REQUIREMENTS AND GUIDELINES FOR DEVELOPING SECURED ACGT SERVICES

Project Number:	FP6-2005-IST-026996
Deliverable id:	D 11.4
Deliverable name:	Requirements and guidelines for developing secured ACGT services
Submission Date:	31/07/2010

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	ACGT
Project Full Name:	Advancing Clinico-Genomic Clinical Trials on Cancer: Open Grid Services for improving Medical Knowledge Discovery
Document id:	D 11.4
Document name:	Requirements and guidelines for developing secured ACGT services
Document type (PU, INT, RE)	PU
Version:	2.5
Submission date:	31/07/2010
Editor: Organisation: Email:	Stefan Castille, Brecht Claerhout Custodix brecht.claerhout@custodix.com

Document type PU = public, INT = internal, RE = restricted

ABSTRACT:

This document defines the concept of an “ACGT enabled” service and specifies the security requirements that it must meet in order to obtain that label. Any service that wishes to process data protected by the ACGT data protection framework must comply with these requirements. Services of a more public nature are recommended to also comply with these requirements and guidelines.

KEYWORD LIST: security requirements guidelines services deployment certificates authentication authorization

MODIFICATION CONTROL			
Version	Date	Status	Author
0.1	21/02/08	Draft	S.Castille
0.2	15/05/08	Draft	S.Castille
0.3	09/10/08	Draft	S.Castille
1.0	24/11/08	Released Version	S.Castille
1.1	07/01/09	Update	S.Castille
1.2	14/04/09	Update	S.Castille
2.0	06/04/10	Update	B.Claerhout
2.1	26/05/10	Update	D.Voets
2.2	25/06/10	Update	D.Voets
2.3	07/07/10	Update	B.Claerhout
2.4	09/07/10	Final Version	D.Voets
2.5	31/07/10	Final, Revised Version	B. Claerhout

List of Contributors

- Stefan Castille, Custodix
- Brecht Claerhout, Custodix
- Juliusz Pukacki, PSNC
- David Voets, Custodix

List of Reviewers

- Alberto Anguita, UPM
- Erwin Bonsma, Philips
- Stelios Sfakianakis, FORTH

Contents

1 Introduction

The ACGT security infrastructure as described in “D11.2: Implementation of the ACGT core security services & Initial implementation of the Pseudonymisation tool” offers service providers the tools to optimally secure their services. Use of the security services is mandatory for services that process data protected by the ACGT Data Protection Framework. Other services are recommended to implement the security services but this is not mandatory. This document lists the specific requirements to be addressed and guidelines to be followed in order for a service to qualify as an “ACGT enabled” service.

1.1 Scope of the Deliverable

The scope of the deliverable is not constrained by a limited interpretation of the word *developing* in the title as “the act of programming”. It is rather interpreted as a reference to the complete software life cycle including design, programming, deployment, configuration, monitoring and maintenance as well as end-of-life management. By considering each step of a service’s lifecycle, security measures can be built in at the appropriate stage and the process of securing a service can be optimized.

1.2 Structure of this Document

“ACGT enabled” is a certification label for services that have been reviewed and considered to be fully compatible with the ACGT infrastructure. In order for a service to meet up to ACGT security standards and be compatible with its security infrastructure, it must meet a minimal set of requirements. In the next chapter the label of “ACGT enabled” will be described from a security viewpoint. Following this high-level description, an overview will be given of the ACGT security infrastructure which **could**, **should** or **must** be used in order for a service to be “ACGT enabled” from a security viewpoint.

The high-level security description of an “ACGT enabled” service is further broken down into more specific requirements and guidelines. In the following chapters these will be mapped onto three main phases of a service’s life cycle:

- Design and development
- Secure deployment and maintenance
- End-of-life of the service

The document concludes with a checklist of security items to be addressed for ACGT compliance, some of which are mandatory while others are advised only (but strongly encouraged).

2 “ACGT enabled” from a Security Point of View

2.1 Introduction

This chapter introduces the definition of an “ACGT enabled” service seen from a security viewpoint. The high-level definition is then used to extract security requirements and guidelines for ACGT services.

An “ACGT enabled” service is a ***secured service*** *that provides access only to those operations and data that are explicitly required by the end user. All access must be obtained only over a* ***secure communication channel****,* ***identity*** *of both communicating parties must be* ***validated*** *and* ***authorization*** *must be confirmed before access is granted. For the purpose of auditing, all operations performed by end users must be* ***logged*** *and* ***archived*** *in a complete and readable format for at least the duration of the ACGT project. An operation performed on an “ACGT enabled” service should be* ***isolated*** *and must leave the service in a* ***consistent*** *state. In case the requested operation cannot be performed due to a problem, the service should provide a* ***clear error message*** *and perform a clean shutdown of the operation. If the service encounters inconsistency or an* ***unrecoverable*** *problem, the service should* ***shut down,*** *preventing further access requests. The impact of a problem on other components of the* ***ACGT infrastructure*** *should be restricted to a minimum.*

Before describing the specific requirements and guidelines, we shall take a closer look at the above definition.

2.2 Security Requirements and Guidelines

An “ACGT enabled” service is a **secured service** that provides access only to those operations and data that are explicitly required by the end user.

Minimization and hardening are two recurring concepts when securing a system. Minimization is the process of removing all unnecessary parts of a system. The underlying idea is that every part of a system can contain exploitable weaknesses. By removing parts that are not strictly necessary for normal operation, those weaknesses can never be exploited. Hardening is done at the level of configuration of a system and its environment. It implies choosing the most secure configuration while still allowing normal operation. Securing the environment with physical access control is an integral part of a secured service.

A service and its environment should be minimized and must be hardened before being deployed as an “ACGT enabled” service.

All access must be obtained only over a **secure communication channel**, **identity** of both communicating parties must be **validated** and **authorization** must be confirmed before access is granted.

