An abstract security pattern for Authentication and a derived concrete pattern, the Credential-based Authentication

EDUARDO B. FERNANDEZ, Florida Atlantic University
NOBUKAZU YOSHIOKA, National Institute of Informatics
HIRONORI WASHIZAKI, Waseda University
JOSEPH YODER, The Refactory, Inc.

An Abstract Security Pattern (ASP) is a security pattern that describes a conceptual semantic restriction in a domain which can be a defense to a threat or a way to comply with a regulation, with no implementation aspects. A concrete pattern describes a security solution within a technological context. ASPs can be used as a guideline for the development of concrete patterns. We show here an ASP (Authenticator) from which we derive a concrete pattern (Credential-based Authenticator). The Authenticator ASP is a revisit of our previous version of Authenticator, the Credential-based Authenticator is a new pattern.

Categories and Subject Descriptors: D.2.11 **[Software Engineering]** Software Architectures–Patterns; D.5.1 D.4.6 **[Security and Protection]** Authentication

General Terms: Design

Additional Key Words and Phrases: Security patterns, authentication, secure software, software architecture, conceptual models

ACM Reference Format:

E.B.Fernandez, N. Yoshioka, and H. Washizaki, 2018. Procs. 7th Asian Conference on Pattern Languages of Programs, Asian PLoP'18, March 1-2, Tokyo, Japan. 7 pages.

1. INTRODUCTION: ABSTRACT AND CONCRETE SECURITY PATTERNS

Security restrictions should be defined at the semantic level of applications and reflected to the lower levels (Fernandez 2013). An *Abstract Security Pattern* (ASP) is a security pattern that describes a conceptual semantic restriction in a domain which can be a defense to a threat or a way to comply with a regulation, with no implementation aspects (Fernandez et al., 2014). An ASP describes the essential functions that must be present to handle a threat or regulation in an implementation-independent way. For example, "only the owners of accounts in a financial institution are permitted to withdraw money from them" is a typical semantic restriction. A *concrete pattern* describes a security solution within a technological context, e.g., Authentication in distributed systems, Authorization for XML web services, etc.

ASPs have value in several ways; one of the most important is to serve as a reference and guideline for the development of concrete patterns. We show here an ASP (Authenticator) from which we derive a concrete pattern (Credential-based Authenticator). The Authenticator ASP is a revisit of our previous version of Authenticator (Fernandez 2013), the Credential-based Authenticator is a new pattern. In this way, it is possible to build a Secure Solution Frame (SSF), a collection of related patterns tailored for different architectural levels (Uzunov et al. 2015), by adding for example, X.405-based Certificate Authentication and SAML-based Authentication patterns.

Authentication restricts access to a system to only registered users; it handles the threat where an intruder enters a system and may try to perform unauthorized access to information by taking up somebody else identity. It is clear that there are many ways to perform authentication, that go from manual ways, as done in voting places, to purely automatic ways, as when accessing a web site, they all must conform to the core aspects of the Authentication ASP. Our audience includes designers of security systems and developers of applications where security is a concern.

Authors' addresses: Eduardo B. Fernandez (corresponding author), Dept. of Computer and Electrical Eng. and Computer Science, Florida Atlantic University, 777 Glades Rd., Boca Raton, FL33431, USA; email: ed@cse.fau.edu; Nobukazu Yoshioka, GRACE Center, National Institute of Informatics, Tokyo, Japan, email: nobukazu@nii.ac.jp; Horonori Washizaki, Waseda University, Tokyo, Japan, email: washizaki@waseda.jp; Joseph Yoder, The Refactory, Inc, Urbana, IL 61801, USA, joe@joeyoder.com

2. AUTHENTICATOR ASP

2.1 Intent
When an active entity like a user or system (subject) identifies itself to the system, how do we verify that the subject intending to access the system is who it says it is? The subject presents some information that is recognized by the system as identifying this subject. After being recognized, the requestor is given some proof that it has been authenticated.

2.2 Context
Computer systems contain resources that may be valuable because they include information about business plans, user medical records, and similar. We only want subjects that have some reason to be in our system to get into the system. Malicious subjects may misuse our information. This is also true for physical places, where there may be a fear of physical damage.

2.3 Problem
How to prevent impostors from accessing our system? A malicious attacker could try to impersonate a legitimate user to have access to her resources. This could be particularly serious if the impersonated user has a high level of privilege. How do we verify that the user intending to access the system is legitimate? In a physical environment an intruder could destroy valuable equipment or even kill people.

