The Credential Pattern*

Patrick Morrison and Eduardo B. Fernandez

Dept. of CS & Eng., Florida Atlantic University,
777 Glades Road, Boca Raton, FL 33341-0991
morrison@fau.edu, ed@cse.fau.edu

Credential provides secure portable means of recording authentication and authorization information for use in distributed systems.

Example

Suppose we are building an instant messaging service to be used by members of a university community. Students, teachers and staff of the university may communicate with each other, while outside parties are excluded, perhaps for reasons of privacy. Members of the community may use computers on school grounds, or their own systems, so the client software is made available to the community and is installed on the computers of their choice. Any community member may use any computer with the client software installed. The client software communicates with servers run by the university in order to locate active participants and to exchange messages with them.

In this environment, it is important to establish that the user of the client software is a member of the community, so that communications are kept private to the community. Further, when a student graduates, or an employee leaves the university, it must be possible to revoke their communications rights. Each member needs to be uniquely and correctly identified, and a member's identity should not be forgeable.

Context

Distributed systems where users may need to access remote sites to perform some functions.

Problem

In individual computer systems, the authentication and authorization of a principal can be handled by that system's operating system, middleware and/or application software; all facts of the principal's identity and authorization are created by and are available to the system. With distributed systems, this is no longer the case. A principal's identity, authentication and authorization on one system does not carry over to another system. If a principal is to gain appropriate access to another system, some means of conveying this information must be introduced.

More broadly, this is a problem of exchanging data between trust boundaries. Within a given trust boundary, a single authority is in control, and can authenticate and make access decisions on its own. If the system is to accept requests from outside its own authority/trust boundary, the system has no inherent way of validating the identity or authorization of the entity making that request. At the heart of the external request is the data necessary to make these decisions.

The solution to this problem must resolve the following forces:

- Persistence: Data must be packaged and stored in a way that survives travel between systems.
- Portability: Data must be stored in a way that allows it to be used in contexts other than the one it was created in.
- Protection: The original authentication and authorization information must be preserved intact, including defenses against tampering and forgery.
- Authentication: The data available must be sufficient for identifying the principal to the satisfaction of the accepting system's requirements.
- Authorization: The data available must be sufficient for determining what actions the presenting principal is permitted to take within the accepting system.
- Trust: The system accepting the credential must trust the system issuing the credential.

Solution

Pack authentication and authorization data in a data structure separate from the systems in which the data are created and used. When presented to a system, the data (**Credential**) can be used to grant access and authorization rights to the requestor. In order for this to be a meaningful security arrangement, there must be an agreement between the systems which create the credential (**Credential Authority**) and the systems which allow their use, dictating the terms and limitations of system access.

Structure

In Figure 1, the **Principal** is an active entity such as a person or a process. The **Principal** possesses a **Credential**, representing its identity and its authorization rights. A **Credential** is a composite describing facts about the rights available to the principal. The **Attribute** may flag whether it is presently enabled, allowing principal control over whether to exercise the right implied by the **Credential**. Expiration date allows control over the duration of the rights implied by the attribute.

A **Credential** is issued by an **Authority**, and is checked by an **Authenticator** or an **Authorizer**. Specialization of a **Credential** is achieved through setting A**ttribute** names and values.

Some specific specializations of **Attributes** are worth mentioning. Identity, created by setting an attribute name to, say, 'username' and the value to the appropriate username instance, shows that the subject has been authenticated and identified as a user known to the **Authenticator**. Privilege, named after the intended privilege, implies some specific ability granted to the subject. Group and Role can be indicated in a similar fashion to Identity.