From this section three distinct requirements can be extracted: the need for mutual authentication, encrypted communication and authorization. The requirement to secure the environment (operating system, physical access) is implied indirectly.

For the purpose of auditing all operations performed by end users must be **logged** and **archived** in a complete and readable format for at least the duration of the ACGT project.

Legal requirements[1] regarding processing of sensitive data require that an audit trail is available describing the path data has followed during the process. This data must be stored and archived in such a format that reconstruction of the path can be done. This requires that at least the identity of the end user, the action performed by the service and the exact time of this action is logged. This also implies authentication of the user performing the action (or for whom the action is performed through delegation) as well as accurate timekeeping.

An operation performed on an "ACGT enabled" service should be **isolated** and must leave the service in a **consistent** state.

Interference between different operations and services can lead to irregular behaviour and must be avoided at all cost. This leads to the requirement that operations, services and run-time environments should be sandboxed. Sandboxing means limiting a service or application to a well defined area that is separated from the rest of the environment. A sandboxed system cannot interact with its environment except in ways explicitly allowed (e.g. through explicitly defined network connections).

Actions performed on the same service must also be isolated from each other. A concrete example following this principle is given by the requirement that temporary files are to be stored in unique (per process) locations (and with appropriate access rights) and that these files must be removed when the process exits[2].

In case of a problem executing the requested operation the service should provide a **clear error message** and perform a clean shutdown of the operation.

End-users and relying services require clear and correct error reporting in order to decide whether or not to abort the current application flow. Incorrect or unclear error messages might lead to unsafe operations or bring unnecessary load on the ACGT infrastructure.

If the service encounters inconsistency or an **unrecoverable** problem the service should **shut down,** preventing further access requests.

A service that is stuck in an illegal state should be shut down as soon as possible in order to mitigate the risk of security breaches and avoid problem escalation. In order to quickly detect such an illegal state or unrecoverable problem, the service should be added to the ACGT monitoring system.

Additionally an inconsistency might appear on the operating system (OS) level due to unauthorized access. If this is detected, the service should be considered compromised and should be disabled.

The impact of a problem on other components of the **ACGT infrastructure** should be restricted to a minimum

Measures should be taken to contain possible effects of services that end up in an inconsistent or unstable state. For example, one should prevent such services from opening

[1] We refer to the WP10 deliverables for more information on this subject.

[2] This is in particular an issue when running jobs that have been defined outside the job running system (safeguards must be put in place to avoid that one job can read or modify files used by another independent job).

an uncontrolled stream of new connections, potentially causing a denial of services. Such risks can be mitigated by imposing containment measures such as restricting the number of simultaneous connections a server can open, or enforcing limits on memory and CPU usage. Well chosen limits will make sure that a runaway service runs out of resources before it can do extensive damage, although selective monitoring and automatic shutdown (once an illegal state is detected) offer a more preferable solution. Returning incorrect results should be prevented at all costs. If correctness of a result cannot be guaranteed, an error should be returned instead.

2.3 ACGT Security Infrastructure Services

The ACGT infrastructure provides a set of security services that support services in meeting the security requirements and guidelines defined in this document[3]. Use of some of these services is mandatory (such as the central PKI (Public Key Infrastructure) and authorization service) while others, such as the NTP server and monitoring can be replaced by a local version.

- **Public Key Infrastructure (PKI)**: A dedicated public key infrastructure provided to the ACGT consortium generates certificates for end-users and services. Since all end-users will be provided with an ACGT certificate, the root CA certificate should be trusted by all services that depend on authentication. Certificate Revocation Lists (CRLs) and the Online Certificate Status Protocol (OCSP) are available for obtaining up-to-date information about the validity of the distributed ACGT certificates.
- **Network Time Server (NTP)**: Time synchronisation between services running on the ACGT infrastructure is critical[4]. Audit trails can only be generated correctly if the time kept by the different services is accurate. PKI relies heavily on correct date and time recording, e.g. for the validity of certificates or for services which provide time-limited responses such as OCSP[5]. ACGT services are provided access to a dedicated Stratum-1[6] server.
- **MyProxy Credential Repository**: Services that require delegation of credentials have two options, either direct delegation with the called service or through mediation by a credential repository. Direct delegation requires that both the calling and called service support this. If this is not the case ACGT provides a *MyProxy* credential repository.
- **Authorization Service**: Services that wish to limit access to their operations and/or data can use the central authorization service GAS. This service will return an access decision based on the access policy defined in ACGT and by the service owner. As an immediate result of Virtual Organisation (VO) management, acc*ess rig*hts can be enforced over the entire ACGT infrastructure.
- **Monitoring Service**: A monitoring service is provided by ACGT that continuously checks the status of registered services. The service is a pluggable system where custom health checks can easily be added in any of the following programming languages: perl, php, java, c, shell script, etc. Thanks to the pluggable design *and the*

[3] Obviously all services are assumed to be built upon the ACGT middleware (GRIDGE / GT4)., or support the required functionality to interact with the ACGT infrastructure.

[4] Time synchronisation is a must for any system that is secured.

[5] A clock that is out-of-sync could therefore cause valid OCSP responses to be rejected since they appear to come from the future or they appear to be expired.

[6] The stratum level defines the distance of a time source from the reference. A stratum 1 server is a computer attached to a stratum 0 device (i.e. a time source).

possibility to add custom programs as health checks, complex tests can be created to verify the interoperability between various services.

3 Design and Development

3.1 Introduction

In this chapter, we discuss several requirements that have a direct impact on the design and implementation of the "ACGT enabled" services. Although services could be adapted afterwards in order to address some of these requirements, it is strongly recommended to already take these into account during the design process.

3.2 Secure Application Development

While preventive measures such as isolation are targeted towards prohibiting escalation of security breaches (impact control), others are aimed at minimizing the risk of occurrence itself. Adherence to the principles of *secure application development* allows avoiding common and well-understood pitfalls. Although this does not protect against security flaws inherent to the chosen application design, one can expect that adoption of these principles will have a considerate impact on the number and the severity of occurring security issues. Failure to follow guidelines for secure application development is one of the most important causes of security breaches.

