Grid security infrastructure based on Globus Toolkit

Valentin Vidić
vvidic@irb.hr
Center for Informatics and Computing
"Ruđer Bošković" Institute
Bijenička cesta 54, Zagreb, Croatia

January 2006

Abstract - Grid resources and users are distributed and often belong to different organisations. In order to establish trust in this complex environment some extensions to security standards (SSL, X.509, GSS-API) are required. The most important extension is the introduction of proxy certificates that allow single sign-on access to Grid resources. Grid PKI is also one of a few large scale PKI implementations so it is interesting to see how it is organised and operated. Finally, a Globus GSS-API implementation for securing applications is described.

I. INTRODUCTION

Most of today's Grids consist of worldwide distributed resources connected over the Internet. These resources act as a service for dispersed groups of users belonging to different administrative domains. Establishing trust in this environment is essential as the resources involved (computational and storage clusters, measurement equipment) are often very expensive. One of the first integrated solutions for the problem of Grid security (most notably authentication) is Globus Toolkit[1].

This article presents Grid security implemented by Globus Toolkit from three different aspects. The first part defines the security tokens used for Grid authentication - proxy certificates. This is followed by a description of trust management based on Certification Authorities. Finally, the security interface used by applications is presented. All the descriptions in this article are based on the Globus Toolkit version 3 as distributed in Virtual Data Toolkit[2]. Although newer versions are available, this one was chosen because it is in common use throughout production Grids of today.

II. PROXY CERTIFICATES

Proxying is a common technique used to allow one entity to act on behalf of another. Grid services use proxy certificate to authenticate and perform work on behalf of users. X.509 proxy certificate is a certificate signed by a normal X.509 end entity certificate or another proxy certificate. It is clear that this is in violation with the classic PKI model[3] where only CAs are allowed to sign certificates. Therefore some changes in the certificate verification procedure are also needed.

The two major benefits of using proxy certificates in the Grid are single sign-on (SSO) and delegation. Single sign-on gives the user access to all Grid services by entering his certificate private key password only once (at the beginning of the session). At this point a proxy certificate with a short lifetime (10 hours by default) is generated. All further accesses to Grid services use the proxy certificate to authenticate the user without asking for password. This is possible because the proxy certificate private key is stored unencrypted in a file accessible only by the user. Delegation, on the other hand, allows the user to issue a proxy certificate to a remote Grid service. Grid service can then authenticate acting as the user and perform work on his behalf (execute jobs, transfer files).

Two formats of proxy certificates are in common use today: Globus Toolkit version 2 (GT2) and version 3 (GT3) style proxies. The main difference between the two is the way how proxy type is encoded. In GT2 proxy type is determined from the proxy certificate subject, while GT3 proxies use a special *ProxyCertInfo* extension. Globus Toolkit version 2 recognises two proxy types:

- full,
- limited.

Full proxy is recognised by the final *CN=proxy* element in the proxy certificate subject. The name for this proxy type comes from the fact that the proxy has the same rights as the original user. In contrast, GT2 limited proxy has *CN=limited proxy* in the subject. Rights assigned to this proxy are in some way limited in comparison to full proxy. In practice only Globus gatekeeper service recognises and denies access to this type of proxy. This means limited proxies are allowed to transfer files and access other services but they cannot submit jobs. The consequences of a compromised limited proxy are therefore limited because the attacker can't access other clusters by submitting jobs. Limited proxy also prevents denial of service attacks created by jobs submitting new jobs.

Globus Toolkit version 3 introduced the following proxy types:

- impersonation,
- limited,
- independent.

Impersonation and limited proxies are equivalent to full and limited proxies in GT2, respectively. The name of the full proxy was changed in order to show that full proxy impersonates the user - it has the same access rights as the user issuing it. Independent proxy type is introduced for cases when the proxy is assigned rights independent of the issuer. Independent proxy needs to have unique subject (usually a random number as the last element) because rights are usually assigned by certificate subject. Types for GT3 proxies are encoded with different OIDs in *policy* element of *ProxyCertInfo* extension. Apart from defining proxy type, this extension can be used to limit the delegation by specifying a maximum length of proxy chain (*path length constraint*). Although GT3 style proxies have been standardised in form of RFC 3820[4], most of the proxies used today are of GT2 type.

III. CERTIFICATE MANAGEMENT

For easier management, Grid CAs are usually established per country and organised in geographically based trust federations called Policy Management Authorities (PMA):

- APGridPMA in Asia,
- EUGridPMA in Europe,
- TAGPMA in Americas.

