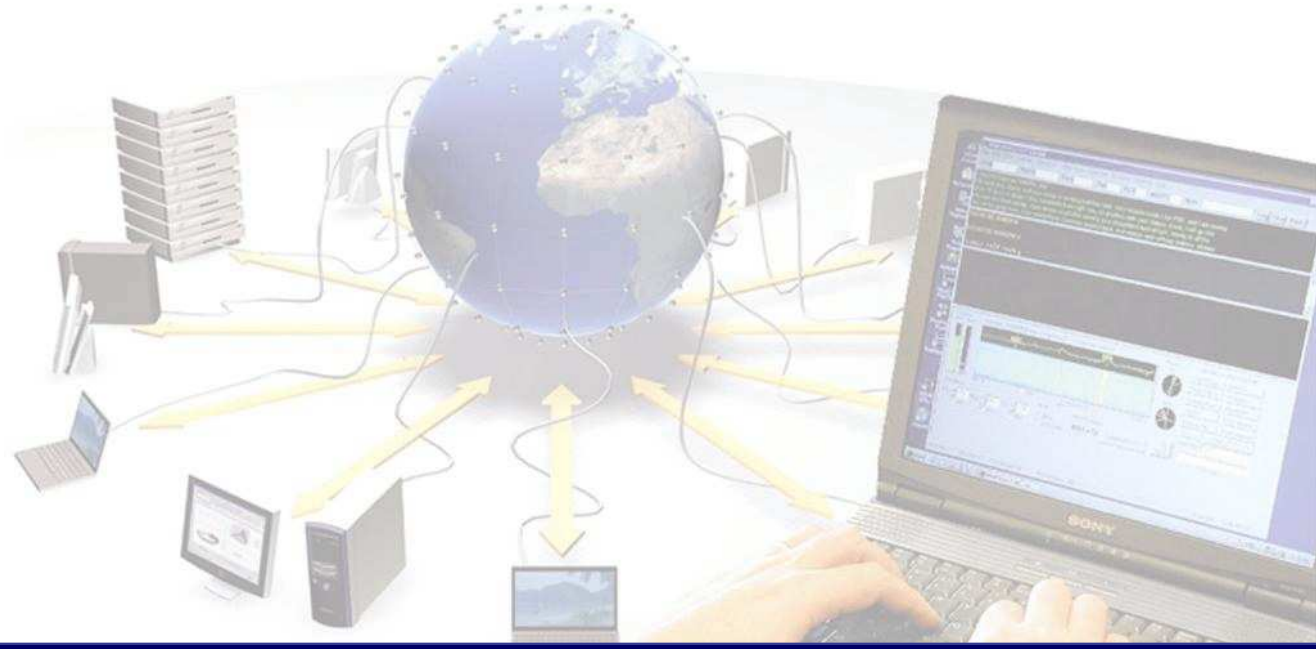
Enabling Grids for E-science in Europe

www.eu-egee.org

IT/CZ cluster meeting, Milano, 3–4 May 2004

Security issues of the IT/CZ cluster

Aleš Křenek



EGEE is a project funded by the European Union under contract IST-2003-508833

Contents

- Summary of general requirements to be propagated to the security group
  - “legacy” EDG ex-WP1 services
  - forseen webservice based implementation
- Dependence on GT 2 libraries

General requirements

- Involved communication: passing job control (submit and cancel), sandbox transfer, accounting information, job monitoring (L&B events, queries, and notifications)
- Encryption required: any communication over network may contain sensitive data (JDL, input/output sandboxes)
- Authentication: mutual by default, single-side or anonymous for troubleshooting only
  - user-service: the service needs to check user's identity for authorization purposes, the user wants to pass sensitive data to trustworthy services only
  - service-service: in most cases a sevice acts on behalf of the user, user's data cannot be revealed to untrustworthy party
- Authorization
  - matchmaking – not all CE's wish to accept this user's/VO jobs
  - job control (cancel), sandbox manipulation, job monitoring info (L&B) – job owner and additional ACL (individual users and groups), the user may manipulate the ACL dynamically (currently implemented in L&B)
  - attach authorization info whenever any data are passed among components, so that authorization can be enforced further
  - ACL/role/attribute management (more or less functionality of EDG VOMS and GACL)

