SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2005; **00**:1–17 *Prepared using* speauth.cls *[Version: 2002/09/23 v2.2]*

The MyProxy online credential repository

Jim Basney[1,*,†], Marty Humphrey[2], and Von Welch[1]

[1] *National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, 605 E. Springfield Ave., Champaign, IL 61820, USA*
[2] *Department of Computer Science, University of Virginia, 151 Engineer's Way, Charlottesville, VA 22904, USA*

SUMMARY

The MyProxy online credential repository has been used by the grid computing community for over four years for managing security credentials in the grid public key infrastructure. MyProxy improves usability by giving users access to their credentials over the network using password authentication, allowing users to delegate their credentials via web browser interfaces to the grid, and supporting credential renewal for long-running jobs. MyProxy helps administrators secure users' private keys by providing an online service from which users retrieve short-lived credentials without distributing long-lived keys to potentially vulnerable end-systems. This paper describes the MyProxy system and its use.

KEY WORDS: grid computing, credential management, public key infrastructure, virtual smart card

INTRODUCTION

Grid computing is fundamentally about computing across administrative domains. The requirement for cross-domain security sets grid computing apart from many other distributed computing scenarios. Security technologies that require a single source of authority, for maintaining passwords or generating trusted credentials, can not scale to meet the needs of the grid community. For this reason, grids today use a public key infrastructure (PKI) with multiple certification authorities (CAs), trusted by relying parties to sign certificates attesting to a user's grid-wide identity.

*Correspondence to: 605 E. Springfield Ave., Champaign, IL 61820, USA
†E-mail: jbasney@ncsa.uiuc.edu
Contract/grant sponsor: National Science Foundation; contract/grant number: ANI–02–22571
This is a preprint of an article accepted for publication in *Software: Practice and Experience.*

The integrity of the user's identity credential (certificate and public/private key pair) is paramount for maintaining trust in the user's authenticated identity across the grid. When accepting a certificate signed by a trusted CA, a relying party believes that the CA has followed appropriate, agreed-upon procedures for identifying key holders when issuing certificates and has maintained the secrecy of the CA's signing key. Additionally, the relying party must be confident that the user's private key has remained secret, so that no one can impersonate the user with a copy of the private key. If the secrecy of the user's private key is compromised, we can only hope that the compromise is detected quickly so the certificate can be revoked and the user can be issued a new identity credential before much damage is done by impersonators.

To minimize the risk of compromise, we need systems and procedures that help users keep their private keys private. It is common for users to store private keys on their desktop systems. In this case, we can encourage users to use good local key management software that encrypts keys, requiring users to choose good passwords for encryption, and restricts access to the keys, so users must give their approval when their keys are used. However, given the limited security of typical desktop systems, we can expect the keys to be vulnerable to theft by keystroke loggers, trojans, viruses, or other software that circumvents the local key protections. Desktop systems are often under limited control of and monitoring by security personnel, making it more difficult to protect against and detect these attacks.

Storing keys on smart cards can provide improved security over desktop key storage, because the keys remain protected on the hardware device. The cards provide a restricted communication interface that allows keys to be used for signing and encryption without allowing the keys to be exported from the card's protected memory, except in specifically allowed, restricted cases. Users can easily detect the physical theft of the card and report the compromise to the CA. The main drawbacks of smart cards are cost and support. In addition to the cost of purchasing cards and readers, the organization must handle lost or forgotten cards and ensure compatibility across the various computing platforms in use (readers, operating systems, applications).

Sandhu et al. [1] use the term "virtual smart card" to describe software systems that protect private keys on secure servers without allowing the keys to be exported, thereby providing secure key storage analogous to physical smart cards. Benefits of virtual smart card systems include centralized usage monitoring, because the secure server(s) participate in all private key operations, and instant revocation, because keys can not be used once they have been removed from the secure server(s). Virtual smart card systems also provide better support for mobility than keys on the desktop or physical smart cards: users can access the virtual smart card over the network from different desktop systems, without needing to manually copy local key stores between systems, and the virtual smart card is not restricted to only those systems supporting physical smart card readers and can be accessed with a password the user remembers versus a physical card the user may forget at home.

The MyProxy online credential repository is an example of a virtual smart card system. MyProxy was developed to meet credential management requirements for grid computing, using the Globus Toolkit [2] grid middleware, and has been widely used by the grid community since its first release in 2000 [3], becoming an important component of major grids including the NEESgrid, TeraGrid, EU DataGrid, and the NASA Information Power Grid. The MyProxy

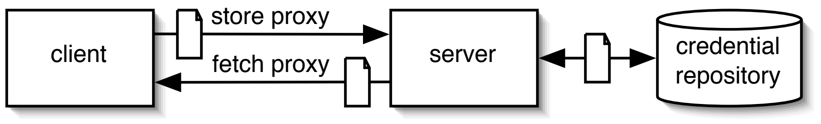

Figure 1. An overview of the MyProxy system.

project has continued development and support of the open source MyProxy software[†] over the past four years, extending the MyProxy system to provide additional capabilities. This paper presents the current MyProxy system design and documents how the MyProxy system has been used over that time.

The rest of the paper is organized as follows. In the next section, we describe the MyProxy protocol and system design. Then, we present four applications of the MyProxy system for grid computing: mobility, renewal, delegation, and enrollment. After that, we describe our recent work to re-cast the MyProxy system as a web service in accordance with the emerging Open Grid Services Architecture. We then present related work and conclude the paper.