2.4 Forces
The following forces apply to the possible solution of the Abstract Authenticator:

- *Closed or open system*. If the authentication information presented by the user is not recognized, there is no access. In an open system all subjects would have access except some who are blacklisted for some reason. The rest of the forces refer to closed systems, which are by nature more secure than open systems.
- *Registration*. Users must first register their identity information so that the system can recognize them later.
- *Flexibility.* There may be a variety of individuals (users) who require access to the system and a variety of system units with different access restrictions. We need to be able to handle all this variety appropriately or we risk security exposures.
- *Availability*. We need to authenticate users in a reliable way. This means a robust protocol and a high degree of availability. Otherwise, users may fool the authentication process or enter when the system authentication is down.
- *Protection of authentication information*. Users should not be able to read or modify the authentication information. Otherwise, they can forge ways to give themselves access to the system.
- *Simplicity*. The authentication process must be relatively simple or the users or administrators may be confused. User errors are annoying to them but administrator errors may lead to security exposures.
- *Reach*. Successful authentication only gives access to the system, not to any specific resource in the system. Access to these resources must be controlled using other mechanisms, typically authorization.
- *Tamper resistance*. It should be very difficult to falsify the proof of identity presented by the user.
- *Cost*. There should be tradeoffs between security and cost, more security can be obtained at a higher cost.
- *Performance*. Authentication should not take a long time or users will be annoyed.
- *Frequency*. We should not make users authenticate frequently. Frequent authentications waste time and annoy the users.

2.5 Solution
Use a single point of access for the interactions of a subject with the system and apply a protocol to verify the identity of the subject. The protocol used may be simple or complex depending on the needs of the application. More specifically, every authentication protocol must include two steps:

--The subject requests to enter a system indicating its identity and presenting some proof of identity (*Identification*).

--If the system recognizes the subject using its identity information, it grants it entrance to the system and provides it with a proof of authentication for further use. If not, the request is denied (*Authentication*).

Concrete realizations implement these steps in different ways but all must perform these two steps.

2.5.1 *Structure*

Figure 1 shows the class model of the Abstract Authenticator, obtained from the realization of the activities above (as it would be done as part of any object-oriented design approach). In this model, class *Subject* indicates the active entity (User or system) that asks for access to the system by issuing a request to class *Authenticator* followed by a *Proof of Identity* (which is owned by the subject). The Authenticator then consults class *Authentication Information* to decide if the subject is legitimate. Class Authentication Information includes whatever information is needed to authenticate users, e.g. a list of passwords, a set of fingerprints, a history of past interactions, or similar. The Authenticator provides the requestor with a *Proof of Authentication* so that the requestor needs not authenticate itself again within some period of time.

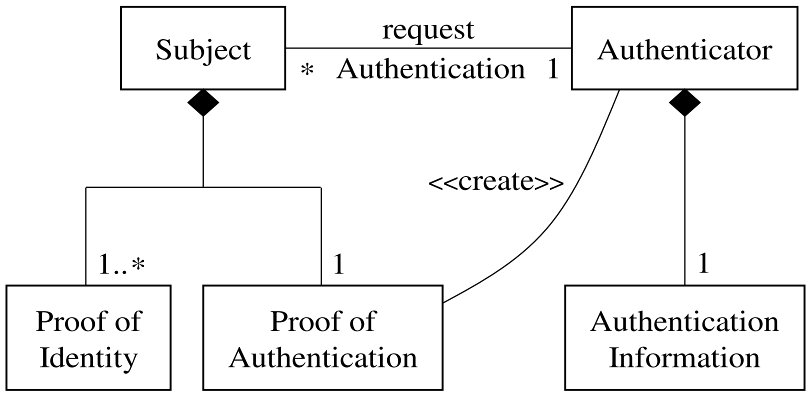

Fig. 1. Class diagram of the Abstract Authenticator pattern

2.5.2 *Dynamics*

Figure 2 shows a sequence diagram for the use case ‘Authenticate a subject’. A subject (actor) requests authentication from the authenticator providing its identity and a proof of identity. The authenticator verifies if this information is true using its authentication information. If true, the authenticator will create a proof of authentication which will be given to the subject.

