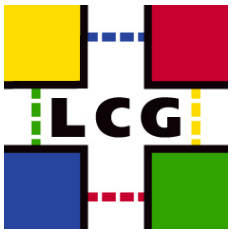

LCG Security Group

Guide to LCG Application, Middleware & Network Security

Date:	**19th July 2004**
EDMS Reference:	**https://edms.cern.ch/document/452128**
Internal Version:	**1.6**
Status:	**final**
Author:	**Ian Neilson**

Document Log			
Issue	**Date**	**Author**	**Comment**
1.0	5th March	Ian Neilson	Initial Version
1.1	27th May	Ian Neilson	Incorporate comments from LCG Sec. Group.
1.2	1st June	Ian Neilson	Comments from Dane Skow
1.3	2nd June	Ian Neilson	Further comments from Dane Skow
1.4	7th June	Ian Neilson	Comments from 4th June phone conf.
1.5	1st July	Ian Neilson	Updated port table
1.6	2nd July	Ian Neilson	Change title and comments from Sec. Group meeting 1st July
	19th July	Ian Neilson	Approved by GDB 13th July 2004

1 Introduction

This document identifies areas of security practice which the LCG[1] Security Group and the Grid Deployment Board consider must be addressed in application and middleware design, planning and deployment processes where such software is to be used by or on the LCG.

The LCG Security and Availability Policy[2] states that

> "All the requirements for the networking security of LCG Resources are expected to be adequately covered by each site's local security policies and practices".

This document also seeks to identify and clarify issues where local security policy and LCG security policy must be aligned.

1.1 The shortest introduction to the Grid

Very brief descriptions of the grid architecture and services are included in this document in the hope of making it accessible to non-grid-specialist administrators who may be involved in the deployment of grid services at a site.

Conceptually, the grid consists of a number of connected *sites* which offer *resources* to remote *users* across the internet. The resources offered by a site (*resource-provider)* are exposed through interfaces implemented by a number of software *grid-services* executing on computers on the site network (*grid-service nodes).* Grid-services, as well as offering functionality directly to users, also serve peer services resulting in a complex web of network interdependencies. The location and characteristics of services and offered resources are registered in an *information system*, itself implemented by a number of service nodes across the grid. Some further details of common grid services are available in Appendix A.

The Grid is a highly dynamic environment. The standards governing grid services are currently being defined and software implementations are subject to revision. The resources available are continually being expanded and changing in configuration. The user community is geographically widely distributed and volatile. Managing security in this environment, whilst retaining the desired level of accessibility for users is a challenge for software designers, implementers and site administrators.

2 Application, Middleware and Service Development

LCG is a collaboration to deploy a production environment of interoperable grid services. As such, LCG expects that the development processes employed by projects whose software LCG deploys will support an adequate and well-documented treatment of security.

The grid environment is highly distributed. An area where current application software requirements and their usage patterns can be misaligned with site security policies is an assumption of the availability of IP network connectivity from anywhere to anywhere. The result of this assumption can be a weakening of site network access control measures and consequent increased likelihood of sites being subject to network attacks from the outside or being used as the source of an outbound attack on a third party. The latter case would be particularly severe if a distributed denial of service attack were to be mounted from a large farm of grid nodes. For this reason sites may choose to restrict both incoming and outgoing network packets at the site boundary or place resources on non-routable internal networks. Solutions taken by system administrators to protect their resources could result in limitations to bandwidth and reliability if not properly planned. The LCG Security Group strongly recommends that application developers, virtual organisations and

[1] Large Hadron Collider Computing Grid – http://cern.ch/LCG

[2] Security and Availability Policy for LCG - https://edms.cern.ch/document/428008

users minimise and structure network connectivity required in their use of resources. Segregating traffic among a well identified and specifiable set of resources enables effective monitoring and access control to be deployed.

Appendix B describes the current connectivity assumptions of commonly deployed grid software used in LCG. The LCG Security Group considers these assumptions to be inappropriate for deploying a production grid and recommends that developers and designers develop ways to more effectively constrain the required access of future systems. Specifically, whilst **application developers MUST NOT rely on the availability of any connectivity not specified in this document,** that described in Appendix B should also **NOT** be read as an agreed minimal set.

