launch sub-tasks
  to fetch data from long-term storage
  to return results to long-term storage
  etc.

most grid solutions use some form of delegation
  e.g. Globus uses a proxy certificate
    user signs a short-term credential using their long-term credential
    relies on middleware/OS to protect the secret key
    steal that key, and you can impersonate the user for a non-trivial
    length of time: \( O(\text{predicted queue+execution time}) \)
Credential Management

Part solution: MyProxy (Basney et al, 2005)
- service which holds proxy credentials on user's behalf
- jobs can use the service to authenticate (as the user) to a third party
- can also hold the long-term credential: proxy's parent
- in principle, better than placing credentials on arbitrary host: MyProxy server can be carefully managed
- can also be used as front-end for credential translation service

Problem:
- proxy server is now an incredibly valuable asset
- security failure at the server has profound, widespread impact
- (so some certificate providers ban the placing of long-term credentials there)
Credential Management

Better Solution: SHEMP (Marchesini, 2005)
Secure Hardware-Enhanced MyProxy
employs hardware mechanisms (IBM 4758, then TCG TPM) to protect
keys and provide attestation
further, incorporates policy engine
can evaluate security properties of environment before deciding whether
to release/use a particular key
Alternative approach

Daoli (Mao et al) uses the Trusted Platform secure storage at each node to secure user credentials. Delegation of authority is achieved by using TPM protocols for migration of keys. The same approach supports migration of virtual machines (see below).

In a similar way, we can restrict group credentials to be available only to jobs owned by collaborating parties (a virtual organisation).
Enhancing Grid Security using Trusted Virtualization

**Problem:** trust asymmetry. User must trust grid resources, but user is not trusted.

**Solution:** (Löhr *et al*, 2006)
- run each grid job in a virtualized system; a safe container
- employ *trusted* virtualization to offer verifiable multilateral security
- use *offline attestation* to allow job submitters to check that partners (and delegated partners) are in a known state
- Build each grid node using trusted hardware
- Measure and attest the grid software
- Use this to launch individual jobs, only in the case that the right grid software/configuration is running
Enhancing Grid Security using Trusted Virtualization

(Löhr et al, )
Middleware Problem revisited

**Problem:**
Having the right grid software does not guarantee the absence of security exploits
Want to maximize value of existing investments: need backward compatibility/interoperability wherever possible
want to *reduce* the trusted computing base, not *expand* it
The dilemma:
  - grid middleware must have access to users’ data
  - Middleware Problem means it can’t be trusted

**Solution:** (Cooper and Martin, 2006/2007)
make a minimal change to existing grid software
launch jobs as virtual machines
effectively, overlay a security layer independent of the middleware
Solving the Middleware Problem

Security in a virtual machine
“Job Security Manager” (JSM)
JSM enforces access control policies
User pushes down security software in another virtual machine
distributed alongside the grid job
Interoperability Advantages
Service provider no longer needs to pre-install security software
job security managers are user-selectable
can be cached, for performance reasons
Security can be enforced anywhere on the grid
Execute Host Architecture

grid middleware VM

resource access VM

job security manager VM

job attestation service

secure storage service

grid job VM

hypervisor / virtual machine monitor
Job Security Manager

correct instance can be attested to user (or third party) job uses it to mediate all access to resources – local or remote could be the subject of negotiation between user and resource owner: *choose from a menu*, etc.
similarly, data provider may specify acceptable JSMs JSM must attest the job itself – and may attach that information to a signature over the job's output JSM runs only for duration of job: opportunity to subvert running code is limited performance issues:
  attestation at start-up is probably feasible: relatively fast compared to the typical job execution time routing data through two VMs is likely to cause bottle-neck: policy may allow direct connections (unencrypted, or encryption done within job).
Using encryption