SYSTEM DESIGN

MyProxy is a client-server system, where clients can store credentials, with access control policies, in an online repository for later retrieval as illustrated by Figure 1. MyProxy uses X.509 proxy certificates [4] to support storing and retrieving credentials without exporting private keys. Users typically store long-lived credentials, with lifetimes in the range of weeks to years, in the MyProxy repository and retrieve short-lived credentials, with lifetimes of one day or less, for grid computing sessions, so the user's long-lived credentials remain protected on the MyProxy server.

MyProxy client and server software is implemented in the C programming language, in approximately 6000 lines of code, using the GSSAPI [5] library provided by the Globus Toolkit, which in turn uses the OpenSSL library for the TLS protocol [6] implementation and X.509 certificate handling [7]. The MyProxy implementation is included in the U.S. National Science Foundation Middleware Initiative GRIDS Center software distribution. MyProxy clients have also been implemented in Java for the Java CoG Kit [8].

Core protocol

The core MyProxy protocol proceeds as follows. The client establishes a TCP connection to the server and initiates the TLS handshake protocol. The server must authenticate to the

[†]Available for download from http://myproxy.ncsa.uiuc.edu/.

client via TLS with an identity of “fqhn”, “host/fqhn”, or “myproxy/fqhn”, where “fqhn” is the fully-qualified hostname of the server. The client may also authenticate to the server via TLS, but to allow clients without existing X.509 credentials to retrieve credentials from the MyProxy server, the server does not require client-side TLS authentication. (Server access control mechanisms and policies are described in further detail in a later section.)

The client and server establish a private (encrypted) TLS channel for encapsulation of the MyProxy application protocol. Once the TLS handshake is complete and the secure channel has been established, the client sends one byte with zero value over the TLS channel, to indicate that the client is not delegating a credential at this stage, as part of the Globus Toolkit GSSAPI protocol [9].

The client then sends a request message to which the server replies. The messages are formatted as NULL-terminated ASCII text strings containing one or more lines of ATTRIBUTE=VALUE statements, separated by the ASCII newline character 0x0a (‘$\backslash n$’). Each message begins with VERSION=MYPROXYv2 to identify the protocol version. The client’s request follows with:

```
COMMAND=<ASCII decimal integer>
USERNAME=<ASCII string>
PASSPHRASE=<ASCII string>
LIFETIME=<ASCII decimal integer>
```

followed optionally by additional ATTRIBUTE=VALUE lines. The COMMAND line specifies the requested command:

0. Retrieve a proxy credential.
1. Store a proxy credential.
2. Retrieve information about stored credentials.
3. Remove a stored credential.
4. Change the password protecting a stored credential.

The USERNAME line specifies an ASCII string identifying the account for storing the credential. This account need not correspond to any system account outside MyProxy. The PASSPHRASE line specifies an ASCII password used to protect the stored proxy credential. The LIFETIME line specifies the lifetime of the retrieved proxy credential in seconds (ASCII digits).

The server replies with:

```
RESPONSE=0
```

to indicate success or:

```
RESPONSE=1
ERROR=<error text>
ERROR=<error text>
...
```

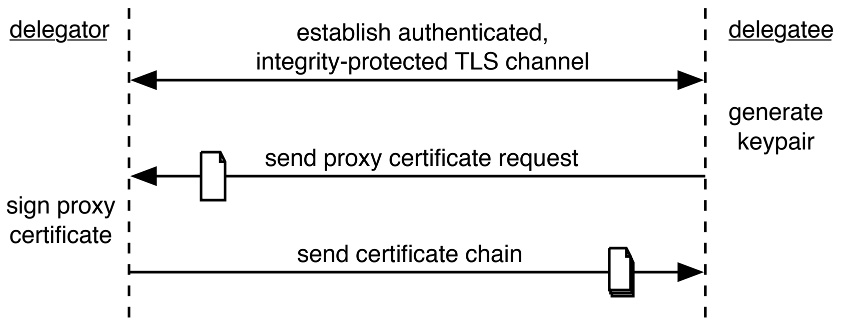
delegator
delegatee
establish authenticated, integrity-protected TLS channel
generate keypair
send proxy certificate request
sign proxy certificate
send certificate chain

Figure 2. The proxy delegation protocol.

to indicate error, with one or more lines of human readable error text.

The simple ASCII ATTRIBUTE=VALUE format for these messages has allowed us to add new features to the MyProxy protocol over time while retaining backwards compatibility, by adding new attributes that are ignored by MyProxy versions that do not understand them. For example, we have added an optional CRED_NAME attribute to support storing multiple credentials with different names in a single account.

Proxy certificates

Proxy certificates [4] enable MyProxy clients to retrieve credentials from the repository without exporting private keys from the repository. Proxy certificates are derived from X.509 end entity certificates and signed by the corresponding end entity private key, to provide restricted proxying and delegation. Proxy certificates can also be derived from other proxy certificates, resulting in a certificate chain containing one or more proxy certificates, an end entity certificate, and one or more CA certificates.

Proxy credentials (certificate and public/private key pair) serve as short-lifetime (ex. 12 hour) session credentials, derived from long-lifetime (months or years) end entity credentials, for use with the Globus Toolkit [9]. The proxy credential's private key is typically stored unencrypted on the local filesystem, readable only by the owner, for use during a session, to support single sign-on. The lifetime of the credential, set by the validity dates in the certificate, is set to a short period to limit the potential damage resulting from a stolen proxy credential. One common use of MyProxy is retrieving a proxy credential at the start of a session.