Other use cases would include Register Subject, Remove Subject, and Update Subject Information.

2.5.3 Threats

For the Abstract Authenticator we can have:

- T1. Present fake or stolen proof of identity, to let the attacker impersonate a legitimate subject.
- T2. Steal the proof of authentication to grant access to the attacker.
- T3. Unauthorized reading of authentication (identity) information, to let the attacker forge a proof of authentication.
- T4. Unauthorized modification of identity information, to deny access to legitimate users.
- T5. Register a subject using false information.

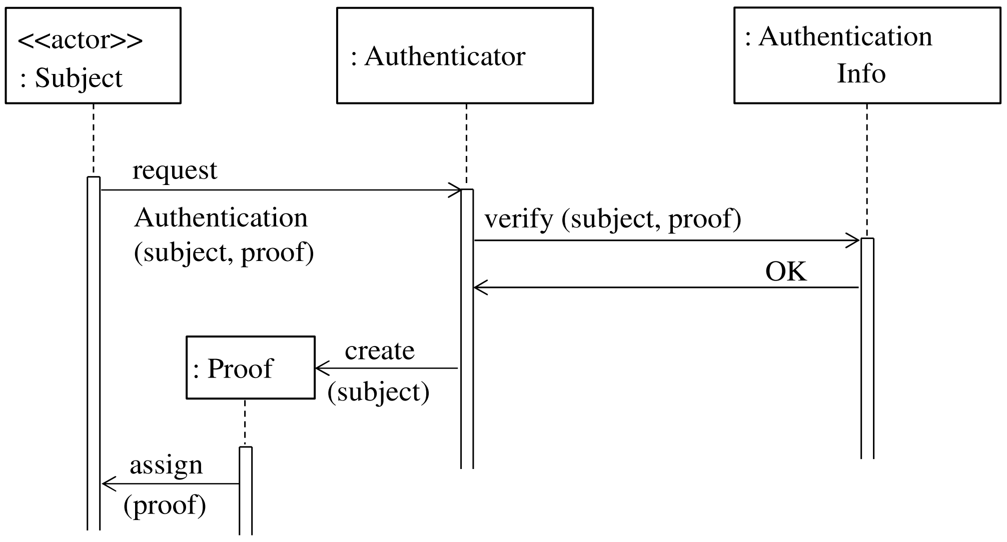

Fig 2. Sequence diagram for the use case Authenticate a subject. .

2.6 Known uses

- Authentication of voters during political elections. The proof of identity is some document, e.g. a driver's license or a passport. The proof of authentication is a paper vote or a code to vote electronically.

- Authentication of arriving passengers to an international airport. The proof of identity is a passport and the proof of authentication is a stamp in the passport.

- Authentication of users when logging in some operating system. The proof of identity is usually a password and the proof of authentication is a session.

2.7 Consequences

The Authenticator provides the following advantages:

- *Closed or open system*. In the access policy of the Authenticator we can implement whitelisting or blacklisting policies. The rest of the consequences refer to closed systems, using whitelisting.
- *Registration*. If a subject is not registered there will be no authentication information for it and it will be denied access.
- *Flexibility.* The authentication information can be defined so as to accommodate any type of subjects.
- *Availability*. We can add redundancy to the Authenticator to increase its availability (Buckley et al. 2011).
- *Protection of authentication information*. We can use standard security mechanisms such as authorization to protect the authentication information. Threats T3 and T4 in 2.5.3 can be handled in this way.
- *Simplicity*. We can adjust the protocol to be as simple as necessary for the type of users and the degree of security required.
- *Reach*. Successful authentication only gives access to the system, not to any specific resource in the system. We can add further controls once a subject is admitted.
- *Tamper resistance*. We can define constraints on the type of authentication proof to make it hard to imitate.
- *Cost*. By adjusting the type of authentication proof we can trade off cost for degree of security.
- *Performance*. By adjusting the type of authentication proof we can also adjust the time it takes for authentication. In this way we can trade off performance for degree of security.

- *Frequency*. The proof of authentication avoids the need to re-authenticate subjects.

Liabilities include the extra complexity and time required for authentication; which depend on the type of concrete authentication that will be used. In general, more secure methods require more time, are more complex, and are more costly.

2.8 Related patterns

- This pattern is based on the one presented in (Fernandez 2013), improved with abstraction concepts.

