Patterns for Application Firewalls

Nelly Delessy-Gassant, Eduardo B. Fernandez, Saeed Rajput, and Maria M. Larrondo-Petrie
Dept. of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
ndelessy@fau.edu, ed@cse.fau.edu, saeed@cse.fau.edu, maria@cse.fau.edu

Abstract

Enterprise applications are typically implemented as web applications and may use web services. This provides a flexible architecture suitable for B2B communication but it may introduce new security threats. We discuss here an analysis pattern for an Application Firewall, intended to protect the applications from some of these threats. We also present a pattern for an XML Firewall, a specialization of the Application Firewall, and used to filter the contents of XML documents.

Introduction

Computer data security is an important issue for organizations. They must protect their information assets from attacks [Fer01b]. However, driven by business imperatives, they also have to open up their systems to a wide variety of partners, customers or mobile employees. This introduces a new variety of security threats.

Typical enterprise or institution applications include stock trading, health care, student registration and many others. These applications make use of the so-called application-layer in network architectures and which include for example, file transfer and mail exchange. Enterprise applications are typically implemented as web applications and may use web services. Their users, typically interacting through some internal network are the subjects that request access to resources or services.

A user can be a member of a Role, as defined in the common Role-Based Access Control Model [San96]. A Role defines a set of rights. In this case, its use makes easier the administration of the policies, as the addition of new users and new policies become mutually independent.

There are different types of network firewalls, as illustrated by Figure 1 [Fer03]. Each type of firewall depends on the inspection of information generated by the protocols that use the Internet layers.

Packet filters consider only addresses at the IP layer. Stateful firewalls keep track of established connections and filter packets based on the status of these connections. Proxy firewalls filter the use of services at the network application layer

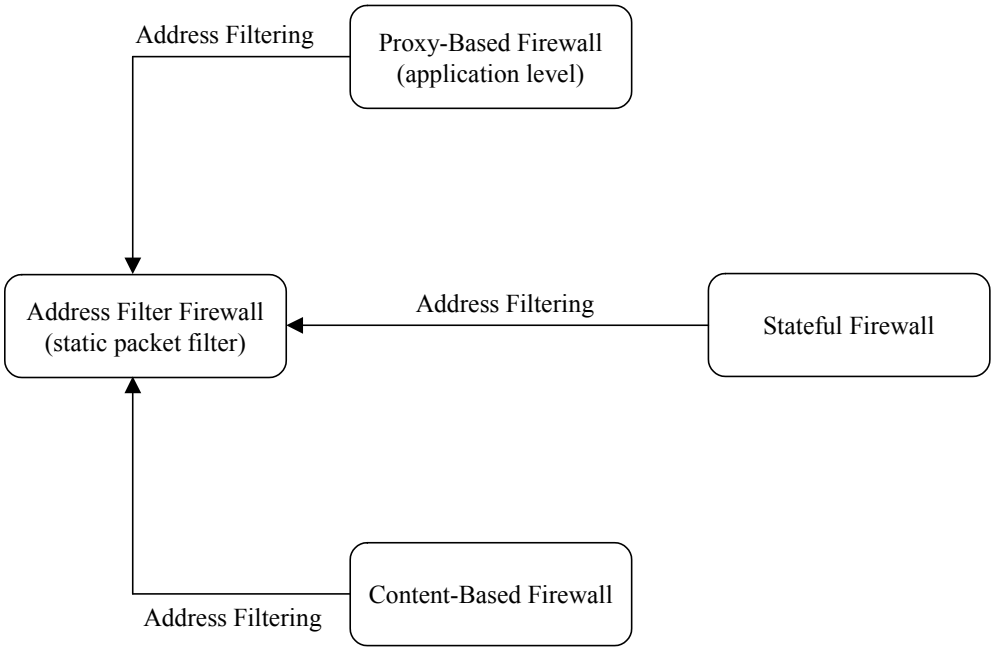

Figure 1: Firewall Pattern Language

However, the accesses to enterprise applications use formats that are not known by these firewalls. The payloads of these messages can embed harmful data. An emerging solution for shielding an organization's internal network from these new threats is to add an Application Firewall to the traditional line of defense defined by network-based firewalls.

The Application Firewall's primary goal is to enforce the organization's access control policies by filtering messages based on the users' identities or roles and the intended type of access. This filtering is realized in a centralized and uniform way for all the applications in the network, thus facilitating administration of the policies through the organization. As a complement, the XML Firewall performs XML content checking and security checks such as encryption/decryption or digital signature verification. We describe patterns for these two architectures in the next sections.