Delegation

To obtain a proxy credential, the client sends the request (with COMMAND=0), indicating the username and password for the credential, the requested credential lifetime, and other optional attributes such as credential name. If the server gives a successful reply, MyProxy performs the proxy delegation protocol as implemented in the Globus Toolkit [10] and illustrated in Figure 2. The client generates an RSA public/private key pair and sends a NULL-terminated PKCS#10 [11] certificate request containing the public key to the server. The server signs the certificate request with the private key of the stored credential and sends the signed certificate with corresponding certificate chain back to the client. The format of the server's message is: one byte encoding the number of certificates to be sent[‡], followed by the newly signed proxy certificate, followed by the certificates in the chain, with each certificate encoded in X.509v3 ASN.1 format. The MyProxy repository can contain both X.509 proxy credentials and X.509 end entity credentials, since the delegation protocol works equally well with either type of credential.

The validity period of the new proxy certificate, signed at the MyProxy server and returned to the client, is typically shorter than the validity period of the stored credential, to limit the vulnerability of credentials used outside the MyProxy server. MyProxy allows both credential owners and server administrators to set a per-credential and server-wide maximum lifetime for credentials delegated by the server. A common configuration is for the MyProxy server to hold credentials with lifetimes of weeks, months, or years, with policies that restrict the lifetime of delegated credentials to 12 hours or less. If the client requests a credential with a lifetime greater than the configured maximum, the server will delegate a credential with the maximum allowed lifetime.

To store a proxy credential on the server, the client sends the request (with COMMAND=1), indicating the username and password for the credential, the maximum lifetime for delegated credentials, and other optional attributes such as credential name. If the server gives a successful reply, indicating the client is authorized to store the credential, the proxy delegation protocol proceeds as above, except in reverse. The server generates a public/private key pair and sends a NULL-terminated PKCS#10 certificate request containing the public key to the client. The client signs the request with its private key and sends the new certificate and the entire certificate chain to the server in the same format as described above.

Access control

MyProxy allows both server administrators and credential owners to set access control policies on stored credentials to be enforced by the MyProxy server.

The MyProxy server administrator can control who is allowed to store credentials by requiring client-side TLS authentication for credential storage requests and configuring a regular expression that must match the client certificate's subject (distinguished name) for storage requests to be allowed. The administrator can set a similar regular expression for credential retrieval requests if desired. An example MyProxy server configuration file containing these policy settings is given in Figure 3.

[‡]The Globus Toolkit delegation protocol does not include the one byte encoding the number of certificates. This variation in the MyProxy protocol is simply a historical accident, preserved for backwards compatibility.

```
# Willing to store NCSA credentials:
accepted_credentials "/C=US/O=National Center for Supercomputing Applications/CN=*"

# Allow any client (including clients that do not authenticate via SSL) with a valid
# password to retrieve credentials. This is the recommended setting, as it gives users
# the flexibility to set their own policies on individual credentials.
authorized_retrievers "*"
```

Figure 3. An example MyProxy server policy configuration.

Recent versions of the MyProxy software have added the capability for the MyProxy server administrator to require authentication via a supported SASL [12] mechanism for storing and retrieving credentials, using the Cyrus SASL library [13]. MyProxy can use the SASL PLAIN mechanism, together with configured PAM [14] modules on the server, to interface with local password-based authentication mechanisms at a site, such as LDAP [15], Kerberos [16], or one-time passwords. The SASL GSSAPI mechanism also enables MyProxy to support authentication with Kerberos tickets. Integrating MyProxy with local site authentication mechanisms provides a bridging mechanism between local site security and the grid PKI. For example, a local user with a Kerberos ticket could retrieve a proxy credential from the MyProxy repository without needing to enter another password, if the policies are configured to allow it. Support for one-time passwords is particularly attractive, to mitigate the risk of stolen passwords by keystroke loggers, trojans, or other means.

Credential owners control access to their credentials by setting a password when storing the credentials. The MyProxy server encrypts the private key in the MyProxy repository with the user-chosen password included in the credential storage request message using Triple DES in CBC mode. The MyProxy server does not store the password in the repository, as this would make it easier for an attacker to decrypt keys if the repository were compromised. Thus, to retrieve a proxy credential, the client must send the correct password to the MyProxy server in its request, so the server can decrypt the key to sign the proxy certificate.

The credential owner can also set regular expression policies on the client certificate's subject, similar to the administrator's access control mechanism. Credential owners can optionally require client-side TLS authentication on proxy retrieval requests, requiring the client certificate's subject to match the regular expression. In addition to the client-side TLS authentication, credential owners can require clients to also prove possession of a valid credential with certificate subject matching the stored credential. This last mechanism is used to allow authorized clients holding valid credentials nearing expiration to authenticate to the MyProxy server and "renew" those credentials, by obtaining a new proxy credential with the same certificate subject.

The per-credential password-based and certificate-based access control mechanisms can be used together or separately. In some cases, it is not convenient to require clients to have a password to retrieve a credential, for example, when granting trusted services the ability to renew credentials on a user's behalf. In this case, the credential may be stored without a password and only certificate-based access control may be used. Likewise, we can allow clients without existing credentials (for example, a user at the start of a login session), to obtain credentials with password authentication only, without requiring client-side TLS authentication.