A group of security experts from more than 30 international organizations regularly updates a list of the 25 most dangerous programming errors [1]. This list also specifies how these can be avoided. Developers of ACGT services should take these guidelines for secure programming into account. The published list (and in general the SANS website [2]) offers a good starting point for secure application development, but should not be considered complete.

> Secure application development has an impact on all sections of the "ACGT enabled' definition. A service should be audited according to the principles of secure application development before it is granted the "ACGT enabled" label.

3.3 Authentication

Authentication is a crucial security concept. Other security components can easily be bypassed if authentication is not properly performed. Access control can be circumvented by impersonating users with (more) extensive privileges. Failure to adhere to authentication requirements renders other security measures such as encrypted communication ineffective.

Authentication within ACGT is based upon public key authentication. The underlying trust is provided by a hierarchical X.509 Public Key Infrastructure (PKI) which provides certificates to ACGT servers and users.

In simplified form, public key authentication is performed as follows:

- The source sends its certificate (containing its public key and often the certificate of the issuing CA[7]) to the relying party.

[7] The chain of intermediate CAs should be included up to the point where a CA is reached that is explicitly trusted by the relying party.

- The source uses its private key to sign a data structure representing an authentication request (provided by the relying party). This may be either explicit (i.e. bootstrap for the communication protocol) or implicit (as part of the normal application flow). The authentication request typically contains other data that is unique to the conversation context (e.g. a nonce[8] in order to thwart replay attacks).
- The relying party validates the certificate using the CA information and other external resources (CRL, OCSP), the certificates of the CAs are also validated at this time.
- The destination verifies that the source is in possession of the secret key by validating the signature on the authentication request.

Authentication of both parties is essential for achieving secure communication.

The capability to produce a verifiable digital signature is taken as a proof-of-possession of the corresponding private key. Possession of the private key is considered proof that the presented identity corresponds to the real identity of the authenticated party. This system is only effective as long as the private key is properly protected at all time, e.g. it resides in a secure physical location (e.g. smartcard), is strictly personal (never shared with others), is protected by a strong password.

3.3.1 Validation of the credentials

There are several steps to be taken when validating a public key credential. If any of these steps fail then the complete authentication process must fail. The cause of failure should be logged for later reference (both for debugging and auditing purposes).

1. **Verify the validity (expiration status) of the certificate**: When a certificate is created, its lifetime is restricted and must fall within the lifetime of the signing Certificate Authority. If its lifetime has expired, the certificate must not be accepted anymore. Each certificate contains information on when the certificate should be considered valid. This information[9] is embedded in the encoding structure of the certificate.

2. **Verify the origin of the certificate**: When a certificate is issued it is signed using the private key of the Certificate Authority. In turn this CA's certificate may be signed by another CA creating a chain of Certificate Authorities. At one point a CA must be reached that is explicitly trusted. If the end of the certificate chain has been reached without encountering such an explicitly trusted certificate, the certificate must be considered invalid. Each intermediate CA between the certificate and the trusted CA must be valid (recursive process).
   In the case of ACGT, only a limited number of CAs will be accepted as trusted, at this point in time there is only one official ACGT CA.

3. **Check revocation of the certificates**: Although not expired, it is possible that a certificate is no longer valid because it has been prematurely revoked by the Certificate Authority (e.g. after reported key loss or suspected key compromise). A list of certificates that is no longer available is made public by the Certificate Authority that has signed the

[8] nonce, i.e. "number used once", a random or pseudo-random number typically used in cryprographics systems to avoid replay.

[9] This information can easily be extracted using popular tools like OpenSSL. Certificate validity is expressed in a validity period, e.g. Not Before: Jul 12 13:49:56 2007 GMT - Not After : Jul 11 13:49:56 2012 GMT.

certificate. In case of an X.509-based certificate infrastructure (the case of the ACGT security infrastructure), such information is typically published using the following common protocols: CRL and OCSP.

a. **A CRL (Certificate Revocation List)** is a relatively static list that is updated frequently by the CA. The list itself has an expiration date and there is no guarantee that the list will be updated before it has expired. This means that there may be a considerate grey period between the actual revocation of the certificate and the time where the revocation becomes visible through the CRL. The location of the CRL is included in the certificate information[10].

   When using the Globus Toolkit for service deployment, the certificate is checked with CRLs that are available locally on the server. The service **must** have a locally up-to-date version of the CRL. This can be achieved by using a scheduled command that downloads the CRL of each trusted CA at a regular interval (eg. *cronjob* in *NIX environments or scheduled tasks in windows). Since the set of accepted CAs is limited keeping the CRLs up to date will cause a negligible overhead on service management.

b. Another option, besides CRL is the **OCSP (online certificate status protocol)** service. This service can be queried to check whether a given certificate is valid. Since the OCSP service always provides up-to-date information on the revoked certificates, this option is recommended. Not all CAs support an OCSP service as OCSP can have a significant overhead on the service, both in response time and in network usage. This is due to the fact that for any incoming connection, multiple requests need to be made to the OCSP service (one for each certificate in the certificate chain). Also, from a CA perspective, the resources needed for running an OCSP service may be considered too expensive (the OCSP protocol requires each response to be digitally signed). Therefore use of OCSP is not mandatory. As with a CRL, information on the OCSP service is provided by the certificate if there is an OCSP service available[11].
   At this moment there is no base support for OCSP in Globus toolkit so any OCSP check made needs to implemented by the service itself. However, an API is available for java [3].

3.4 Delegation

When a user executes complex workflows which take a long time to complete, (s)he cannot be expected to remain online to perform any authentication step that might be required at certain points in the workflow. Instead, (s)he can delegate a (software) agent to act on his/her behalf and perform this task for him. This agent requires proof of delegation. In ACGT, delegation is provided by means of X509 Proxy certificates [4].