General requirements (cont'd)

- Credential delegation
  - full or restricted – e.g. credentials delegated to WM not usable for submit to WM, those delegated by WM to CE not usable for further submissions
  - both client-to-server and server-to-client
  - also in the middle of communication – CE connects to TQ, TQ decides which job will be sent, i.e. which credentials to use
- Support for short term credentials (GSI proxies, Kerberos tickets)
  - verification routines (ACL's typically contain DN's of users' primary credentials, not the proxies)
  - mechanism for their secure renewal (like MyProxy)
- Credential handling
  - store and re-read credentials later to/from specific location (file)
  - pass security context among processes/threads
  - allow multiple security contexts in a single process (e.g. not addressed by GSS-API well)
- reasonable performance – comparable to SSL (even using stored session context)
- support for asynchronous/timeouting communication

Webservice based implementations

Enabling Grids for E-science in Europe

- All the general requirements hold
- Implementations required in C/C++ and Java. Any others?
- Message level vs. transport layer security
- Interoperability with non-Grid WS world is an issue
  - httpg (http over GSI) addresses most of the requirements but is not compatible with the standard https
  - https can't delegate credentials, problems with GSI proxies (verification, loading to non-GSI SSL implementations)
  - message level – several non-interoperable standards, implementation not available in all required languages
  - there is no general solution but we have to decide consistently for the whole middleware
- Tools to be used (gSoap, Axis, . . .) – what are the known caveats?

Legacy Globus libraries

- `globus_ssl_util` used by L&B
  - API for handling GSI proxies verification callbacks passed to the OpenSSL calls, loading/saving proxy credentials etc.
  - originated in GT 2.0, disappeared in GT 2.2
  - maintained as third party software
  - new proxy format not supported
- `gss_assist` used by `socket++`
  - thin layer on top of GSS-API, hiding some details
  - continued in GT 3
- proxy renewal uses MyProxy libraries (Globus GSS in turn)
- VOMS (used by UI and L&B), uses its own `globus_ssl_util`
- another dependence?
- Globus modified clone of OpenSSL
  - uses Globus common libraries
  - required by `globus_ssl_util`, `gss_assist` and Globus GSS

Legacy Globus libraries – proposed solutions

- cleanup of `globus_ssl_util`
  - turn it into a small library using plain OpenSSL, including multi-threaded support
  - eliminated dependence on Globus common and SSL
  - no support for new proxy format
  - should it make sense, `socket++` must be rewritten accordingly
  - quite extensive effort, short sighted
- migration to `globus_gsi_cred` (available from GT 2.2.4, included in 3.x)
  - dependence on Globus common and SSL
  - both old and new proxy format
  - fairly stable currently, support likely to be continued
  - moderate effort, poor man quick solution

Legacy Globus libraries – proposed solutions (cont'd)

Enabling Grids for E-science in Europe

- migration to GSS-API (and GGF GSS-API Extension)
  - IETF standard, more abstract layer, aplication does not rely on specific security mechanism (in theory)
  - certain required features missing (loading & storing credentials, delegation at any time, etc.), addressed by GGF extension proposal (not quite welcome by IETF)
  - exact server identity must be known in advance – prevents more flexible behaviour (e.g. client is happy with both general credentials of the target host and specific service credentials)
  - creation of security context uses environment, unsuitable for multi-threaded programs (GGF extension has "import credentials" call)
  - the only GSS-over-SSL implementation is Globus, implies dependence on Globus common
  - clean design, dependence on specific security mechanism isolated
  - extensive effort, more code affected, GGF GSS-API Extension necessary
  - still prefered long-term solution