Password quality

To improve the security of the keys in the repository, the MyProxy server administrator can enforce policies on the user-chosen passwords by configuring the MyProxy server to run a plug-in program or script, written by the administrator, before accepting any user-chosen password. For example, the following Perl script provides a plug-in to test password quality using the CrackLib library:

```
use Crypt::Cracklib;
chomp($password = <STDIN>);
$reason = fascist_check($password);
if ($reason ne "ok") {
    print $reason, "\n";
    exit(1);
}
exit(0);
```

This can offer greater control over user-chosen passwords when compared with many scenarios where user keys are stored on desktop systems and administrators have limited control over the software used to protect them.

USE IN PRACTICE

The MyProxy service has a number of applications in grid computing, as described below.

Mobility

Credential mobility is the ability for users to obtain credentials when needed from different locations on the network. Grid users do not always begin their sessions from the same computer. For example, the TeraGrid [17] consists of diverse computational resources across nine partner sites in the United States. TeraGrid users obtain resource allocations that span these various resources and use different resources at different times for different tasks in their scientific studies and in reaction to resource availability. Rather than copy their X.509 credentials onto the different TeraGrid systems they use, the TeraGrid users can store their credentials on a MyProxy server, protected by a password, and use the MyProxy clients to retrieve a credential when needed, as illustrated in Figure 4.

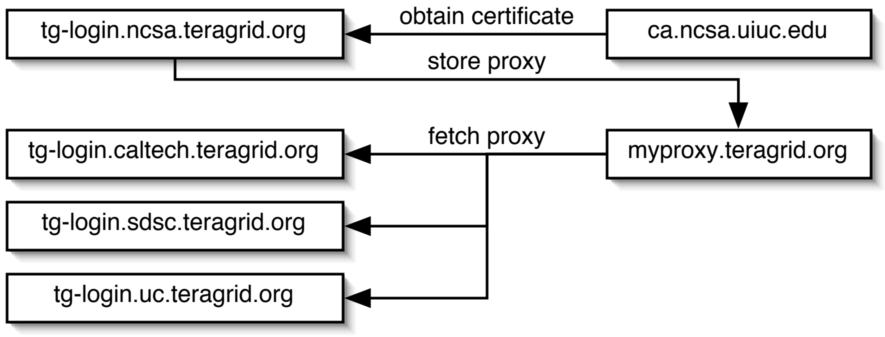

Figure 4. Credential mobility on the TeraGrid.

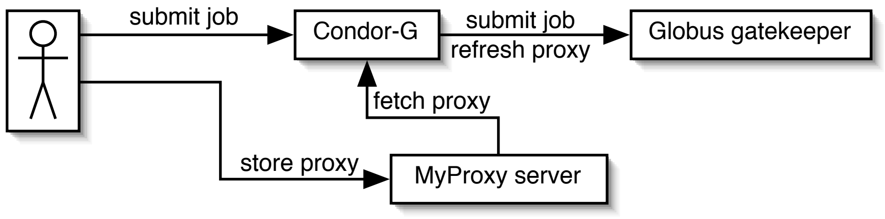

Figure 5. Credential renewal for long-running jobs.

Renewal

Delegating long-lived credentials to long-running jobs on the grid brings an increased risk that a credential will be compromised and misused. Additionally, it is often difficult to predict the run-time of jobs on the grid, due to changes in application performance and resource load, making it difficult to set the lifetime of the delegated credential in advance. The EU DataGrid project [18] developed a solution to this problem using MyProxy, where users delegate short-lived credentials to their jobs, and the workload management system (WMS) monitors jobs to renew their credentials as needed. To renew a job's credential, the WMS connects to the MyProxy server, authenticates both with its service credential and the job's credential (about to expire) to retrieve a new proxy credential for the job. MyProxy can be used in a similar fashion with the widely-used Condor-G [19] job management software. After retrieving a new proxy for the job from MyProxy, Condor-G uses the proxy refresh capability of the Globus Toolkit GRAM job management protocol [20]. This process is illustrated in Figure 5.

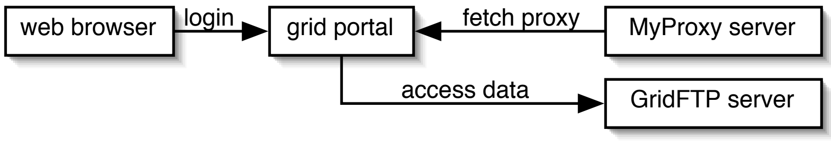

Figure 6. Delegating proxy credentials to a grid portal.

Delegation

The original motivation for the development of MyProxy was to add proxy delegation to existing protocols that do not support it [3], and MyProxy has continued to be used for this purpose in the grid community. One popular example is grid portals, which provide a grid computing interface compatible with standard web browsers. To interact with grid resources on the user's behalf, the grid portal requires access to the user's credentials. However, standard web browsers do not support proxy delegation. Instead, the user stores a credential with a password on the MyProxy server, then connects to the portal via HTTPS [21] and enters the MyProxy username, password, and server address in a web form (submitted over the encrypted HTTPS connection). The grid portal uses this information to contact the MyProxy server and retrieve a proxy credential for the user, which it then uses to submit jobs, transfer files, and perform other grid computing tasks on the user's behalf, as illustrated in Figure 6. Storing longer-lived credentials on the dedicated MyProxy server and giving the grid portal access to only short-lived proxy credentials, on demand, protects the long-lived keys in the MyProxy repository and minimizes the risk of compromise on the grid portal's general-purpose web server to a limited number of short-lived keys. Grid portals that use MyProxy include the NASA Information Power Grid Launch Pad, the PACI HotPage, and the NEESgrid [22] CHEF portal. Additionally, major grid portal toolkits interface with MyProxy, including GridPort [23], GridSphere [24], and the Open Grid Computing Environment (OGCE) portal.

