Credential Wallets

Jim Basney

July 22, 2001

1 Introduction

This document presents requirements for credential management services for the Grid Security Infrastructure (GSI) [1, 3] and proposes a credential wallet service that makes proxy credentials available securely over the network to authorized clients. It is not exhaustive but is meant as a springboard for further discussion.

Having users manage their own long-term credentials has a number of drawbacks. First, secure storage of private keys can be a challenge. Users must backup their keys to protect against failures and loss of key storage devices. Second, users may access secure services from many devices, and distributing their private keys to each device can be a burden. Third, users may need multiple credentials to access different secure services because of differing trust policies. Managing multiple credentials increases the burden on the end user. For these reasons, providing a credential wallet service to manage user credentials can improve the ease-of-use of grid security mechanisms.

A credential wallet service can provide a standard, trusted mechanism for unattended credential management for long-running tasks. Schedulers, portals, and user agents can interface with the credential wallet service to securely obtain delegated short-term proxies for a user's jobs without requiring that the user's long-term credentials be shared with these various services.

Credential wallets can help meet the GSI single sign-on requirement in two ways. First, the credential wallet service can support authentication via local credentials, so a user need only sign-on to the local security environment; a second "grid" sign-on is not necessary. Second, by making credentials available from any device, users need not remotely login to their "home" computer (where their credentials are stored) or perform a remote copy of their credential files before signing on to the grid. Users can sign on to the grid directly from the current access device.

Ideally, the GSI should be as transparent as possible to novice users. Users at sites with sufficiently strong local authentication mechanisms should be able to access grid resources transparently, with GSI credentials generated transparently "behind the scenes." When users obtain new allocations that require new credentials, those credentials may be silently added to their credential wallet according to agreements between administrators.

2 Related Work

The IETF Securely Available Credentials (SACRED) working group[1] is developing protocols for secure credential portability. Current SACRED drafts describe a credential server that securely stores credentials associated with a username, password, and tag in a credential repository. The credentials can then be retrieved at a later time by authenticating to the server with the username and password and indicating the

[1] http://www.ietf.org/html.charters/sacred-charter.html

requested credentials using the tag. The credentials are considered opaque data by the credential server. The server doesn't perform any form of delegation.

The K5Cert service provides proxy credentials to Kerberos [6] authenticated clients. K5Cert acts as an online certificate authority (CA), generating proxy credentials on-the-fly for the Kerberos principle.

The MyProxy service [7] stores delegated X.509v3 [5] credentials that can be retrieved via a user generated passphrase. Users can store their credentials on the MyProxy server and access them securely from a web browser, enabling grid resource access via web portals.

3 Types of Credential Wallets

The GSI Roadmap [8] outlines three types of credential wallets.

Online certificate authority (CA): In this implementation, the credential wallet service acts as a certificate authority, generating proxy credentials for the user on the fly. There are no long-term credentials for the user to manage in this case, so there are no private keys stored on potentially insecure end-user devices such as laptops and PDAs. However, the online CA represents a single point of vulnerability for all of the users it serves. A compromise of the online CA would allow an attacker to create proxy credentials for any user. Another drawback to this approach is that it represents an additional CA to be trusted in the system. K5Cert provides this functionality for Kerberos authenticated users.

Long-term credential repository: In this implementation, the user either uploads his or her long-term credentials to the credential wallet or the credential wallet service obtains long-term credentials for the user from the CA. The credential wallet securely stores the user's credentials and generates proxy credentials for the user via delegation when requested. One advantage of this approach is that it doesn't introduce an additional CA. The credentials would be stored encrypted with the user's passphrase in the repository, so a compromise of the repository would not immediately compromise all user keys; an additional offline attack on the keys would be required.

Proxy repository: In this implementation, the user uploads (delegates) proxy credentials to the credential wallet for later retrieval. This option allows expert users to retain and protect their own private keys without sharing them with the credential wallet service. A compromise of the proxy repository yields only short-lived proxy credentials; long-term credentials would not need to be revoked and re-issued. The service can automatically delegate proxies with shorter lifetimes than the uploaded proxy's lifetime for enhanced security. A drawback to this approach is the user must periodically manually refresh the proxy stored in the wallet. MyProxy implements this type of service.

All three types of credential wallets may be useful in a single domain, so an integrated credential wallet service that provides all three components may provide the greatest utility.

4 Client Authentication

Clients must authenticate to the credential wallet before uploading or downloading credentials. In the case where the credential wallet generates the proxy credential on the user's behalf, authentication must serve to prove the identity to be claimed in the proxy certificate. In the case where credentials are stored as opaque, encrypted data in the credential wallet, authentication ensures that an attacker must successfully perform an online attack on the server before retrieving encrypted credentials to attack offline.

If possible, it is preferable for the client to authenticate with an existing set of credentials, such as a Kerberos ticket, so the user need not perform an additional sign-on operation. Otherwise, the client can

authenticate to the server with a password over a encrypted TLS [2] connection or with a strong password-based protocol, such as the Secure Remote Password (SRP) protocol [9].