- Identity patterns (Delessy et al., 2007, Fernandez 2013). They include Circle of Trust, Identity Federation, and Identity Provider. They are necessary to uniquely identify each legitimate user or system. The Circle of Trust pattern allows the formation of trust relationships among service providers in order for their subjects to access an integrated environment. The Identity Provider pattern allows the centralization of the administration of subjects' identity information for a security domain. The Identity Federation pattern allows the formation of a dynamically created identity within an identity federation consisting of several service providers. Therefore, identity and security information about a subject can be transmitted in a transparent way for the user among service providers from different security domains.

- This pattern is normally complemented with some variety of Authorizer pattern that controls access to resources.

3. THE CREDENTIAL-BASED AUTHENTICATOR

3.1 Intent
Perform authentication using a certified credential as proof of identity.

3.2 Context
Computer systems contain resources that may be valuable because they include information about business plans, user medical records, and similar. We only want subjects that have some reason to be in our system to get into the system. Malicious subjects may misuse our information. This is also true for physical places, where there may be a fear of physical damage. In distributed systems users may need to access different autonomous systems.

3.3 Problem
The users of some system may need to access resources in another system, under the control of a different administrator. The users are not registered in the other system, how can they be recognized as legitimate users?

3.4 Forces
All the forces of the ASP apply but one of them needs to be reinterpreted:

Registration. The subject must be registered in some domain from an Identity Federation. This domain must have its own Identity Provider.

We also need a new force:

Trust. We need to have a way to establish trust to be able to accept authentication proofs from other domains.

3.5 Solution
Use a certificate validated by some recognized authority to prove identity

3.5.1 Structure
Figure 3 shows the class model of the Credential-based Authenticator pattern, a concrete type of Authenticator. The Credential-based Authentication pattern includes the complete Abstract Authenticator definition where its classes are reinterpreted as: the *Principal* corresponds to a responsible subject, Proof of Identity becomes a *Credential* carried by the Principal, Proof of Authentication is now a *Validated Credential* and the Authentication Information has a procedure to validate credentials (Fernandez 2013). A *Certification Authority* generates credentials, and the Credential includes a set of Attributes that authenticate a subject and maybe includes authorization rights and other descriptions of the subject. Authentication is performed by the Authenticator using the certificate after previously authenticating the Authority that issued the credential. In this case the identification information comes from the certification authority; the Authenticator only needs to have a list of recognized authorities.

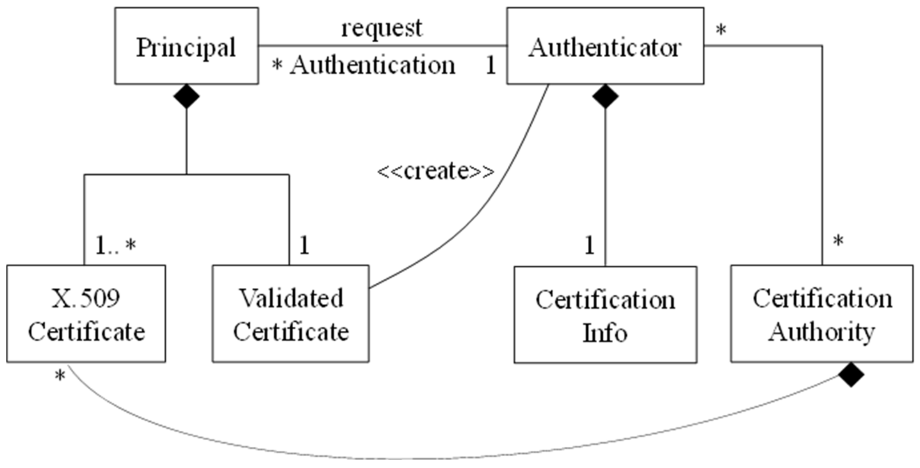

Fig. 3. Class diagram of the Credential-based Authentication pattern

3.5.2 Threats and countermeasures
The threats of the credential include concrete versions of the abstract threats; for example, replacing Proof by Credential.
New threats include:

- Using expired or invalid credentials. This can be controlled using validity dates and creating lists of invalid credentials.
- Stealing certificates. This is easier than stealing passwords because of their portable nature. Some solutions are discussed in (Fernandez 2013, Chapter 14).

3.6 Known uses

- Authentication by a country of foreign nationals based on passports is a real-life use of this pattern.
- Many distributed systems use ISO X.405 certificates.
- XML web services use SAML assertions as credentials.