Enrollment

Integrating MyProxy with the CA streamlines PKI enrollment and avoids end-user management of long-lived private keys. When a new user account is created at a site, the accounting system can request a certificate on the user's behalf from the CA and load the user's credentials directly into the MyProxy repository using the initial password for the user's new account. When notified of the initial password for the new account, the user can immediately retrieve a proxy credential from the MyProxy server. Rather than requiring a separate process to obtain PKI credentials, this solution combines PKI enrollment with standard computer system account enrollment at the site. This process is illustrated in Figure 7. The current MyProxy software distribution includes scripts for integrating the MyProxy repository with the Globus Toolkit Simple CA package.

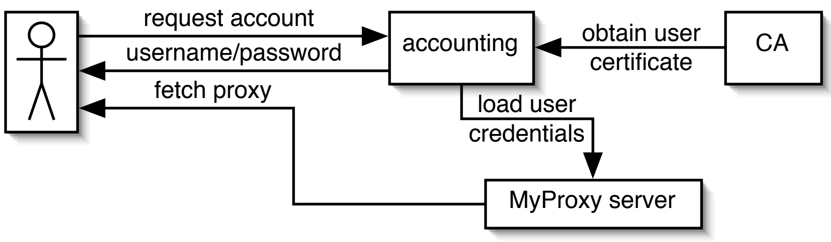

Figure 7. Integrating MyProxy with the PKI enrollment process.

In summary, MyProxy satisfies many requirements for credential management on the grid, by providing mechanisms for credential mobility, renewal, delegation, and enrollment.

OPEN GRID SERVICES ARCHITECTURE

As the grid community has moved recently to adopt web services for the Open Grid Services Architecture (OGSA) [25], the MyProxy project has investigated how credential management services can be implemented in a service oriented architecture.

The Open Grid Services Infrastructure (OGSI) [26] was standardized in the Global Grid Forum in June 2003 as an extension to web services for grid computing. We have developed an OGSI CredentialManager service[§] [27] that allows users to store proxy credentials, encrypted by a pass phrase, for later retrieval over the network. As with MyProxy, credentials are stored and retrieved via a proxy delegation protocol, in which the delegatee generates a new key pair and the delegator signs a proxy certificate containing the delegatee's public key, so that no private keys are transferred over the network.

The OGSI CredentialManager service follows the OGSI factory/instance pattern. To store a credential, the client sends a request to the CredentialManagerFactoryService to create a CredentialManager service instance to which the client then delegates a proxy credential. The client's request to the factory includes the pass phrase and a name for the credential, which are passed as initial parameters to the newly created service instance. The instance generates a key pair, completes the delegation protocol, and immediately encrypts the private key with the user's chosen pass phrase. The delegated credential (private key and certificate chain), along with other parameters of the service instance, is stored as a persistent service property so the service's state can be recovered after failures or restarts. The lifetime of the instance is set equal to the lifetime of the delegated credential, so instances containing expired credentials will themselves expire and be reclaimed by the OGSI hosting environment. Users can also destroy

[§]Available for download from http://myproxy.ncsa.uiuc.edu/ogsa/.

their service instance(s) at any time (analogous to removing a credential from a MyProxy repository).

The CredentialManager service is implemented for the Globus Toolkit OGSI container using the operation provider model [28]. Initial service parameters are passed in the Factory createService method using extensible CreationParameters, and the standard GridService destroy method is used to destroy the service. The CredentialManager provides only one method in addition to the standard GridService methods: the getProxy method for proxy retrieval. Credential information (name and lifetime) is published via OGSI serviceData. Leveraging the capabilities provided by the OGSI hosting environment allowed us to implement the CredentialManager service in about 500 lines of code.

Using the IndexService

When the client creates the service instance, it obtains the Grid Service Handle (GSH) for the new instance. The GSH is a (long) URI that can be used to later locate the service instance to retrieve a proxy credential. It is not convenient for users to remember this GSH, however, so we allow users to choose names for their CredentialManager instances when they are created. Each instance advertises its name, along with other parameters, to the Globus Toolkit IndexService, and CredentialManager clients query the IndexService to obtain the GSH for an instance with a particular name. This results in a user interface very similar to the standard MyProxy software, whereby users can retrieve a proxy credential by name and pass phrase, without needing to know the internal details of GSHs and service instances. CredentialManager service instances re-register with the IndexService periodically in case the IndexService is restarted. If multiple credentials are registered with the same name, the CredentialManager clients display an error message and require the user to choose one of the matching instances by GSH.

Authorization

The CredentialManager service leverages the authorization methods provided by the Globus Toolkit OGSI hosting environment. To control who may store credentials with the CredentialManager, the system administrator can set an access control list (using "gridmap" authorization) for the CredentialManagerFactory service in the hosting environment configuration file. Only authorized users will be able to access the factory to create service instances to hold credentials. This is analogous to the MyProxy "accepted_credentials" policy.