For instance, by requiring that network communications pass into and out of a site through one or more recognised grid service nodes or other service proxies, these nodes can act as managed gateways between internal resources and external services and be protected by appropriately configured access control measures and monitoring tools.

Detailed guidance on best practice for software development is outside the scope of this document but the LCG Security Group considers that, as a **minimum**, the following practices should always be observed.

2.1 Design and development process

- Evaluate and document the risks in the current and foreseen threat environment *before* starting and concentrate effort where the risk is highest.

  Prioritising risks early in the development process and ensuring that appropriate control points and mitigating measures are fed into the requirements gathering process will assist in ensuring that the final application usability and security are not compromised by a narrow focus on functionality or performance.

- Adhere to the published practices of the development project. This should ensure, at minimum, a consistent and maintainable product on which to build application security.

  Projects should apply a development methodology appropriate to the requirements. Neither a classic waterfall nor agile development process is appropriate in all cases. A consistency of approach ensures that design documents and code are easily accessible to all members of the project. This facilitates activities such as design review and code walk-through during the development cycle and enables traceability and debugging of problems during testing and eventual deployment. Care should be taken to ensure that configuration instructions are accurate and complete as this is a common area where otherwise secure systems are left vulnerable.

- Apply principle of least privilege.

  Implementing a policy with levels of authorization can assist in maintaining the integrity of services and data under attack and limit damage due to software failure or unintentional use. At its simplest this could be written as “don’t run all services as the root user”.

2.2 Coding practice

- Code for clarity first and optimise during testing if necessary.

  Complexity is the enemy of security. Well-structured, clear code allows for better understanding of intent and appropriate algorithms. It also reduces the likelihood of the introduction of errors which may lead to security problems.

- Reuse tested code where possible.

Software which has been subjected to extended analysis and use is less likely to have exploitable security holes than new software.

- Test all applications for function and fault conditions. Do not assume friendly inputs.

  Even if communications are authenticated and integrity is assured, preventing exploit of buffer overflows and parsing errors limits the propagation of a local security failure across the network.

- Document external code dependencies and include these in the packaging if possible.

  Correct behaviour of software usually depends on the build environment and a large number of external components. Wherever possible the development and packaging should actively prevent an insecure deployment by dependency management. However, it should be noted that overly strict and detailed dependency specification may lead to restrictions in upgradeability. For instance, it should not be required that the version number of a shared library is exactly equal to the version with a critical security bug fix, but rather, equal or greater.

2.3 Communications security

- All network communications should be authenticated and integrity checked. (GSS API[3], GSI API)

  Ensuring that communicating parties are trusted (or at least known) and that communications are not altered makes it much harder for malicious or accidental behaviour to damage the system without being traceable to a cause.

- Any network communication containing sensitive or personal data should be encrypted.

  Whilst the design process must take account of varying legal requirements related to the storage and communication of personal information, identity theft and inappropriate or illegal use of the information gathered from communications across an insecure network remains a possibility if such details are transmitted as clear-text.

- Do not invent new protocols when existing ones can be used.

  It is often tempting to assume that for performance or other reasons an application requires a new or modified protocol to be developed. Experience shows that this is usually not the case and existing standards, which have been open to study and use over an extended period of time, avoid the many subtle failures that can be induced in the development of security protocols.

2.4 Functional security

- All use of resources should be appropriately authorized.

  By deployment of appropriate access control points the system should be designed to ensure that only properly authorized use is made of resources (compute, storage, network etc.).

- Degrade and fail gracefully and with meaningful error reporting.

  When an error occurs (e.g. due to lack of resources, loss of network or communication), determining the cause of the problem and subsequent corrective action is greatly assisted by appropriate predictable behaviour and logging. This is particularly so in a distributed environment where failure patterns can be complex, loosely coupled and poorly reproducible making early capture of the necessary information critical. Failure may be

[3] Generic Security Service API - http://www.ietf.org/rfc/rfc2743.txt

deliberately induced as part of an attack and the system should consequently be designed to remain stable, controllable and secure in all states.

- Log security "state" transitions: connected, authenticated, authorized, disconnected.

  Security failures will occur and detection and analysis is only possible if appropriate information is available.

- Avoid leakage of information through temporary files.