5 Central Versus Distributed Storage of Private Keys

The trade-off of a centralized credential wallet service, compared to distributed user management of private keys, is that while the central service can be professionally administered with a high level of security, it provides a single point of compromise for a large number of credentials. To make a fair comparison, we should note that in many environments, user managed keys are (or would be) stored locally in users' home directories on a network file server. A compromise of that network file server (or a well-placed network sniffer reading unencrypted I/O traffic) could also collect a large number of keys in such an environment. In any case, the credential wallet service must provide a high level of security to address the potential vulnerability of a large number of private keys stored in a central location.

6 Multiple Credentials

Some mechanism is required to differentiate between multiple credentials stored in a user's wallet.

The client can associate each uploaded credential with a tag and use that tag to retrieve the needed credential at a later time. The utility of client-generated tags is limited, however, because clients must retain information about the tags or get information from the user to know which tag is associated with the currently needed credential.

If the credential wallet can parse uploaded credentials (i.e., the credentials are not opaque), then clients can query and retrieve credentials based on standard credential fields such as certificate type, issuer, and distinguished name.

7 Operational Requirements

The following general operational requirements for the credential wallet must be considered.

Security: The security of the credential wallet is obviously a primary concern. Access to credentials must be restricted, and credentials must be backed up to protect against hardware failure. The credential wallet service should meet or exceed the security provided by current accepted practice, i.e., user control of private keys.

Performance, scalability, and availability: The wallet must support fast credential retrieval, scale to large numbers of stored credentials, and maintain a high level of availability. In contrast to a locally stored private key, a remote credential service is vulnerable to network and server outages. If the wallet is unavailable, users will either resort to other, potentially less secure, credential access mechanisms or be denied access to grid services. Therefore, replication support may be an important requirement for the successful adoption of a credential wallet service.

Query capability: The repository must support efficient query operations to support locating the appropriate credential for access to a given service. Certificate-specific query semantics, such as searching certificate chains for qualifying certificates, may be required.

8 Additional Issues

Limited delegation of user rights: Support for limited delegation in the credential wallet is not strictly required, since the client (for example, a job scheduling agent) could add restrictions with an additional level of delegation after obtaining the proxy from the wallet. However, it may be useful to implement policies for restricted delegation in the credential wallet to limit the rights transferred to some clients. For example, the wallet may deliver proxies with different restrictions based on the security of the client platform, giving only limited rights to clients on mobile computers versus full rights to clients on professionally administered servers.

Notification and logging of grid events: Security events, such as the generation and distribution of proxy credentials by a credential wallet, should be securely logged, enabling online monitoring by a trusted third-party and offline auditing to detect and diagnose problems. Users should be able to obtain information about security events previously carried out on their behalf and request notification of and control over future events. For example, a user may request that all uses of his or her credentials be authorized at run-time by a dialog box on the user's desktop. Or, the user may request an email notification for each security event involving the user's credentials or a daily email summary of such events.

GSI audit and certification: GSI specifications and implementations should be audited and certified by a third-party for greater assurance of good software engineering and security practices.

Download of long-term credentials: The SACRED working group is focusing on secure access to private keys rather than access to proxy credentials. Downloading private keys instead of proxies can reduce load on the credential server because proxies can be generated locally on demand once the private keys are downloaded.

Decentralized credential wallets: An alternative to the centralized credential wallet service proposed in this document is a service that enables secure exchange of credentials between end-systems. The SACRED working group is considering this type of direct transfer service in addition to the credential server approach.

9 Conclusion

The SACRED, K5Cert, and MyProxy efforts each address some aspect of the credential wallet idea. We should investigate where standardization can improve interoperability between these efforts, with the understanding that integration can improve ease-of-use for end-users. We also need to understand the different places where credential wallets can fit into the grid infrastructure. For example, credential wallets can potentially provide credential management services to application schedulers like CondorG [4].

References

[1] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch. A national-scale authentication infrastructure. *IEEE Computer*, 33(12):60–66, December 2000.

[2] T. Dierks and C. Allen. The TLS protocol version 1.0. IETF RFC 2246 (Standards Track), January 1999.

[3] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 83–92, 1998.

[4] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke. Condor-G: A computation management agent for multi-institutional grids. In *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)*, August 2001.

[5] R. Housley, W. Ford, W. Polk, and D. Solo. Internet x.509 public key infrastructure certificate and crl profile. IETF RFC 2459 (Standards Track), January 1999.

[6] B. C. Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32(9):33–38, September 1994.

[7] J. Novotny, S. Tuecke, and V. Welch. An online credential repository for the grid: Myproxy. In *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)*, August 2001.

[8] S. Tuecke. Grid security infrastructure (GSI) roadmap. Grid Forum Security Working Group Draft, February 2001.

[9] T. Wu. The SRP authentication and key exchange system. IETF RFC 2945 (Standards Track), September 2000.