To control who may retrieve credentials, the system administrator can set an access control list on the CredentialManager instances by setting the "instance-authorization" parameter in the hosting environment configuration file. For example, the administrator may allow only trusted grid portals to retrieve credentials, much like how the MyProxy "authorized_retrievers" policy is used today. By default, the CredentialManager allows any client presenting the correct pass phrase to retrieve a credential, including so-called "anonymous" clients that do not have an existing proxy credential. This allows users to obtain their initial proxy from the CredentialManager when needed, without needing to manage long-term certificates and private keys in files on their workstations.

Unlike MyProxy, it is not currently possible to set access policies on individual credentials (i.e., individual CredentialManager service instances) because the capability to dynamically manage service authorization is not provided by the Globus Toolkit OGSI hosting environment. Also, the current MyProxy service allows regular expressions in its access control policies but the Globus Toolkit OGSI gridmap authorization does not.

OGSI credential renewal

The CredentialManager service can also be used to automatically refresh the credentials of long-running jobs submitted to the Globus Toolkit ManagedJobService. Credential refresh is useful in cases when it is difficult to predict the run-time of submitted jobs and the user doesn't want to delegate a long-lived credential to the job service for security reasons. In that case, the user instead creates a new CredentialManager service instance, initialized with the GSH for the ManagedJobFactoryService associated with the user's job, and delegates a long-lived credential to the new CredentialManager service instance. The CredentialManager service periodically queries the service data of the ManagedJobFactoryService to find all ManagedJobService instances, then queries the service data of the instances to find jobs owned by the user with credentials nearing expiration. Implementing this functionality required modifying the ManagedJobService to publish this service data. As the user submits new jobs with short-lived credentials, the CredentialManager service will discover them and renew their credentials as needed.

The underlying mechanism for credential refresh is provided by Globus Toolkit OGSI security. A client renews a service's credential by calling any method of the service using WS-SecureConversation and delegating the new credential as part of the WS-SecureConversation handshake. Delegating a proxy credential to the ManagedJobService makes it available to the running job so it can access secure grid services during its run.

The CredentialManager must be able to renew credentials automatically without user intervention, so it is not feasible to encrypt the proxy credential with a pass phrase provided by the user. The CredentialManager currently does not have a reliable long-term encryption key—it typically runs with a Grid Resource Identity Mapper (GRIM) credential that is periodically re-generated with a new key-pair by the Globus Toolkit OGSI container [29]. Thus, the CredentialManager currently must store the user's proxy unencrypted to support credential refresh. It would be convenient if OGSI hosting environments provided secure storage for a service's persistent data.

Ongoing work

We have begun investigating how our experience with the OGSI CredentialManager service can be applied to the WS-Resource Framework (WSRF) [30] announced in January 2004, and we have done some prototyping of credential managing capabilities using WSRF.NET [31]. We are also developing additional credential management services, including a WS-Trust [32] implementation [33] and a SACRED [34] implementation [35].

RELATED WORK

Much of the design of the MyProxy system is inspired by previous work in cryptographic key management, which has continued to be an active research area. We review related work in this section.

The Kerberos authentication service [16] is a widely-used password-enabled cryptographic system, providing password-based access to cryptographic keys from an online Key Distribution Center (KDC). Each user shares with the KDC a secret key generated from a password. Clients retrieve limited-lifetime credentials encrypted with the secret key from the KDC and decrypt them with the user's password. Clients can also contact the KDC to renew credentials.

The SPX [36] authentication system shares many similarities to MyProxy and the Globus Toolkit security infrastructure. An SPX server holds users' long-term PKI credentials, encrypted with the users' passwords, and users authenticate to the SPX server to retrieve their long-term credentials which they use to sign short-term credentials that are stored unencrypted on the local filesystem to be used for the remainder of the session. Like with proxy credentials, the short lifetime of the SPX session credentials limits their exposure to compromise. SPX differs with MyProxy in the fact that the long-term keys are exported from the SPX server. SPX includes support for delegation by transferring keys, in contrast to the proxy delegation protocol that adds a new proxy certificate to the chain to avoid key transfer.

The Kerberos KDC, SPX server, and MyProxy server are all attractive targets for attack to obtain a large number of user keys. In both the SPX server and the MyProxy server, user keys are encrypted with user-chosen passwords, so a dictionary attack is required upon server compromise to obtain the keys. This vulnerability motivated work to integrate the MyProxy server with the IBM 4758 cryptographic co-processor [37].

Sandhu et al. [1] survey two classes of password-enabled PKI solutions: virtual soft tokens and virtual smart-cards. In their terminology, virtual soft token systems allow users to retrieve private keys from a credential server (for example, a traditional credential repository) and then use the keys directly without further interaction with the server, whereas in virtual smart-card systems users don't have direct access to their private keys but instead must interact with the server to perform signing operations. Virtual smart-card systems split the private key into two components, one computed from the user's password and the other stored on a secure online server. The user and server participate in a signing protocol that does not require either of them to disclose their split key to the other or to reconstruct the complete private key at any point. If the virtual smart-card server is compromised, the attacker can perform a dictionary attack against the server key shares, since the corresponding user shares are computed from user passwords. However, this type of attack is much slower than a dictionary attack against DES encrypted keys. Yaksha, an early example of a virtual smart-card system, replaced shared user secrets in Kerberos with split RSA keys to protect against KDC compromise [38]. It is possible to further protect against server compromise by distributing threshold split keys to multiple servers such that a subset t of n servers work together to generate a threshold signature without reconstructing the complete private key, so a compromise of less than t servers cannot reconstruct the complete private key [39].