3.7. Consequences
In addition to those in Section 2.7, this pattern has two advantages:

- *Registration.* This requirement provides a way for a domain to control its own subject access to the domain resources that can be extended to access other systems for some subjects.
- *Trust*. Most countries have certification authorities, recognized within the country and by other countries. (Circle of Trust pattern).

Liabilities of credentials include:

- Since they are carried in some other software they are easy to steal.
- It is not easy to revoke them so they must carry an indication of time of validity and a unique identity.
- The overall structure is more complex because of the need for a certification authority and the management of certificates.

3.8 Related patterns

- The specific credential used must be based on some standard structure. Two possibilities are X.509 certificates and SAML Assertions (Fernandez 2004). In this moment, there is no pattern for an X.509 Authenticator but we intend to write one shortly.
- The Spoofing Web Services misuse pattern in (Fernandez 2013) discusses the use of stolen credentials to access unauthorized web services.

4. CONCLUSIONS

This is the first pattern ASP published as an ASP, trying to emphasize its abstract properties. To show a possible use we derived a new pattern from it, the Credential-based Authenticator". It is possible to see that using the ASP as a guideline one can produce a more correct and complete pattern compared to producing a similar pattern directly. For example, in our first attempt to define an Authenticator pattern we omitted several forces [Fer13]. It is also much simpler because the main features of the pattern are already defined.

Future work includes patterns for X.509-based authentication and for X.509 Certificates.

ACKNOWLEDGEMENTS

Our shepherd Sumit Kalra provided valuable comments that significantly improved this paper. The work of E.B.Fernandez was supported by a grant from NII for a three-week visit to Tokyo.

REFERENCES

Ingrid Buckley, Eduardo B. Fernandez, and Maria M. Larrondo-Petrie, "Patterns Combining Reliability and Security", Procs. of PATTERNS 2011: The Third International Conferences on Pervasive Patterns and Applications, September 25-30, 2011 - Rome, Italy

F. Buschmann, R. Meunier, H. Rohnert, P. Sommerland, and M. Stal., Pattern- oriented software architecture, Wiley 1996.

N. Delessy, E.B.Fernandez, and M.M. Larrondo-Petrie, "A pattern language for identity management", Procs. of the 2nd IEEE Int. Multiconference on Computing in the Global Information Technology (ICCGI 2007), March 4-9, Guadeloupe, French Caribbean. http://www.computer.org/portal/web/csdl/doi/10.1109/ICCGI.2007.5

E. B. Fernandez, "Two patterns for web services security", Procs. of the 2004 Intl. Symposium on Web Services and Applications (ISWS'04), Las Vegas, NV, June 21-24, 2004.

E. B. Fernandez, Security patterns in practice: Building secure architectures using software patterns, Wiley, April 2013.

E. B. Fernandez, Nobukazu Yoshioka, Hironori Washizaki, and Joseph Yoder, "Abstract security patterns for requirements specification and analysis of secure systems'', Procs. of the WER 2014 conference, a track of the 17th Ibero-American Conf. on Soft. Eng.(CIbSE 2014), Pucon, Chile, April 2014

M. Fowler, Analysis patterns -- Reusable object models, Addison- Wesley, 1997.

E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design patterns –Elements of reusable object-oriented software, Addison-Wesley 1994.

D. Gollmann, Computer security (3rd Ed.), Wiley, 2011.

J. Rumbaugh, I. Jacobson, and G. Booch, The Unified Modeling Language Reference Manual, Addison-Wesley, Boston, Mass., 1999.

J.Saltzer and M.Schroeder, "The Protection of Information in Computer Systems". Proceedings of the IEEE 63, 9 (September 1975), 1278-1308.

A. V. Uzunov and E. B. Fernandez, "An Extensible Pattern-based Library and Taxonomy of Security Threats for Distributed Systems"- Special Issue on Security in Information Systems of the Journal of Computer Standards & Interfaces, vol. 36, No 4, 734–747, 2013, http://dx.doi.org/10.1016/j.csi.2013.12.008

A. V. Uzunov, E. B Fernandez, Katrina Falkner, "Security solution frames and security patterns for authorization in distributed, collaborative systems", *Computers & Security*, 55, 2015, pp. 193-234, doi: 10.1016/j.cose.2015.08.003 .