Application Firewall

Intent

To filter calls and responses to/from enterprise applications, based on an institution access control policies.

Aka

Content Firewall

Example

Consider a medical record application in a Hospital. One of the services it provides is to allow patients to lookup their personal medical records from home. To ensure that only patients can access this service, the patient must first be authenticated and then she must be identified as a patient. Finally, the application must ensure that only the medical records belonging to the patient are returned (i.e., match the name in the medical record with that of the user).

One way to provide this security is to let application maintain list of all valid patients with their authentication credentials, and implement the code for blocking unauthorized access from within the application level code of the application.

This approach has several problems. In the future, if the hospital decides to allow patients to be able to schedule appointments, it will have to repeat the implementation of the access control code for the scheduling application as well. Furthermore, if there are changes in hospital business policies, e.g. they want to allow external primary care physicians access the medical records of their own patients, these applications will have to be rewritten. In this changing scenario, a new access control list for authorized primary care physicians will have to be added to medical record application, and a list of patients will have to be associated with each physician to indicate the patients belonging to a doctor. Such application modifications are time consuming, difficult to manage, expensive and error prone.

Context

Enterprise applications executing in distributed systems accessed from a local network, the Internet, or other external networks. These distributed systems typically include packet filter and/or proxy-based firewalls.

Problem

Enterprise applications in an organization's internal network are accessed by a broad spectrum of users that may attempt to abuse its resources (leakage, modification or destruction of data). These applications can be numerous, thus implementing access control independently in ad hoc ways and may make the system more complex and thus less secure.

Moreover, traditional network firewalls (application layer firewalls or packet filters), do not make it possible to define high level rules (role-based or individual-based rules) that could make the implementation of business security policies easier and simpler.

Forces

- There may be many users (subjects) that need to access an application in different ways; the firewall must adapt to this variety.
- There are many ways to filter application inputs, we need to separate the filtering code from the application code.
- There may be numerous applications that may require different levels of security. We need to define appropriate policies for each application.
- The business policies are constantly changing and they need to be constantly updated; hence it should be easy to change the firewall filtering configuration.
- The number of users and applications may increase significantly; adding more users or applications should be done transparently and at proper cost.
- Network firewalls cannot understand the semantics of applications and are unable to filter out potentially harmful messages.

Solution

Interpose a firewall that can analyze incoming requests for application services and check them for authorization. A client can access a service of an application only if a specific policy authorizes it to do so. Policies for each application are centralized within the Application Firewall, and they are accessed through a PolicyAuthorizationPoint. Each application is accessed by a client through a PolicyEnforcementPoint that enforces access control by looking for a matching policy in the PolicyBase. This enforcement may include authenticating the client through its identity data stored in the IdentityBase.

Class diagram

Figure 2 shows the class diagram for the Application Firewall. Classes **Client** and **Service** have the usual meaning. A Client accesses a service provided by an application. The access requests are controlled by authorization rules (denoted here as *policies* to follow the usual industrial notation), and represented by class **Policy**. Policies are collected in a policy base (**PolicyBase)**.

The firewall consists of one PolicyAuthorizationPoint which centralizes the definition of the policies and identities throughout the institution, and several PolicyEnforcementPoints, which are intended to actually check the accesses to the applications.

