Secure SGE using Kerberos5

Dr. Wolfgang Friebel
Deutsches Elektronen-Synchrotron

Contents

- Motivation
- Available solutions
- SGE Kerberos5 Integration at DESY
- Proposal for tight K5 integration into SGE

Why we need Kerberos5 in SGE

- Verification of the user identity
  - SGE consists of several daemons doing work on behalf of users
  - daemons provide resources on remote hosts
  - SGE does assume the user identity on the submit host is the same as on remote hosts
  - this is host based security as in NFS or with rsh
  - Solution only secure, if users have no control over submit hosts

Why Kerberos5 in SGE (2)

- Granting access to services on execution host
  - most importantly access to Andrew File System
  - with a Kerberos Ticket Granting Ticket an AFS token can be obtained for privileged AFS access
  - other kerberized services could be accessed
- From the security.html document in SGE:

  „the assumption is made that the Grid Engine cluster is not exposed to any malicious attacks“
  - this also means no manipulation of submit hosts!

Enhancing security in SGE

- according to the document cited
  - by using reserved ports (helps in trusting hosts)
  - by using Kerberos/DCE authentication
    - helps in trusting users, not in getting AFS tokens
    - does no longer compile under SGE 6.1, would also require patches for newer MIT Kerberos versions
  - by using Kerberos (currently not usable)
  - by using SSL (now CSP Installation, see docs)
    - helps in trusting users, not (yet) in getting AFS tokens

Kerberos5/DCE Authentication

- Described in the SGE sources under security
  - uses GSSAPI to get Kerberos5 functionality
  - security module gets compiled with aimk -gss
  - requests a K5 ticket for the service sge and does reestablish it on execution host
  - user is authorized to use service sge on exec host
  - cannot be reused for service 'AFS' to get token
- K5 Authentication valid only for limited time
  - no mechanism seen to prolong trust

The AFS problem

- Access to parts of the AFS requires a token
  - typically valid for 25 hours (for security reasons)
  - can be obtained by a valid K5 ticket granting ticket (TGT), which should have similar lifetime
- An AFS token or TGT can be obtained at job submission time, must be valid at job start
  - SGE master needs to babysit TGT or token
- AFS token must still be valid at job end
  - SGE coshepherd needs to continue babysitting

AFS Integration in SGE

- AFS support can be enabled at run time
  - it activates hooks to call get_token_cmd at job submission time and set_token_cmd at job start and regularly later on
- Original solution forges AFS token
  - AFS token gets decoded by set_token_cmd, updates the time stamp and is encoded again
  - cannot be done with Kerberos tickets, as ticket encoding is (by default) host dependent

AFS integration using Kerberos

- Making use of the two hooks for AFS
  - verify the user identity at job submission time by checking that he has a valid TGT (not strictly required)
  - create new tickets on the exec host by
    - using stored user keys on all exec hosts or
    - use admin principals on all exec hosts to get tickets
    - contact a daemon on a secure host to obtain tickets
- DESY is currently using the daemon solution

Problems with Kerberos/AFS

- No checks for job data integrity on master
  - Authentication from submit host can be enforced, but does not go into SGE protocol
  - submit host forgery much harder, but possible
- No hooks for babysitting a Kerberos ticket after submission and before job start
- AFS integration requires to deal with K5 TGT's instead of service tickets
  - can derive AFS token from TGT only

The DESY solution

- Currently no user authentication
  - would require new solutions for job submission from cron and from long running scripts
- Daemon solution using SASL, written in perl
  - kstart client contacts arcxd server
  - arcxd server can obtain K5 admin privileges and request tickets on behalf of users
  - can be downloaded from

`ftp://ftp.ifh.de/pub/unix/gnu/perl/modules/`

Why SASL

- Simple Authentication and Security Layer
  - Comes with several ready to use plugins
    - GSSAPI(Kerberos5, GSI), md5, plain, anonymous...
  - Can be implemented without knowing details of the underlying authentication method
  - Same code in the application can be used for different authentication methods

qrsh Integration

- We are using ssh to provide interactive login
  - both ssh and SGE would provide K5 ticket, but only the SGE mechanism is used
  - conflict solved by special ssh configuration for SGE as explained in the talk of Andreas Haupt
  - qrsh does open a terminal window and uses the X11 tunneling from ssh
  - qlogin and qtcsh are not supported at DESY

Experiences with K5/AFS at DESY

- High stability of the service by software and hardware redundancy
  - client contacts a daemon, if result not ok, contacts the next one on a different host
  - daemons are preforked and handle a limited number of requests only
  - client configuration ensures that load is distributed to several (we use 2) servers
  - service is monitored by external means

Summary & Outlook

- Our present solution serves well our needs
- Security concerns remain
- Therefore we are in favour of a solution that does a tight K5 integration in SGE
  - user authentication preferably using GSSAPI or even SASL (would integrate K5 and GSI)
  - encryption of the traffic (block faked job submits)
  - qmaster stores, maintains and forwards users TGT's to execution hosts

Tight SGE/Kerberos integration

- At job submission time
  - User is authenticated and has a K5 TGT
  - User (qsub) requests a forwarded TGT for qmaster
  - qsub securely transfers new TGT to qmaster
- Before job execution
  - qmaster renews TGT if required
- At job execution time
  - qmaster requests forwarded TGT for exechost
  - qmaster securely transfers TGT to exechost
- During job execution
  - TGT is renewed on exechost if required