  The location and naming of temporary files, the manner in which they are created, the access-rights assigned and their contents can all lead to the inadvertent or malicious disclosure of information. Similarly, analysis of patterns of communication can lead to inferences about system usage which effectively disclose restricted information.

3 Application, Middleware and Service Deployment

As with software development, LCG expects that instructions to ensure secure application and service deployment will be included in the documentation accompanying the products of associated projects supplying LCG. The LCG Security Group believes at least the following areas should be addressed by those responsible for service deployment.

- Evaluate and document the risks in the current and foreseen threat environment *before* starting and concentrate effort where the risk is highest.

  Security risks to be accommodated in deploying software will vary depending on the circumstances at each site. In some cases the needs of the grid software may be in conflict with established practice and these issues should be understood and addressed before deployment begins.

- Establish a clear network access control policy.

  It is often the case that the administrators of grid resources are not the same individuals as those of site access control systems (e.g. firewalls). Consequently, it is important for the reliability and availability of grid services that the connectivity requirements are properly communicated and agreed by the network managers. This will reduce the likelihood of ports being closed unexpectedly and facilitate the proper monitoring of traffic.

  The table in Appendix B describes the IP connectivity requirements of current grid software.

  Additional guidance for the configuration and use of the Globus Toolkit is available in the Globus Firewall Requirements[4] document. Currently for LCG, only the sections of this document applying to Globus Toolkit V2 (GT2) are applicable.

- Apply Configuration Management and automate wherever possible.

  There are many interdependencies between the configuration of grid services, the operating environment and other peer services. The secure deployment of user-level application software must also be taken into account. Whilst, given sufficient familiarity with the software, manual configuration of a resource is possible, to assist in the generation of a reproducible service interface, it is recommended that administrators make use of a configuration management tool and automate as much of the installation and configuration as possible. Such automation allows for updates to be deployed in a consistent, timely and controlled fashion

- Keep systems patched with security updates.

[4] Globus Firewall Requirements - http://www.globus.org/security/v2.0/firewalls.html

The prompt application of security updates reduces the time window during which the exploit of a known attack is possible.

- Configure & retain audit logs

  Retention of logs for purposes of audit is mandated by the LCG Audit Requirements document[5]. Sufficient information should be retained to enable a complete trace from resource usage back to initial user authentication. This information can be useful for troubleshooting purposes and may also be needed in the investigation of security incidents as described in the Agreement on Incident Response document[6]. Care should be taken to ensure that the logs gathered are securely archived and the integrity of these archives is guaranteed and access appropriately restricted.

[5] LCG Audit Requirements - https://edms.cern.ch/document/428037

[6] Agreement on Incident Response - https://edms.cern.ch/document/428035

4 Further Information

There are many excellent references available to support secure software development and networking. The author recommends the following as recent general texts with further bibliographies to explore.

a) *Secure Coding: Principles and Practice* by Graff & van Wyk (O'Reilly, 2003 ISBN 0-596-00242-4) also www.securecoding.org

b) *Practical Unix & Internet Security* by Garfinkel, Spafford & Schwartz (3rd Edition, O'Reilly, 2003 ISBN 0-596-00323-4)

c) *Firewalls and Internet Security: Repelling the Wily Hacker* by Cheswick & Bellovin (2nd Edition, Addison-Wesley, 2003 ISBN: 020163466X)

d) *Security Engineering* by Anderson (John Wiley, 2001 ISBN 0-471-38922-6)

At the time of writing, up-to-date general information on the changing environment of grid security is harder to find. *The Grid, Second Edition* edited by Foster & Kesselman (Morgan Kaufmann, 2004 ISBN 1-55860-933-4) contains a slim chapter on the subject. The Globus Project[7], Global Grid Forum[8] and other grid project websites can be expected to contain references to the latest developments.