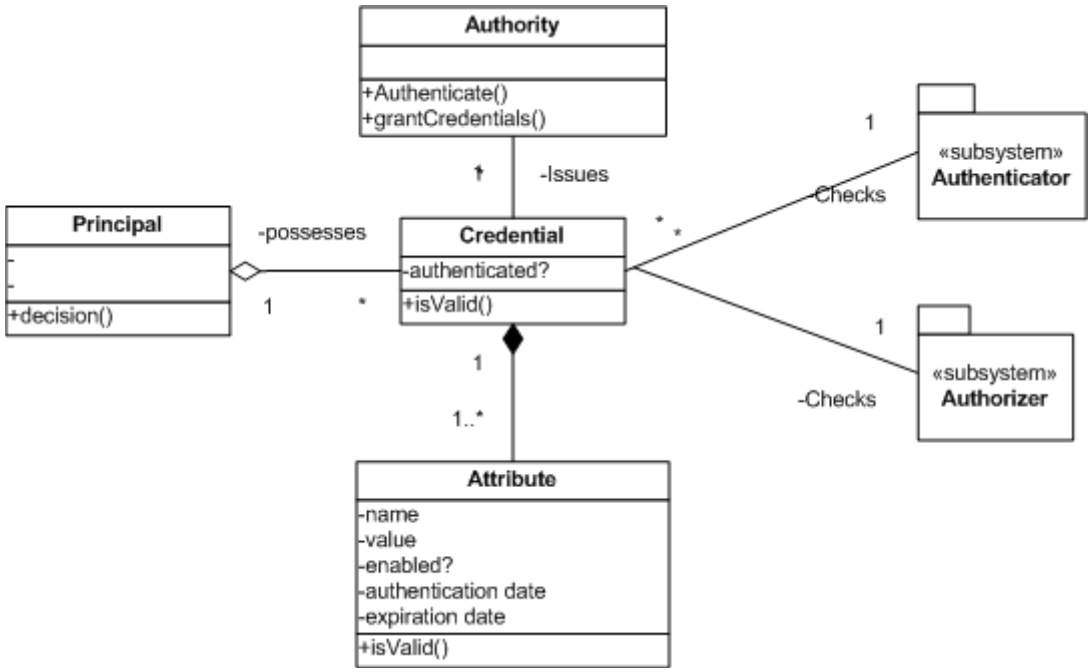

Figure 1: Credential Class Diagram

Dynamics

There are three primary use cases; Issue Credential, by which a Credential is are granted to the Principal by an Authority, Principal Authentication, where an Authenticator accepts a Credential provided to it by a Principal, and makes an access decision based on the Credential, and Principal Authorization, where the Principal is allowed access to specific items. This paper describes the first two use cases.

Issue Credential

The Principal presents itself and any required documentation of its identity to an Authority (Figure 2). Based upon its rules and what it ascertains about the Principal, the Authority creates and returns a credential. The returned data may include an identity credential, group and role membership credential attribute, and privilege credential attributes. As a special case, the Authority may generate a defined 'public' credential for Principals not previously known to the system. This credential is made available to Authenticators which reference this Authority.

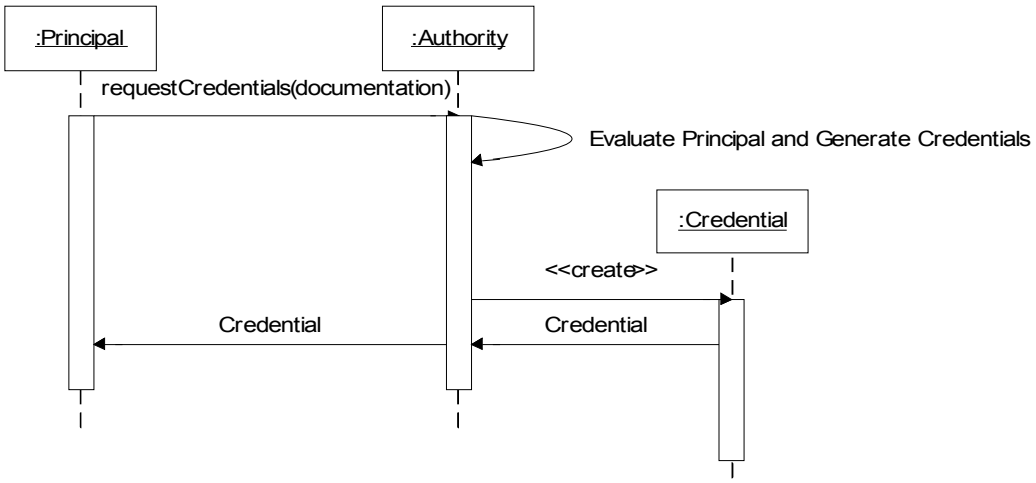

Figure 2: Issue Credential sequence diagram

Principal Authentication

The Principal requests authentication at an Authenticator, suppling its name and authentication Credential (Figure 3). The Authenticator checks the Credential and makes an access decision. There are different phases and strengths of check that may be appropriate for this step, discussed in the Implementation section.