A number of commercial PKI credential repositories are available, typically as an add-on to certification authority products to support credential mobility [40, 41]. The nCipher netHSM (www.ncipher.com/nethsm/) provides a network-attached hardware security module for storing private keys. Also, the recently published Securely Available Credentials protocol provides a standard for network-based access to credential repositories [34].

Online certification authorities [42, 43, 44, 45], which create new credentials on demand, can be an alternative to credential repositories. Users authenticate to the online CA and issue a certificate request. The CA sets the user's authenticated identity in a certificate then signs and returns the certificate to the requester. This allows users to retrieve short-term certificates from the online CA on demand without needing to manage long-term private keys. For example, KCA [44] is a popular online CA for grid sites that allows users to authenticate via Kerberos to retrieve short-lived PKI credentials. Like a traditional CA, the security of the online CA's private key is paramount. The CA's private key may be secured by a hardware security module. Threshold split-key approaches can distribute CA functionality across multiple CAs for further protection [45]. One drawback to online CAs is the potentially high cost of adding a new CA to the PKI, which may require renegotiating trust agreements with relying parties. Credential repositories can provide a more flexible solution, since they need not be tied directly to a CA but could be deployed to manage credentials for a single user, a small group within an organization, or a large collaborative group that spans organizations (and CAs).

CONCLUSION

The MyProxy online credential repository provides mechanisms for credential mobility, renewal, delegation, and enrollment for grid computing. The MyProxy software has seen wide use in the grid community over the past four years, and the MyProxy project has continued to support and develop MyProxy to add new capabilities and follow changes in the grid computing infrastructure, including recent work with web services. Virtual smart card systems like MyProxy can provide significant usability and security benefits when compared with more traditional PKI credential management practices.

ACKNOWLEDGEMENTS

We wish to thank the following contributors to the MyProxy software: Shiva Shankar Chetan, Jarek Gawor, Daniel Kouril, Zhenmin Li, Jason Novotny, Feng Qin, Miroslav Ruda, Sumin Song, Benjamin Temko, and Xiao Tu. Thanks also to Mehran Ahsant, Randy Butler, Steve Chan, Doug Engert, Jarek Gawor, Carey Kireyev, Markus Lorch, Miron Livny, Steve Mock, Benjamin Temko, Mary Thomas, Steve Tuecke, Keith Thompson, and Gregor von Laszewski for contributing ideas to the design of the MyProxy system.

REFERENCES

1. Sandhu R, Bellare M, Ganesan R. Password-enabled PKI: virtual smart-cards versus virtual soft tokens. *Proceedings of the 1st Annual PKI Research Workshop*. Gaithersburg, MD, April 2002.
2. Foster I, Kesselman C. Globus: a metacomputing infrastructure toolkit. *International Journal of Supercomputer Application* 1997; **11**(2):115–128.