In order to limit the security risk that is introduced by the use of proxy certificates the following guidelines must be followed:

- Delegation must be performed either directly or through the official ACGT *MyProxy* credential management system.

[10] For ACGT: URI: http://ca.custodix.com/crl/crlStore?issuer=cn=ACGT%20CA%202006,ou=Primary,o=Custodix,c=BE

[11] The ACGT OCSP is located at URI: http://ca.custodix.com/ocsp/responder

- Delegated credentials must be protected with a username and password, unique for each delegation session and these must not predictable[12].
- The lifetime of the proxy certificates created for delegation must be limited. The lifetime of the proxy should be enough to perform the requested actions but no longer.
- Proxy certificates must be destroyed after use.

Use of delegation should be limited to the minimum required and must only be allowed for fully authenticated parties. Logs must be kept of all delegation activities, including a timestamp, the credential on which delegation is based as well as the identity of the delegation target.

3.5 Secure Communication

Communication over the Internet is inherently insecure. There is no control over the route taken by the data from source to destination nor over who can see the data as it is transmitted. There is no guarantee that the data packets do not cross country borders, and thus potentially subject to different legal jurisdiction domains.

In order to protect the integrity and confidentiality of the data, secure communication channels need to be set up between relaying parties. A secure communication channel uses encryption to ensure that no third party can intercept the data as it is transmitted, integrity controls based on message digests and/or digital signatures and authentication techniques in order to ensure that no third party can initiate a “man in the middle attack”.

In case of communication patterns were messages need to be routed over intermediate hops, mechanisms must be established to ensure point-to-point confidentiality and integrity between hops (combined with trust establishment or propagation) or end-to-end confidentiality and integrity must be ensured (i.e. intermediate hops only act as message routers and do not perform any content-level processing).

Proprietary applications that do not provide secure communication capabilities can be adapted to meet secure communication requirements:

- By encapsulating unsecured communication into secure tunnels (with tools such as *ssh software* (e.g. *openssh*, *putty)* or *stunnel)*.
- Setup up a Virtual Private Network (VPN) between the application and the services it communicates with. The approach is similar, but is situated at the lower network layer.

Note that the above solutions can also be applied in order to allow communication with services using proprietary secure communication mechanisms.

All communication from and to an “ACGT enabled” service must be done over a secure channel to protect the integrity and confidentiality of data and identities. This is supported by the ACGT middleware. ACGT enabled services can for example use the session-layer SSLv3/TLSv1 protocol with client authentication enabled (mutual authentication).

3.6 Authorization

ACGT uses a centralised Authorization system which must be used by “ACGT enabled” services. The Authorization system is based on the Gridge Authorization System (GAS) [5].

[12] See for instance http://www.us-cert.gov/cas/tips/ST04-002.html

GAS acts as a central Policy Decision Point (PDP) and can be used by services through existing GAS plug-ins (gridFTP, GRAM, WS-GRAM) or by interfacing with GAS directly using one of the available language bindings. A developer guide containing an API is available at `http://www.gridge.org/files/gas/doc/devel/html_one/view/gas-developer-guide-1.0.html`.

Before a service can become “ACGT enabled” it must be configured to use GAS and the access policy for the service must be provided to the GAS administrator.

Authorization decisions are made based on three components:

- Subject: the entity requesting the operation or access to data.
- Object: the service or data to be accessed.
- Operation: the operation requested on the service or data.
- Objects and subjects can be described by a set of attributes with their corresponding values (i.e. name-value pairs).

For auditing purposes all “ACGT enabled” services must use the ACGT authorization service, even when access is public, to ensure that the audit trails are complete.

4 Secure Deployment and Maintenance

4.1 Service Minimization

A service may provide operations that are not needed by regular end-users. Such operations include tests created during development or management operations for the service administrators. It is recommended that such operations are disabled in production environments. However, some management operations are required for maintenance purposes and cannot be disabled.

If possible, access to administrative instances should be separated from regular access in such a way that it can be restricted on multiple levels (i.e. not only on the application level). For example, an administrative instance could listen on a different network port or be deployed at a different server, such that additional network layer access control can be enabled.

A service deployed as an “ACGT enabled” service should only make regular (application-level) operations accessible to end-users. Operations for management, maintenance, testing and monitoring should be shielded from end-users as much as possible[13].

4.2 System Hardening

System hardening is the process of securely configuring or a computer system (at the OS level) or installing specialised system level tools that protect against the unauthorized access, intruders, hackers and other security vulnerabilities.

Many articles and guides are available on the subject for each of the major operating systems (Windows, various Linux and UNIX flavours [6] [14]).

Reducing available vectors of attack includes the removal of unnecessary software, unnecessary usernames or logins, the disabling or removal of unnecessary services and the restriction of access rights for users and servers to the bare minimum.

Hardening includes:

- Minimising the system install
- Scrutinising system and service configuration files, often through specialised scripts, such as e.g. Solaris security toolkit JASS [7].
- Enabling (or installing) specialised hardening software or deploying specific security oriented OS-versions (examples include Windows DEP (preventing buffer overflow exploits) and equivalents, Solaris Trusted Extensions [8], grsecurity [9], Security-Enhanced Linux (SELinux) [10], etc.

[13] “as much as possible” For some services the main goal is to provide administration functionality, e.g. GAS.

[14] Cf. any popular search engine for “hardening <specific_OS>”

4.2.1 Minimization of the Operating System

When using its default installation configuration, an operating system contains a large number of services that are not critical for correct operation of deployed services. These include superfluous services such as graphical tools, remote login services, file sharing, ftp services, printing services, Internet sharing, etc. As every running service and installed software package is a possible attack vector (because of bugs, misconfiguration, etc), secure systems need to be minimized.

> System minimization is achieved by setting up an operating system with only a minimum set of required services installed or removing unnecessary services after installation.