Federation defines minimum requirements (profile) for all member CAs[5]. This includes operational procedures (user identification, certificate and CRL publishing) and security controls (physical safety, signing procedures, auditing). Minimum requirements are used for checking new CA compliance during accreditation but also for relying parties (deferent Grid projects) to determine if the federation CAs can be trusted for a particular purpose.

CA defines its own procedures in form of Certificate Policy (CP) and Certification Practice Statement (CPS). Certificate policy is defined as a set of rules that indicate the applicability of a CA issued certificates for a particular community or a class of application with common security requirements. Certification Practice Statement is a statement of the practices that a certification authority employs in issuing, managing, revoking, and renewing or rekeying certificates. In other words, CP is a general definition of roles for all the participants (CA, RA, users, relying parties). CPS, on the other hand, defines how these roles are to be implemented in terms of procedures and security controls.

In practice, CP and CPS are merged in a single document based on the framework defined in RFC 3647[6]. Such CP/CPS usually has the following outline:

1. Introduction - defines the given CA and its scope (applicability),

2. Publication and repository - defines the parameters of the certificate repository (usually web based),
3. Identification and authentication - procedures defining identification and authentification of users,
4. Certificate life-cycle operational requirements - management of certificates,
5. Facilities, management and operational controls - non-technical controls implemented by the CA,
6. Technical security controls - protection and management of private keys,
7. Certificate, CRL, and OCSP profile - format of the certificates and CRLs issued,
8. Compliance audit - procedure for assessment of compliance,
9. Other business and legal matters - service fees and legal responsibilities (personal privacy, intellectual property, warranties, liability, terms etc.)

Typical CA issues certificates for period of one year. Certificate signing is done offline to protect the CA private key. Signing machine is allowed to be online only if the CA private key is stored in a hardware security module. CRL issuing is done at least once per month or more often in case of certificate compromise.

Candidate CA is evaluated by two accredited members in a peer-review process. After the CP/CPS has been reviewed, CA is presented on the next PMA meeting. Accreditation follows after possible remaining issues are addressed. Operations of accredited CAs are periodically audited to check if it still complies with the CP/CPS. Examples of CP/CPS documents can be seen in [8] and [9].

Accredited CAs from all the federations are made available as a distribution in several different formats by the International Grid Trust Federation (IGTF). IGTF also coordinates the namespace of certificate subjects and serves as central point of contact for certificate related incidents. Typical CA certificate package distributed by IGTF consists of the following files:

- CA certificate,
- CA namespace policy,
- URL for CRL download,
- additional information about the CA.

Basename for all the files related to a single CA is an MD5 hash of the CA subject condensed to 32 bits (e.g. *ff94d436* for SRCE CA). This hashed naming scheme allows fast lookup by CA name. This speeds up verification because it is not necessary to scan the whole directory to find a given CA and allows updates to CA list while the application is running. Although improbable, it is possible that two CAs produce the same hash, so a number acting as hash bucket index is appended to the file extension (e.g. *ff94d436.0* is used for the first CA with hash *ff94d436*). The CA certificate is stored in a file without a special extension (e.g. *ff94d436.0* where *0* is the hash bucket index), usually in PEM format.

Namespace policy files define the namespace the CA is allowed to use when signing certificates. Namespace is usually specified using regular expressions with a constant prefix. For example, regular expression *"/C=HR/.*"* in the policy rule matches all certificates signed by given CA whose subject starts with *"/C=HR/"*. Original format for namespace policies is Globus Extended ACL (EACL). Such policy is stored in a file with the extension *signing_policy* (e.g. *ff94d436.signing_policy*). Policy in newer namespace constraints file format[7] is distributed in files with extension *namespaces* (e.g. *ff94d436.namespaces*). At the moment only the Globus EACL format is used for checking namespace policies.

Certificate Revocation List (CRL) lists serial numbers of revoked certificates and is digitally signed by the CA. Since CRL changes over time, it is not distributed in the CA package. Instead, a file with the extension *crl_url* (e.g. *ff94d436.crl_url*) contains the URL where the CA publishes latest CRL. New CRL is periodically downloaded, verified and installed into file with *r* extension (e.g. *ff94d436.r0* where *0* is the hash bucket counter).

Some additional information about the CA is included in the *info* file (e.g. *ff94d436.info*), for example: CA accreditation status and version, aliases and dependencies between CAs (for multi level

CAs), official CA e-mail and URLs pointing to CA certificate, CRL and RA web. As most of this is already included in the CA certificate, *info* file is relevant and used only for PMA management.