[7] The Globus Project – http://www.globus.org

[8] The Global Grid Forum – http://www.ggf.org

Appendix A. Grid Services

The deployment of grid services brings with it new dictionary of acronyms which can make understanding documentation difficult. The most common LCG service acronyms are described here and a more extensive collection is maintained by the UK GridPP project here:
http://www.gridpp.ac.uk/docs/GAS.html

UI	User Interface The machine where the user is logged on, submits jobs to RBs or CEs and retrieves the output.
RB	Resource Broker Sometimes called the Workload Manager. On receipt of a user's job request, this service matches the job requirements to advertised resources and sends the job to the appropriate place to be run.
CE	Computing Element A service that acts as an interface between the grid and a site's resources. It receives job requests (from an RB or directly from the UI) and manages the running of the job on a local batch system.
SE	Storage Element Whilst a job request has some facilities to 'carry' small volumes of input and output data, the SE is used for bulk data storage and retrieval.
BDII	Berkeley Database Information Index A grid service which forms part of a network of similar service nodes which gather and publish information about grid resources enabling the discovery of resources and their capabilities.
WN	Worker Node A machine in a local batch farm on which jobs are run.
PX	MyProxy Server A service into which a user stores long-term proxy credentials from which RBs can renew short-term proxy credentials.
RLS	Replica Location Service A service which makes available and manages a catalogue of data replica locations.
GRIS	Grid Resource Information Service A service which publishes information about a resource using the LDAP protocol.
VO	Virtual Organization A service which publishes VO membership using the LDAP protocol.

Appendix B. IP Connectivity Table

The following table gives requirements for IP connectivity for the various grid services. Information in this table is derived from the European DataGrid Project[9] firewall table.

Note: The most recent version of this table can be obtained here: *http://lcgdeploy.cvs.cern.ch/cgi-bin/lcgdeploy.cgi/lcg2/docs/lcg-port-table.pdf*

Node	Service	From		To		
		src	port (tcp)	dest	port (tcp)	comment
all						
	ntpd	ntp servers	123/udp	localhost	123/udp	
BDII						
	LDAP	*{RB,UI,WN}	*	localhost	2170	
		localhost	*	*{BDII}	2170	
		localhost	*	*{CE,SE}	2135	
CE						
	Globus & EDG Gatekeepers (GRAM)	*{RB}	C	localhost	2199	
	JobManager	*{RB}	C	*{RB}	C	
	GridFTP Control	*{UI,SE,CE,WN}	C	localhost	2811	
	GridFTP data (single channel)	*{UI,SE,CE,WN}	C	localhost	C	
	GridFTP data (multiple channel)	*{UI,SE,CE,WN}	C	localhost	C	!! Direction of connection is as for dataflow
		localhost	C	*{UI,SE,CE,WN}	C	!! Direction of connection is as for dataflow
RB						
	Logging & Bookkeeping	*{UI}	C	localhost	9000,9001	
	locallogger (logd)	*{CE}	*	localhost	9002	
	CondorG	*{CE}	*	localhost	7771	
	NetworkServer (GRAM)	*{UI}	C	localhost	7772	
	MySQL	localhost	*	localhost	3306	
	Modified GridFTP (see CE)					

[9] European DataGrid project - http://eu-datagrid.web.cern.ch/eu-datagrid/default.htm

Node	Service	From		To		
		src	port (tcp)	dest	port (tcp)	comment
PX						
	MyProxy	*{RB,UI}	*	localhost	7512	
RLS						
	LRC (tomcat)	*{RB,UI,WN}	*	localhost	8080, 9101-9120	
	RMC (tomcat)	*{RB,UI,WN}	*	localhost	8080, 9201-9220	
	MySQL	localhost	*	localhost	3306	
SE						
	RFIO	site{WN}	*	localhost	3147	
	GridFTP (see CE)					
	SRM - httpd(apache)	*	*	localhost	80	
GRIS						
	MDS (LDAP)	*{BDII,RB,UI,SE,CE,WN}	*	localhost	2135	
VO						
	LDAP	*{RB,SE,CE}	*	localhost	389	
misc						
	NFS	site{SE,CE,WN}	*	localhost	2049	Requirement depends on site configuration
	portmap	site{SE,CE,WN}	*	localhost	111 (udp & tcp)	Requirement depends on site configuration
	openssh	site{CE,WN}	*	localhost	22	Requirement depends on site configuration
Key:						
C	Controlable Ephemeral range (e.g. 20000-25000). Note: In practice, although this port-range is locally configurable using the GLOBUS_TCP_PORT_RANGE environment variable, the values applying at a remote service cannot be predicted. Consequently reliable connection can only be established if all ports >1023 are left open for outbound connections.					