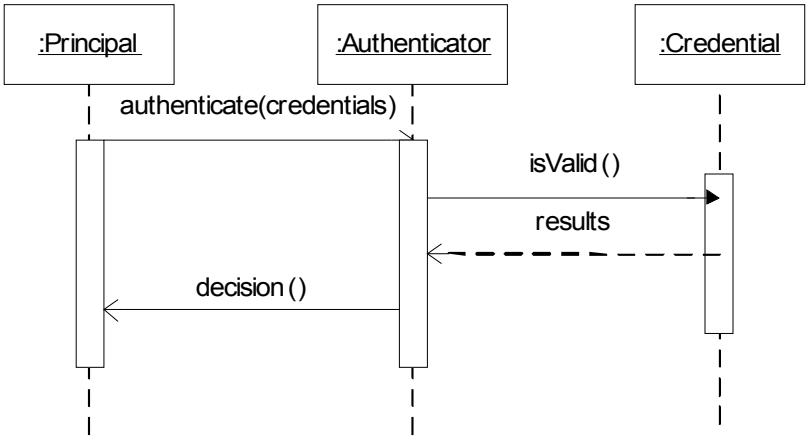

Figure 3: Subject Authentication sequence diagram

Implementation

The most significant factor in implementing Credential is to determine the nature of the agreement between the participating systems. This begins with consideration of the functions to be provided by the system to which credentials will give access, the potential users of those functions, and the set of rights which are required in order for each user to fulfill its role. Once these are understood, a clear representation of the subjects, objects and rights can be developed. This representation forms the basis for storing credentials in some persistent medium and sets the terms of authentication and authorization. It also forms the basis for portability, as persisted data may be placed on portable media for transmission to the location(s) of its use.

The problem with a clear representation of security rights is that bad actors can read them as well as valid participants in the systems in question. In the physical world, anti-forgery devices for credentials take the form of embedding the credential data in media that is too expensive to be worth forging for the benefit received; driver's license and other id cards, passports, and currency all are based on the idea that it is too expensive for the majority of users to create realistic fakes. In the digital world, copies are cheap. There are two common means of addressing this. One is to require that credentials be established and used within a closed context, and encrypting the communications channels used in that context. The other is to encrypt the credentials when they are issued, and to set up matching decryption on the authenticating system. This further subdivides into "shared secret" systems, where the issuing and accepting systems share the cryptographic keys necessary to encrypt and decrypt credentials, and "public key" systems, where participating systems can establish means for mutual encryption/decryption without prior sharing. These design choices are part of the terms set by the Authority agreement under which the credentials apply. The Authenticator must use the same scheme as the Authority. Kerberos tokens and X.509 certificates are examples of this that require more specific approaches, see [Lop04].

In implementing the Principal Authentication use case, there are different phases and strengths of check that may be appropriate. For example, when entering my local warehouse club, I need only flash a card that looks like a membership card to the authenticator standing at the door. When it comes time to make a purchase, however, the membership card is checked for validity, expiration date and for whether it belongs to the person presenting it. In general, the authenticator is responsible for checking the authenticity of the credentials themselves (anti-forgery), whether they belong to their bearer, and whether they constitute valid access to the requested object(s). There is a good discussion of levels of inspection on page 246 of [And01b].

Consequences

This pattern has the following advantages:

- Fine-grained authentication and authorization information can be recorded in a uniform, persistent and portable way.
- A Credential from a trusted authority can be considered proof of identity and of authorization.
- It is possible to protect credentials using encryption or other means.

This pattern has the following disadvantages:

- It might be difficult to find an authority that can be trusted. This can be resolved with chains (trees) of credentials, where an authority certifies another authority.
- Making credentials tamper-resistant takes extra time and complexity.
- Storing credentials outside of their using systems leaves system authentication and authorization mechanisms open to offline attack.

Example Resolved

Create a credential authority, “IM Registration”. Give it the responsibility of verifying identity and granting a username and password, in the form of an id card, to university community members when they join the university community. This login embodies the authority of the granting agency, and embodies the identity of the subject as verified by the agency. Set policy and user guide policies so that members are encouraged to keep their login information private.