System minimisation is in practice a difficult task. There is in general no obvious way to identify all necessary system components on which a service depends. Creating a minimal installation is thus often a time consuming procedure of trial and error[15]. Due to the time-intensive nature of the minimization process it is frequently omitted or only partially completed in practice.

4.2.2 Configuration Management and Host Intrusion Detection

Default configurations (of necessary system services) are often geared towards user friendliness, typically reducing security. An important aspect of system hardening is thus scrutinising system configurations.

In the same sense, (unexpected) changes of (system) configuration files on a running system can be an important warning signal. They can be caused by unauthorized access to the server and thus must be investigated thoroughly and acted upon.

Systems deploying an “ACGT enabled” service could perform regular automated checks of the system configuration files and binary files for unexpected changes in file attributes, permissions and content.

Next to the many available commercial solutions, automated auditing tools for checking system and common application configurations and configuration changes can be downloaded freely from the Internet for all major OSs[16].

4.2.3 Least Privilege Users and Least Open user

“The Principle of Least Privilege[17] requires that each subject in a system be granted the most restrictive set of privileges (or lowest clearance) needed for the performance of authorized tasks. The application of this principle limits the damage that can result from accident, error, or unauthorized use.” [11]

This principle equally applies to services or applications that are invoked or executed on behalf of a user. Further, a process should only delegate the minimum level of privileges required for normal operation. In practice this implies for example that (temporary) files created by the service should only be accessible by the service itself.

[15] Either top down or bottom up (the best approach). Top down minimizing start from a “full” install from which unnecessary components are removed (frequently performed as corrective measure). The bottom up approach starts from a “minimal install” option that is offered by many OS installers, to which all necessary components can be added afterwards.

[16] E.g. AIDE on Linux (http://sourceforge.net/projects/aide)

[17] From the (old) "Orange book", aka US Department of Defense DOD-5200.28-STD Trusted Computer System Evaluation Criteria (TCSEC)

ACGT enabled services should adhere to the following guidelines:

- **Use a low privilege user**: A service should run under a Least Privilege User (LPU). Modern Operating Systems (OSs) provide advanced capabilities for configuring LPUs (e.g. Solaris 10 mechanisms that go beyons standard *NIX capabilities [12]).
- **Use of strictest possible permissions:** Files and other system-level resources created by a service must be accessible only by the system account under which the service is executed. Features such as Access Control Lists (ACL) [13] should be used if they are available.

4.2.4 Isolation

Computer security not only deals with avoiding security breaches, it also deals with limiting the impact of breaches once they occur. One aspect is to avoid further escalation of a breach on the system (host) level, by ensuring that different services (cf. computer processes) run in strict isolation from each other. This is called “sandboxing” of services running on the same physical machine [14].

The most relevant form of sandboxing for this discussion is through virtualisation. A virtual machine (VM) is a software implementation of a machine (i.e. a computer) that executes programs like a physical machine [15].

Different levels of virtualisation exist, ranging from process level virtualisation (e.g. java virtual machine) over OS-level Virtualizations in which different virtual system share the same kernel (e.g. Solaris Zones, FreeBSD jails), to “full” virtualisation which emulates a complete computer system on which a conventional OS can run just as it would on a physical machine (e.g. VMware, VirtualBox).

Virtualisation finds its origin in efficient use of computer physical computer resources, it’s use as security feature has however always been clear. It is good practice to isolate services in a virtual environment where possible.

4.3 Securing Physical Access

Physical access is a very effective way for an unauthorized person to inflict damage. Simply unplugging a cable or physically destroying a component can cause Denial of Service (DOS). Taking out a disk is an efficient way to steal data. It is in general not difficult to access all data or get high privileged access to a computer system when one has physical access.

Software measures can be taken to reduce the impact of unlawful physical access such as disk-level or database-level data encryption[18], but could introduce excessive overhead on system management or system load and thus should be considered carefully.

As such an “ACGT enabled” service must be installed in a data centre secured to current best-practices. A clearly described security policy must be available. The security aspects that should be covered[19] include:

[18] As the most obvious physical attacks unplug or reboot the systems, volatile encryption keys would be deleted (and can only be restored by a authorised administrator). There are however publications on very advanced techniques which thwart this defence mechanism.

[19] Such a policy typically includes a description of disaster protection and recovery, availability measures (redundancy of power, network, computer equipment), fire protection, etc.

- Physical access
  - Description of the environment (security aspects of walls, doors, computer racks, etc)
- Personnel access
  - Description of the Access Control system to the room. Access should be restricted to a well defined set of people and all access needs to be logged on a personal basis.
  - Access to the physical machines on which ACGT services are deployed should be overseen by people bound to the ACGT general terms.
- Surveillance
  - Description of alarm system and video surveillance present
  - Description of security guard surveillance
- Media management
  - Off site backup security (if this requires media to be taken out of the
  - Disposal of hardware policy(e.g. to ensure that no data accidentally leaks on for example a dumped HD).

4.4 Securing Network Access

Proper network security (i.e. at the network layer, infrastructure) is a prerequisite for securing systems from remote attacks. ACGT services should only be hosted in environments with a network infrastructure which:

- ***Must*** filter (at the transport layer) internet traffic with firewalls (e.g. dual/triple firewall setup, bastion host configuration) and logically separates the different data streams through (firewall controlled) subnetting and virtual LANs.
- ***Should*** filter incoming service requests (at the application layer) with an application firewall or reverse proxy (in a demilitarized zone (DMZ))
- ***Could*** have a Network Intrusion Detection System (IDS) running.