3. Novotny J, Tuecke S, Welch V. An online credential repository for the grid: MyProxy. *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10).* IEEE Press: San Francisco, August 2001.
4. Tuecke S, Welch V, Engert D, Pearlman L, Thompson M. Internet X.509 Public Key Infrastructure (PKI) proxy certificate profile. *RFC 3820.* IETF, June 2004.
5. Linn J. Generic security service application program interface, version 2. *RFC 2078.* IETF, January 1997.
6. Dierks T, Allen C. The TLS protocol version 1.0. *RFC 2246.* IETF, January 1999.
7. CCITT. Recommendation X.509: the directory–authentication framework. 1988.
8. von Laszewski G, Foster I, Gawor J, Lane P. A java commodity grid kit. *Concurrency and Computation: Practice and Experience* 2001; **13**(8–9):643–662.
9. Butler R, Engert D, Foster I, Kesselman C, Tuecke S, Volmer J, Welch V. A national-scale authentication infrastructure. *IEEE Computer* 2000; **33**(12):60–66.
10. Welch V, Foster I, Kesselman C, Mulmo O, Pearlman L, Tuecke S, Gawor J, Meder S, and Siebenlist F. X.509 proxy certificates for dynamic delegation. *Proceedings of the 3rd Annual PKI R&D Workshop.* Gaithersburg, MD, April 2004.
11. RSA Laboratories. PKCS #10 v1.7: certification request syntax standard. *Public-Key Cryptography Standards (PKCS).* May, 2000.
12. Myers J. Simple Authentication and Security Layer (SASL). *RFC 2222.* IETF, October 1997.
13. Cyrus SASL project. http://asg.web.cmu.edu/sasl/ [14 July 2004].
14. Samar V. Unified login with pluggable authentication modules (PAM). *Proceedings of the 3rd ACM conference on Computer and communications security.* ACM Press: New Delhi, India, 1996.
15. Howard L. An approach for using LDAP as a network information service. *RFC 2307.* IETF, March 1998.
16. Neuman C, Ts'o T. Kerberos: an authentication service for computer networks. *IEEE Communications* September 1994; **32**(9):33–38.
17. Reed D. Grids, the TeraGrid, and Beyond. *IEEE Computer* 2003; **36**(1):62–68.
18. Frohner A, Kouril D, Bonnassieux F, Nicoud S, Hahkala J, Silander M, Cecchini R, Mulmo O, Groep D, Cornwall L, Jensen J, Kelsey D, McNab A. Final security report. European DataGrid Project, January 2004.
19. Frey J, Tannenbaum T, Foster I, Livny M, Tuecke S. Condor-G: a computation management agent for multi-institutional grids. *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10).* IEEE Press: San Francisco, CA, August 2001.
20. Czajkowski K, Foster I, Karonis N, Kesselman C, Martin S, Smith W, Tuecke S. A resource management architecture for metacomputing systems. *Proceedings of the IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing.* Springer-Verlag: Orlando, FL, 1998.
21. Rescorla E. HTTP over TLS. *RFC 2818.* IETF, May 2000.
22. Spencer B, Finholt TA, Foster I, Kesselman C, Beldica C, Futrelle J, Gullapalli S, Hubbard P, Liming L, Marcusiu D, Pearlman L, Severance C, Yang G. NEESgrid: a distributed collaboratory for advanced earthquake engineering experiment and simulation. *Proceedings of the 13th World Conference on Earthquake Engineering.* Vancouver, August 2004.
23. Thomas M, Mock S, Boisseau J, Dahan M, Mueller K, Sutton D. The GridPort toolkit architecture for building grid portals. *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10).* IEEE Press: San Francisco, CA, August 2001.
24. Novotny J, Russell M, Wehrens O. GridSphere: an advanced portal framework. *Proceedings of the 30th EUROMICRO Conference (EUROMICRO'04).* IEEE Press: Rennes, France, 2004.
25. Foster I, Kesselman C, Nick J, Tuecke S. Grid services for distributed system integration. *Computer*, 2002; **35**(6):37–46.
26. Tuecke S, Czajkowski K, Foster I, Frey J, Graham S, Kesselman C, Maguire T, Sandholm T, Snelling D, Vanderbilt P. Open Grid Services Infrastructure (OGSI) version 1.0. *Global Grid Forum Proposed Recommendation*, June 2003.
27. Basney J, Chetan S, Qin F, Song S, Tu X, and Humphrey M. An OGSI CredentialManager service. *Proceedings of the UK Workshop on Grid Security Practice.* Oxford, July 2004.
28. Sandholm T, Gawor J. Globus Toolkit 3 Core: a grid service container framework. http://www-unix.globus.org/toolkit/3.0/ogsa/docs/gt3_core.pdf [2 July 2003].
29. Welch V, Siebenlist F, Foster I, Bresnahan J, Czajkowski K, Gawor J, Kesselman C, Meder S, Pearlman L, and Tuecke S. Security for grid services. *Proceedings of the Twelfth International Symposium on High Performance Distributed Computing (HPDC-12).* IEEE Press: Seattle, WA, June 2003.
30. Czajkowski K, Ferguson D, Foster I, Frey J, Graham S, Sedukhin I, Snelling D, Tuecke S, Vambenepe W. The WS-Resource Framework version 1.0.

http://www.globus.org/wsrf/specs/ws-wsrf.pdf [3 May 2004].

31. Wasson G, Humphrey M. Exploiting WSRF and WSRF.NET for remote job execution in grid environments.
http://www.cs.virginia.edu/ humphrey/papers/WSRFRemoteExecution.pdf [8 October 2004].
32. Kaler C, Nadalin A (ed.). Web services trust language (WS-Trust) version 1.1.
ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf [24 May 2004].
33. Del Vecchio D, Basney J, Nagaratnam N, Humphrey M. CredEx: user-centric credential selection and management for grids.
http://www.cs.virginia.edu/ humphrey/papers/CredEx.pdf [8 October 2004].
34. Farrell S (ed.). Securely available credentials protocol. *RFC 3767*. IETF, June 2004.
35. SourceForge.net SACRED Project. http://sacred.sf.net/ [5 August 2005].
36. Tardo J, Alagappan K. SPX: global authentication using public key certificates. *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE Press: Oakland, CA, May 1991.
37. Lorch M, Basney J, and Kafura D. A hardware-secured credential repository for grid PKIs. *Proceedings of the 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2004)* IEEE Press: Chicago, IL, April 2004.
38. Ganesan R. Yaksha: augmenting Kerberos with public key cryptography. *Proceedings of the Symposium on Network and Distributed System Security*. IEEE Press: San Diego, CA, February 1995.
39. Wang X. Intrusion tolerant password-enabled PKI. *Proceedings of the 2nd Annual PKI Research Workshop*. Gaithersburg, MD, April 2003.
40. Sarbari G. Security characteristics of cryptographic mobility solutions. *Proceedings of the 1st Annual PKI Research Workshop*. Gaithersburg, MD, April 2002.
41. Basney J, Yurcik W, Bonilla R, and Slagell A. The credential wallet: a classification of credential repositories highlighting MyProxy. *Proceedings of the 31st Research Conference on Communication, Information, and Internet Policy (TPRC 2003)*. MIT Press: Arlington, VA, September 2003.
42. Gutmann P. Plug-and-Play PKI: a PKI your mother can use. *Proceedings of the 12th USENIX Security Symposium*. Washington DC, August 2003.
43. Hsu Y, Seymour S. An intranet security framework based on short-lived certificates. *IEEE Internet Computing* April 1998; **2**(2):73–79.
44. Kornievskaia O, Honeyman P, Doster B, Coffman K. Kerberized Credential Translation: a solution to web access control. *Proceedings of the 10th USENIX Security Symposium*. Washington DC, 2001.
45. Zhou L, Schneider FB, van Renesse R. COCA: a secure distributed on-line certification authority. *ACM Transactions on Computer Systems* November 2002; **20**(4):329–368.