Code the client software to implement an Authenticator when someone wishes to start a session. Grant or deny access based on the results of the authentication. Implement checks on the servers to ensure that the member's credential is not expired.

Known Uses

This pattern is a generalization of the concepts embodied in X.509 Certificates, CORBA Security Service's Credentials [And01], Windows security tokens [Bro05], SAML assertions [Hug05], and the Credential Tokenizer pattern [Ste05]. Capabilities, as used in operating systems, are another implementation of the idea.

Passports are a non-technical example of the problem and its solution. Countries must be able to distinguish between their citizens, citizens of nations friendly and unfriendly to them, trading partners, guests, and unwanted persons. There may be different rules for how long visitors may stay, and for what they may engage in while they are in the country. Computer systems share some of these traits; they must be able to distinguish between members of their user community, and non-members. These non-members may be eligible or ineligible to gain system access or participate in transactions.

Related Patterns

Metadata-based Access Control [Pri04] describes a model where credentials can be used to represent subjects. The *Credential* pattern complements *Security Session* [Sch06] by giving an explicit definition of that pattern's 'Session Object', as extracted from several existing platforms. The *Authenticator* pattern [Bro99] and the Remote Authenticator/Authorizer [Reg02] describe types of authenticator. An Authorizer is a concrete version of the abstract concept of Reference Monitor [Sch06]. Delegation of credentials is discussed in [Wei06]. [Ste05] describe a Session Object pattern that "abstracts encapsulation of authentication and authorization credentials that can be passed across boundaries".

That is an incorrect interpretation of the concept of credentials. Credentials abstract authentication and authorization rights. They confuse credentials with rights.

Acknowledgements

We thank our shepherd Jorge Ortega Arjona and Ralph Johnson for valuable comments that improved this paper. The FAU Secure Systems Research Group also contributed valuable ideas. This work was supported through a Federal Earmark grant from the Defense Information Systems Agency (DISA), administered by Pragmatics, Inc.

References

[And01] R. Anderson, *"*CORBA Security Service Specification*"*, OMG 2001. http://www.omg.org/docs/formal/02-03-11.pdf

[And01b] R. Anderson, *Security Engineering*, Wiley 2001.

[Bro99] F.L. Brown, J. DeVietri, E.B. Fernandez, "The Authenticator Pattern", *Proceedings of Pattern Language of Programs (PloP'99)*, August 15-18, 1999.

[Bro05] K. Brown, *The .NET Developer's Guide to Windows Security*, Addison-Wesley, 2005.

[Del06] N. A.Delessy, E.B.Fernandez, and M.M.Larrondo-Petrie, "Patterns for identity management standards" , submitted for publication.

[Hug05] J. Hughes, E. Maler, "Security Assertion Markup Language (SAML) 2.0 Technical Overview", http://xml.coverpages.org/SAML-TechOverview20v03-11511.pdf

[Lop04] J. Lopez, R. Oppliger, and G. Pernul, "Authentication and authorization infrastructures (AAIs): a comparative survey", *Computers & Security,* vol. 23, 2004, 578-590.

[Pri04] T. Priebe, E.B.Fernandez, J.I.Mehlau, and G. Pernul, "A pattern system for access control ", in *Research Directions in Data and Applications Security XVIII,* C. Farkas and P. Samarati (Eds.), *Procs of the 18th. Annual IFIP WG 11.3 Working Conference on Data and Applications Security,* Sitges, Spain, July 25-28, 2004.

[Reg03] R. Warrier, E.B. Fernandez, "Remote Authenticator/Authorizer", *Pattern Languages of Programs Conference (PLoP), 2003*

[Sch06] M. Schumacher, E. B. Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating Security and Systems Engineering*, Wiley 2006.

[Ste05] C. Steel, R. Nagappan, and R. Lai, *Core Security Patterns: Best Strategies for J2EE, Web Services, and Identity Management,* Prentice Hall, Upper Saddle River, New Jersey, 2005.

[Wei06] M. Weiss, "Credential delegation: Towards grid security patterns", accepted for the *Nordic Pattern Languages of Programs Conference (VikingPLoP), 2006*