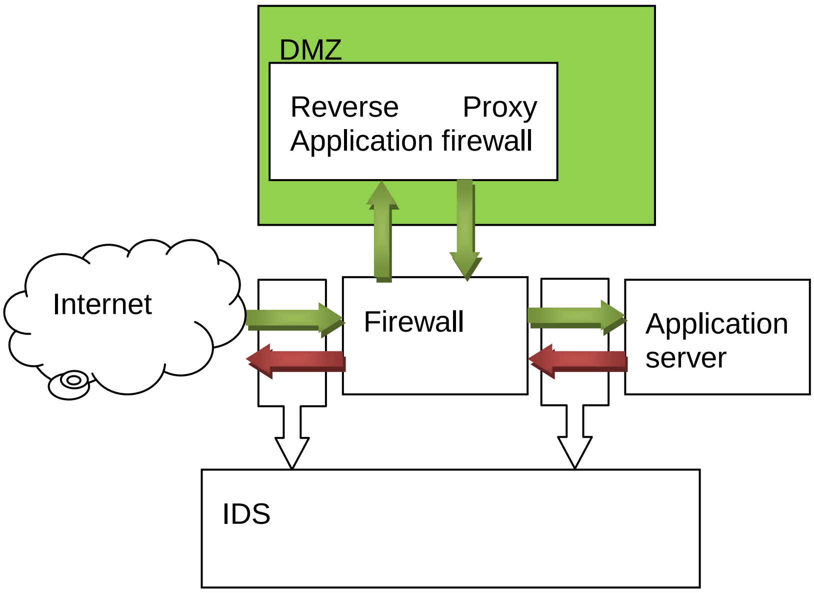

Figure 1: Secure network setup with bastion host

In Figure 1 an example secure setup is shown as it should be used for ACGT services (example with a bastion host firewall). Incoming connections to the service are routed through a firewall to the reverse proxy / application firewall in the DMZ. The reverse proxy / application firewall should perform filtering and scrubbing on the application level while the firewall filters do this on the network layer. If multiple application servers are deployed then the reverse proxy may also take care of load balancing and provide high availability services. The reverse proxy/application firewall continues the connection to the application server (again firewalled).

Firewalls should be configured to block all communication (both incoming and outgoing connections) by default. Communication required for services should be explicitly enabled. For ACGT services this generally includes:

- Connections from clients to the DMZ on the port used for the service
- Connections from the DMZ to the (internal) application server on the port used for the service
- Connections from the application server to
  - An internal DNS server (protocol UDP port 53)
  - An internal or the ACGT NTP server (protocol UDP port 123)
  - The ACGT PKI CRL and OCSP service (ca.custodix.com protocol TCP port 80)
  - The ACGT MyProxy repository if the service requires it (myproxy.custodix.com protocol TCP port 7512)
  - The ACGT Authorization service GAS (gas.custodix.com protocol TCP port 12355)

For extra security, outbound connections from the application server to external services can be routed through a local (forward) proxy. The IDS should be made part of the standard monitoring of the service. Irregular traffic can be detected by the IDS and well-known attack signatures are recognised by most IDS systems.

4.5 Audit and Logging

4.5.1 Introduction

The ACGT data protection framework lists auditing trails as one of its core components in protecting private data. These audit trails are reconstructed using the different logs created by various services in the ACGT infrastructure. A central audit bus has not been fully implemented in the course of the project (although all access decisions are logged in the central GAS service).

In order to be able to reconstruct a complete audit trails over multiple services, a number of requirements need to be met by the individual logs of ACGT services.

4.5.2 Information to be logged

For auditing purposes the “ACGT enabled” service should keep a record of the following information.

- Time of the operation or access: Knowing the exact time of an event is critical when reconstructing an audit trail. Correct timekeeping on all ACGT servers is necessary so that different events (as logged on different servers) can be placed in the correct order. In order to support this, an ACGT NTP server is provided. Alternatively, a different time source of choice may be employed, provided it is of high enough quality (stratum-2 or better).

  - If possible, time when the operation or access was completed: This provides a boundary within the current log for the audit trail.

- Unique identifier (which can be linked to the ACGT identity) for the ACGT User that requests the operation or access: The end user for whom the operation is to be executed or access granted. This information can be used for both repudiation and for constructing user-bound audit trails. The unique identifier must be identical on all services (ACGT identity) for any given user or an identity mapping service must be available for resolving user identities for each involved service provider.
- Authorization decision taken
- Unique identifier for the object of the authorisation request (a service, a dataset, etc) and the operation requested with any given parameters.
- Any external service called in order to complete the request.This can point to other service logs where the audit trail continues.

Logs created by “ACGT enabled” services must be archived for the log retention period determined by the ACGT Data Protection Authority. These logs must be stored in a secured environment and be part of the normal backup schedule. Given the fact that logs typically contain sensitive information, encryption of archived logs is encouraged.

4.6 Monitoring

4.6.1 Introduction

The ACGT environment is composed of a multitude of components, each of which can malfunction at unpredictable moments (either by accident or though an attack). Monitoring can not only detect existing problems with services, but also function as an early warning system. For example, by monitoring system resources such as CPU and memory, a heavily loaded system can be spotted before the load becomes problematic for the service it provides.

Monitoring relates to security in several ways. Systems that become unstable can become vulnerable to attack, monitoring allows to detect these weak spots (and thus allows for corrective measures). Equally, performance degradation or bizarre usage patterns can be an indication of an ongoing attack.

To help ACGT service administrators, an active monitoring service is provided that continuously performs health checks on the distributed components. This monitoring system is pluggable and can be extended with custom made checks.

4.6.2 Active Monitoring

Active monitoring consists of performing a predefined action on the service and then comparing the result received from that service with a predefined correct result. If the result of the service does not match the expected result then the service is considered to be in an erroneous or inconsistent state. In this case the administrator should be informed and the service shut down until the problem can be resolved.

The main advantage of active monitoring is that is can test a complex interaction between various services in one single test. Not only the service is tested but at the same time all services it depends on, including network connectivity, authorization services, etc.