IV. APPLICATION SECURITY

Applications use the GSI through Generic Security Services Application Program Interface[10] (GSS-API). Unlike most other GSS-API implementations that use Kerberos as the backing mechanism, Globus GSS-API is based on X.509 certificates and SSL protocol. This section describes the most commonly used GSS-API C functions[11] and their Globus implementation. Security session using GSS-API is usually a sequence of the following function calls:

- *gss_acquire_cred* for loading certificates,
- *gss_init_sec_context* or *gss_accept_sec_context* to establish a secure session,
- *gss_wrap* and *gss_unwrap* for secure exchange of application data,
- *gss_delete_sec_context* for closing the session.

As the name suggests, *gss_acquire_cred* is used to acquire security credential (certificate) to be used in the session. Globus GSI library searches for a valid certificate in the filesystem using the following search order:

- service certificate
- host certificate
- user proxy certificate
- user certificate

After a valid certificate is read, client and server proceed to establish a security context with requested parameters. This process is asymmetric in the sense that client initiates the communication by calling *gss_init_sec_context.* The function generates an output token (buffer) and it is the responsibility of the client to transport it to the server, most probably over some kind of network socket. This makes GSS-API independent of the underlying communication mechanism. Server application accepts the connection from the client, receives the initial token and passes it as input to *gss_accept_sec_context.* This function also generates a response in form of an output token. Server forwards this token to the client and this token exchange continues until the context is fully established (indicated by the return value of functions).

Token exchange between client and server can be split into two phases:

- SSL handshake,
- proxy delegation.

During SSL handshake, client and server exchange standard SSL messages starting from *Client Hello* up to *Finish*[13]. As expected, *Server Certificate* message contains the (proxy) certificate of the server, *Certificate Request* lists the CAs server recognises and *Client Certificate* contains the client's proxy certificate. Although there are no changes in the protocol itself, most of the standard available SSL libraries would not support this exchange due to use of proxy certificates. Standard SSL libraries recognise only two types of certificates:

- CA certificates (possibly multilevel),
- End entity certificates (EEC) - host or user certificates.

Two types are distinguished by the X.509 critical extensions which allow a CA certificate to sign user certificates (or other CA certificates). When a standard SSL library tries to verify a proxy certificate it sees an EEC signed by another EEC. Since this is not allowed in the classical PKI model, verification of the whole certificate chain fails. In order to support proxy certificates, Globus GSS-API implementation overrides the certificate verification rules of the underlying SSL library (OpenSSL). In addition to standard certificate checks (lifetime and digital signature), verification of proxy certificate chains introduces the following changes:

- *key usage* critical extension check is ignored in user and proxy certificates to allow proxy certificates to be verified,
- mixing proxies of different types (GT2, GT3) is not allowed,

- CRLs are checked for revocation of user and CA certificates,
- signing policy is checked for user and respective CA certificate,
- critical extensions for proxy certificates are checked (maximum proxy chain length, proxy policy),
- maximum length of non proxy certificate chain is checked (if defined in the CA certificate).

If the SSL handshake finishes successfully, peers proceed with the delegation protocol[12]:

1. Client sends character *D* if it wants delegation. If delegation was not requested by the application, client sends *0* and the session goes into established state.
2. If delegation is requested, server generates a certificate request and sends it to the client.
3. Client checks the request and if no problems are found, signs the request and sends the new proxy certificate to server.

Certificate request and signed proxy are exchanged using DER encoding. Note that private key is never transfered over the network (it always stays on the server) so the delegation protocol is safe even if used over unencrypted connection. Message integrity is, however, required in order to protect against connection hijacking.

Application can influence the properties of the session through a set of flags passed to session establishment functions. Available flags are:

- anonymity - client certificate is not passed to server,
- confidentiality - strong encryption algorithm is requested,
- delegation - delegate full proxy,
- limited delegation - delegate a limited proxy,
- SSL compatibility - delegation is skipped altogether,
- limited proxy check - whether to accept peer with limited proxy,

If confidentiality flag is not set, encryption is disabled by setting *NULL* (none) cipher as the preferred one in the first SSL message (*ClientHello*). As a result, only integrity protection might be available in the subsequent calls to *gss_wrap* (server might not accept proposed *NULL* cipher). Also note that some flags can't be used together. For example, it is not possible to delegate a proxy and remain anonymous; SSL compatibility is in conflict with delegation because delegation protocol is not defined in SSL standard.