Encrypting/decrypting large volumes of data may impact performance
  suites must be selectable, and include “null” option
At the server/data node we may pre-encrypt grid data
  hardware acceleration may be available: unlikely to be available at the compute node
Grid job is encrypted at (or before) submit time
  Grid middleware downloads encrypted data/job VM
  Grid middleware also downloads the job security manager VM
It starts running like any other grid job
Attests itself to obtain the decryption key
Job security manager decrypts and runs the grid job
Trusted Job Execution Path
Key Management Service

a new, additional service; behaviour is not unlike MyProxy/SHEMP
KMS holds keys on behalf of user and/or data provider
  to protect confidentiality of the job itself
  to protect confidentiality of input data
  to protect confidentiality and integrity of results data
can have multiple KMSs: even per-user, if desired
JSM presents credentials and attestation tokens to KMS: keys are released only to trusted security manager
Benefits of this solution

Grid middleware is no longer trusted
Works with all grid middleware without major modifications
launch VMs instead of launching regular job
User credentials (and authorization tokens, if necessary) are used to launch the VMs, but the middleware does not need any delegated authority
Stealing such credentials is not a problem – they only allow the attacker to download encrypted data.
DRM the same way

Digital Rights Management has had mixed success
Offers huge potential in grid-style applications
  locked proprietary data – processed only where licence exists
  locked patient records – ensuring privacy
  attaching policies to outputs, based on policies attached to inputs
    huge research potential here
big idea: key management service attests the security manager;
security manager enforces policy upon job (Cooper and Martin, 2006)
  e.g. “read but don't write”
  “no outbound network access after reading”
  “read:write ratio must exceed 100:1”
  etc.
create overlay network: collection of arbitrary jobs/services which
are constrained only to communicate with each other
... c.f. Virtual domains (Bussani et al, 2005)
Daoli: Work in progress

Daoli implements virtualization for the middleware and for the grid jobs. Significant emphasis on job migration: this is difficult!
Towards a trusted instance of Condor

Current Work (Namiluko 2008)

Problem:
hosts may join a Condor pool simply by announcing themselves to the master host.
How to ensure that rogue hosts do not join, and begin to collect jobs (perhaps by advertising excellent resources), which they then subvert?

Solution:
Condor hosts are already able to use PKI host certificates to establish their identity.
But, of course, these (and the accompanying secret) can be stolen – the secret must exist in unencrypted form when a Condor daemon is running. So, we move the secrets into the Trusted Platform Module, and seal them so that they are available only to a good instance of Condor. This seems a good trade-off for performance and exploiting existing code and protocols.

Next step is to exploit Condor's support for virtualization, and implement the ideas described above.
Summary and Conclusion

grid and cloud computing are a wide target for trusted infrastructure there are several different ideas about how to achieve the goal “sweet spot” right now is desktop grid systems in managed world potential to expand to public participation platforms, and to servers/data centres this is happening now; plenty more to come. most ideas not limited to a particular vision of grid (or cloud), but easiest application is...
right now, the cloud computing paradigm is looking stronger than the grid ideas
can our work on grid inform that world too?
  yes, but this kind of security may not be the most pressing challenge
  much benefit comes from large racks of anonymous servers in remote managed data centres
  conversely most current cloud uptake is for applications with relatively low criticality for security properties
  “hardcore” requirements on confidentiality and integrity are yet to be met:
  a combination of locked racks and trusted platforms may be ideal
  active research topic here include trusted migration of virtual machines
SOFTWARE ENGINEERING CENTRE

Dr Andrew Martin

andrew.martin@comlab.ox.ac.uk
www.softeng.ox.ac.uk/Andrew.Martin

October 2008
References


References


Anthony Bussani; John Linwood Griffin; Bernhard Jansen; Klaus Julisch; Guenter Karjoth; Hiroshi Maruyama; Megumi Nakamura; Ronald Perez; Matthias Schunter; Axel Tanner; Leendert Van Doorn; Els A. Van Herreweghen; Michael Waidner; Sachiko Yoshihama, Trusted Virtual Domains: Secure Foundations for Business and IT Services, IBM Research Report, RC23792 in 2005.

References


Mao Wenbo, and others, http://www.daoliproject.org/