On the other hand, due to the fact that active monitoring executes an actual service it adds extra load to the infrastructure. The service and services it depends on are using resources to test the health of the system instead of processing user requests. If active monitoring is used to often it can have a negative effect on the overall usability of the infrastructure. Therefore a good balance between running active tests and overall performance must be found. Further, side-effects caused by running test scripts may interfere with normal operations.

It is recommended that all ACGT services are connected to the ACGT provided monitoring system[20].

4.6.3 Passive Monitoring

Contrary to active monitoring, this type of monitoring does not execute an operation on the service. Information on the general health of the service is gathered from logs and system-level variables such as resource usage. Although this might not provide enough information to determine the exact state of the service, it can be a powerful tool in determining its overall health status.

The general health status of “ACGT enabled” systems should be monitored (this can be done centrally by a monitor system which collects all information from agents installed on the local machines).

4.6.4 Monitoring system resources

Through monitoring of a set of system resources several potential problems can be detected before they occur.

- **Disk space**: Insufficient disk space is a common reason for services and programs to fail. Being unable to create temporary files the service must either abort the request or it might even enter an inconsistent state. An early warning when too much of the file system has been used can prevent these problems. High disk usage can indicate lingering files that are not cleaned up by the service.
- **Memory and Virtual Memory**: Similar to disk space, memory is a crucial component for a service and might result in a crash if not enough memory is available. High memory usage can also indicate a memory leak in the service or very high load (too many simultaneous requests).
- **CPU usage** (system load): High CPU usage might not put a service in an error state but it will slow down response time to requests. This can cause timeouts on the client services causing operations to fail. A consistent high CPU usage can indicate an overloaded system.
- **Network usage**: Above normal network usage can indicated an overloaded network line.

[20] Information on integration of the monitoring system can be found online at: http://wiki.healthgrid.org/ACGT:Testbed/StatusMonitoring

Unexplainable pattern changes in these parameters might indicate attacks (especially Denial Of Service attack). Long time monitoring on the other hand will indicate trends (eg. slowly increasing load) and give the opportunity to update the system before a problem occurs due to resource exhaustion.

5 End-of-life of the service

5.1 Introduction

Legislation and local policies typically imply security requirements that go beyond the lifetime of a service. In particular, destruction of sensitive data and archiving of access logs are high-priority tasks after discontinuation of a service.

5.2 Removal of the Service

All credentials assigned to a service that has reached End of Life status must be revoked immediately to prevent misuse. The Certificate Authority (CA) must be notified in advance of the date and time when the service will be discontinued.

All references within the ACGT infrastructure to the discontinued service should be removed or updated to indicate its discontinued state, this includes service repositories and links to the service through the portal site.

5.3 Cleanup and Archiving

All data stored by the decommissioned service in the context of ACGT must be destroyed. If this data also resides in backups it should be deleted there as well, if possible (and otherwise it must phase out according to a normal backup rotation scheme).

Care must be taken that the deletion is properly executed such that data can no longer be recovered from the physical devices on which it was stored. A policy for secure hardware decommissioning is advised.

Logs and audit records must be archived and stored securely according to local policy, legislation and best practice. Legislation and best practice indicate that logs and audit files must be retained for a certain time after the service has been discontinued. The retention period of log files is specified by the ACGT Data Protection Authority.

The ACGT DPA should be notified that all relevant data has been destroyed upon decommissioning of a service.

6 “ACGT enabled” compliance checklist

#	Category	Description	Compliance	Remarks
1	Secure Communication	All communication over untrusted networks (Internet) must rely on an application or transport protocol supporting confidentiality and integrity (e.g.SSLv3/TLSv1 transport layer protocol).	MANDATORY	See http://www.sans.org/top25-programming-errors/ or http://cwe.mitre.org/top25/
2	Secure Communication	All communication must be mutually authenticated.	MANDATORY	E.g. when using SSLv3/TLSv1, turn on client authentication in your application server configuration.
3	Authentication	Deployed services shall accept certificates issued (indirectly) by the ACGT root CA.	MANDATORY	The ACGT CA root certificate is included in the following bundle: https://acgt.custodix.com/Acgt-ca.tar
4	Authentication	Developed services must be capable of performing revocation checking based on the CRL mechanism provided by the ACGT PKI.	MANDATORY	The current CRL can be downloaded at http://ca.custodix.com/crl/crlStore?issuer=cn=ACGT%20CA%202006,ou=Primary,o=Custodix,c=BE
5	Authentication	Developed services could use the OCSP service provided by the ACGT root CA.	ADVISED	The OCSP responder is located at http://ca.custodix.com/ocsp/responder
6	Authentication	Developed services should accept proxy certificates generated based on ACGT X.509 certificate or other proxy certificates derived hereof.	ADVISED	See http://www.ietf.org/rfc/rfc3820.txt
7	Authentication	Developed services should be capable of requesting new proxy certificates from the ACGT MyProxy service (based on a provided username and password).	ADVISED	See http://grid.ncsa.illinois.edu/myproxy/
8	Authorization	Developed services must rely on GAS for authorization decisions.	MANDATORY	The ACGT GAS service is available at https://gas.custodix.com:12366

#	Category	Description	Compliance	Remarks
9	Authorization	Developed services must issue (pro forma) authorization requests to the ACGT GAS instance even when public resources are being accessed.	MANDATORY	
10	Logging	Access to deployed services must be logged and archived in such a format that reconstruction of the path can be done. This requires that at least the identity of the end user, the action performed by the service and the exact time of this action is logged. .	MANDATORY	
11	Logging	Access logs must be archived in a secured location for the retention period specified by the ACGT Data Protection authority.	MANDATORY	
12	Logging	Archived logs could be encrypted.	ADVISED	
13	Secure Application Development & Deployment	Operations in support of administrative or testing purposes must be separated from regular application operations using separate logical or physical interfaces.	MANDATORY	
14	Secure Application Development & Deployment	Administrators of computer systems on which ACGT enabled services run should be bound by the “ACGT General Terms”, just as the users of these services.	MANDATORY	The ACGT General Terms are specified in Deliverable D.10.1.
15	Secure Application Development & Deployment	(System hardening, privilege management) Deployed services should run under restricted system accounts.	ADVISED	