After the secure session is established, client and server call *gss_wrap* to protect data for transfer. Depending on the parameters this can include both integrity and confidentiality protection or just integrity protection. If *NULL* cipher was selected during session establishment, data will have Message Authentication Code (MAC) appended. If some other cipher was selected, data will first have MAC appended and then both data and MAC are encrypted using the selected algorithm. MAC protection is a secure hash of several components:

- secret session key - sender authentication,
- application data - protection against changes,
- sequence number - protection against replay.

Output token returned by this function is a standard SSL application data record. Receiving end passes this token to *gss_unwrap* function for possible decryption and MAC verification. Function returns a buffer containing original data and flag indicting type of protection used.

To close the session, either side can call *gss_delete_sec_context*. This function releases the allocated data structures and returns a final token containing an SSL close notification alert message used to inform the other end of the session termination.

V. CONCLUSION

Globus GSS implementation described in the previous section shows some problems with the GSS-API specification. Since GSS-API was originally designed with Kerberos mechanism in mind, some Grid use-cases are not well covered by the specification. For example, GSS-API does not allow the server to influence the parameters of the session (there is no input flags parameter). Globus works around this by passing some input flags through output flags parameter. Furthermore, GSS-API allows delegation to happen only at the start of the session. In some cases (long running jobs) association with the Grid service can last longer than the usual proxy lifetime and some mechanism of repeated delegation needs to exists. GSS-API also does not define functions for importing and exporting credentials to files, only direct interprocess exchange is defined. Some of the extensions implemented in GSI are described in [14] and might be included in future versions of GSS-API specification. In some sense, GSI implementation of the GSS interface was a good test of the specification that was intended to be general but in practice only implemented with one backing mechanism - Kerberos.

In comparison to Kerberos realms, Grid PKI infrastructure is somewhat more lightweight. This is mostly due to the fact that CAs currently operate in offline mode. CA server is contacted only a few times a day to download the fresh CRLs, while most Kerberos operations require the Kerberos server to be online all the time. Proxy generation is also an offline operation, as is user/service authentication. Creation of Kerberos tickets and user/service authentication, on the other hand, require a working Kerberos server. This can create a high load on the Kerberos servers and Grid experience show that such centralised services often lead to reliability problems. In this case, malfunctioning Kerberos server can break authentication for some users or services making parts of the infrastructure unavailable. Things get even more complicated as the number of involved organisations increase (current IGTF distribution includes around 70 CAs). In Kerberos this would require establishing some sort of online cross realm trust relationship, while Grid PKI manages this in an offline manner. PMAs establish trust with new CAs and distribute the new list of accredited CAs on monthly bases. All of this suggest that PKI is a better suited for authentication in large loosely coupled distributed systems like the Grid.

References

[1] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Cajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, S. Tuecke, “Security for Grid Services”, *High Performance Distributed Computing*, 2003.

[2] J. Gawor, S. Meder, F. Siebenlist, V. Welch, “GT3 Grid Security Infrastructure Overview”, 2003.

[3] R. Housley, W. Polk, W. Ford, D. Solo, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, *RFC 3280*, 2002.

[4] S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson, “Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile”, *RFC 3820*, 2004.

[5] D. Groep, “Profile for Traditional X.509 Public Key Certification Authorities with secured infrastructure”, 2005.

[6] S. Chokhani, W. Ford, R. Sabett, C. Merrill, S. Wu “Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework”, *RFC 3647*, 2003.

[7] D. Groep, “Namespaces Format Specification”, 2006.

[8] D. Dobrenić, E. Imamagić, J. Ivankov, “SRCE Certification Authority Certificate Policy and Certification Practice Statement”, 2006.

[9] K. Christos, “SEE-GRID Certification Authority Certificate Policy and Certification Practice Statement”, 2004.

[10] J. Linn, “Generic Security Service Application Program Interface Version 2, Update 1”, *RFC 2743*, 2000.

[11] J. Wray, “Generic Security Service API Version 2 : C-bindings”, *RFC 2744*, 2000.

[12] V. Welch, “Grid Security Infrastructure Message Specification”, 2004.

[13] A. Freier, P. Karlton, P. Kocher, “The SSL Protocol Version 3.0”, *Netscape Communications Corp.*, 1996.

[14] S. Meder, V. Welch, U. Chicago, S. Tuecke, D. Engert, “GSS-API Extensions”, 2003.