#	Category	Description	Compliance	Remarks
16	Secure Application Development & Deployment	(System hardening, privilege management) Deployed services should use fine grained access control capabilities to restrict access to operating system resources as much as normal service usage patterns permit.	ADVISED	
17	Secure Application Development & Deployment	(System hardening, minimization) Services should be deployed on a system that has been set up according to a minimal installation profile (no default install) or on which an installation downsizing script was run.	ADVISED	
18	Secure Application Development & Deployment	(System hardening) Services should be deployed on systems that have been hardened, preferably using automated hardening tools.	ADVISED	
19	Secure Application Development & Deployment	Unrelated deployed services should run in isolated physical or virtual system environments.	ADVISED	
20	Network Operations	(Physical access) Services must be deployed on servers that are physically located in a server room, secured according to the state-of-the-art..	MANDATORY	
21	Network Operations	(Physical access) A clearly described security policy must be available for this server room as described in section 4.3. Access restriction should be clearly specified (access to the physical machines on which ACGT services are deployed should be overseen by people bound to the ACGT general terms.).	MANDATORY	

#	Category	Description	Compliance	Remarks
22	Network Operations	(Network Filtering) Internet traffic must be filtered with firewalls (e.g. dual/triple firewall setup, bastion host configuration). Data streams must be logically separated through (firewall controlled) subnetting and virtual LANs.	MANDATORY	
23	Network Operations	Deployed services must rely on a secure and accurate time source (equivalent to NTP stratum-2 or better). Use of the ACGT NTP service is advised.	MANDATORY	The ACGT NTP service is located at 193.121.186.40 on port 123
24	Network Operations	(Network Filtering) Incoming service requests (at the application layer) should be filtered with an application firewall or reverse proxy (in a demilitarized zone (DMZ))	ADVISED	
25	Network Operations	Access to deployed services could be monitored by an Intrusion Detection System (IDS).	ADVISED	
26	Monitoring	Deployed services should be monitored using the central ACGT monitoring service (scripts for active monitoring should be provided).	ADVISED	Available at http://moss1.man.poznan.pl/gridsphere/gridsphere
27	Monitoring	Services could be deployed on systems where agents collect information on system resource usage and configuration changes, to be used as input for the central ACGT monitoring service.	ADVISED	
28	End-of-life management	Credentials issued to a discontinued service must be revoked immediately upon discontinuation.	MANDATORY	
29	End-of-life management	Upon service discontinuation, all data stored by the service in the context of ACGT must be destroyed. If this data also resides in backups it should be deleted there as well, if possible (and otherwise it must phase out according to a normal backup rotation scheme).	MANDATORY	

#	Category	Description	Compliance	Remarks
30	End-of-life management	Logs and audit records must be archived and stored securely according to local policy, legislation and ACGT Data Protection Authority.	MANDATORY	
31	End-of-life management	The ACGT DPA should be notified that all relevant data has been destroyed upon decommissioning of a service.	ADVISED	

7 References

[1] CWE/SANS. CWE/SANS TOP 25 Most Dangerous Programming Errors. [Online] [Cited: 24 04 2009.] http://www.sans.org/top25errors/ .

[2] SANS website. SANS Institute. [Online] [Cited: 17 04 2009.] http://www.sans.org .

[3] http://java.sun.com/javase/6/docs/technotes/guides/security/certpath/CertPathProgGuide.html#AppC

[4] Network Working Group. Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. [Online] 6 2004. [Cited: 23 04 2009.] http://www.ietf.org/rfc/rfc3820.txt

[5] PSNC. GRIDGE. [Online] [Cited: 24 04 2009.] http://www.gridge.org

[6] Puschitz, Werner. Securing and Hardening Red Hat Linux Production Systems. http://www.puschitz.com/ . [Online] 2007. [Cited: 15 04 2009.] http://www.puschitz.com/SecuringLinux.shtml .

[7] http://www.sun.com/software/security/jass/

[8] http://www.sun.com/software/solaris/ds/trusted_extensions.jsp

[9] http://www.grsecurity.net/

[10] http://www.nsa.gov/research/selinux/index.shtml

[11] Microsoft. Applying the Principle of Least Privilege to Users Accounts on Windows XP. [Online] 18 01 2006. [Cited: 16 04 2009.] http://technet.microsoft.com/en-us/library/bb456992.aspx .

[12] Rich, Amy. The Least Privilege Model in the Solaris 10 OS. [Online] Sun Microsystems, 02 2005. [Cited: 16 04 2009.] http://www.sun.com/bigadmin/features/articles/least_privilege.jsp

[13] Sun Microsystems**.** Using ACLs to protect ZFS Files. *Solaris ZFS Administration Guide.* [Online] [Cited: 16 04 2009.] http://docs.sun.com/app/docs/doc/819-5461/ftyxi?a=view .

[14] Stuart E. Madnick, John J. Donovan. An approach to information system isolation and security in a shared facility. [Online] 03 1973. [Cited: 20 04 2009.] http://dspace.mit.edu/bitstream/handle/1721.1/1864/SWP-0648-14442548.pdf?sequence=1

[15] http://en.wikipedia.org/wiki/Virtual_machine

8 Appendix - Abbreviations and acronyms

ACL	Access Control List
API	Application Programming Interface
CA	Certificate Authority
CRL	Certificate Revocation List
DMZ	DeMilitarized zone
DNS	Domain Name System
GAS	Gridge Authorization Service
IDS	Intrusion Detection System
MyProxy	MyProxy certificate repository
NTP	Network Time Protocol
OCSP	Online Certificate Status Protocol
OS	Operating System
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PKI	Public Key Infrastructure
SOA	Service Oriented Architecture
VMM	Virtual Machine Manager
VO	Virtual Organisation