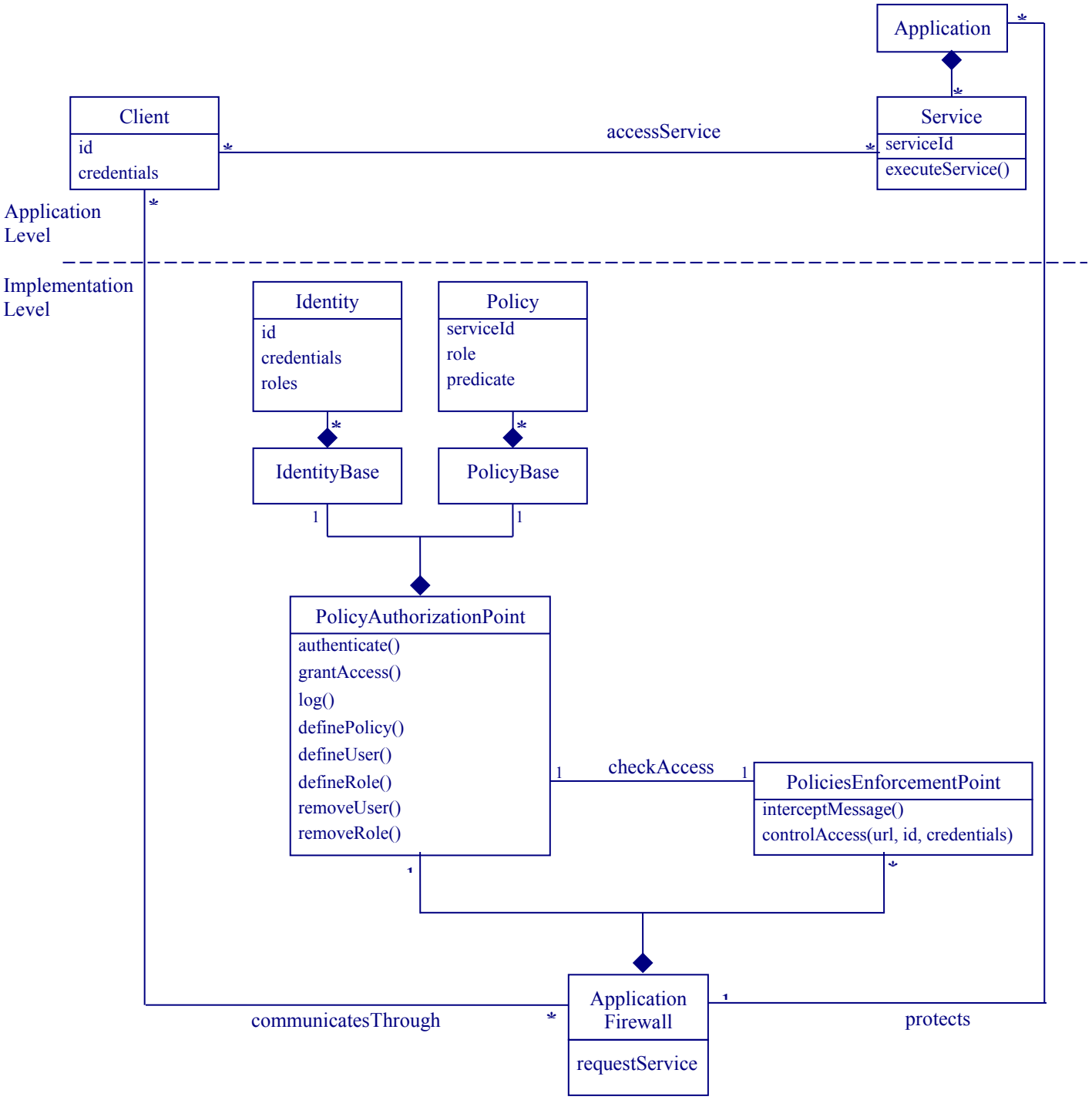

Figure 2: Class diagram for the Application Firewall (case when the policies are described through roles)

The enterprise applications are represented by the class **Application** that is made up of **Service**s. A service is identified by a serviceId, which is usually and URI or an URL.

Dynamics

We describe the dynamic aspects of the Application Firewall using sequence diagrams for two use cases: filtering a Client's request with user authentication and adding a new policy.

Filtering a Client's Request with user authentication:

Summary: A Client requests access to a service of an application to either input or retrieve information. The access request is made through the PolicyEnforcementPoint, which accesses the PolicyAuthorizationPoint to determines whether to accept or deny the request. Figure 3 corresponds to this basic use case.
Actors: A Client
Precondition: Existing IdentityBase and PolicyBase classes must be in place in the firewall. The IdentityBase contains the data necessary to authenticate a Client. The PolicyBase contains specific policies defined by the organization.
Description:

a. A Client requests access to an application.
b. An Application Firewall, through its PolicyEnforcementPoint, intercepts the request and accesses the PolicyAuthorizationPoint.
c. The PolicyAuthorizationPoint authenticates the Client through its IdentityBase. This step may be avoided for each request through the use of a Session class.
d. Once the Client is authenticated and identified, the PolicyAuthorizationPoint filters the request according to the PolicyBase. The request is accepted or denied according to the defined policies.
e. If the request is accepted, the firewall allows access to the service of the application and the access is logged into the Application Firewall.

Alternate Flows:
If the Client is not recognized or if no policy allows the specific Client to access the specified service, the firewall rejects the access request to the service.
If the user has already been authenticated, the Client may not be authenticated again (Single Sign-On use).
Postcondition: The firewall has provided the access of a Client to a service, based on verifying the identity of the Client, and the existence of a matching policy.

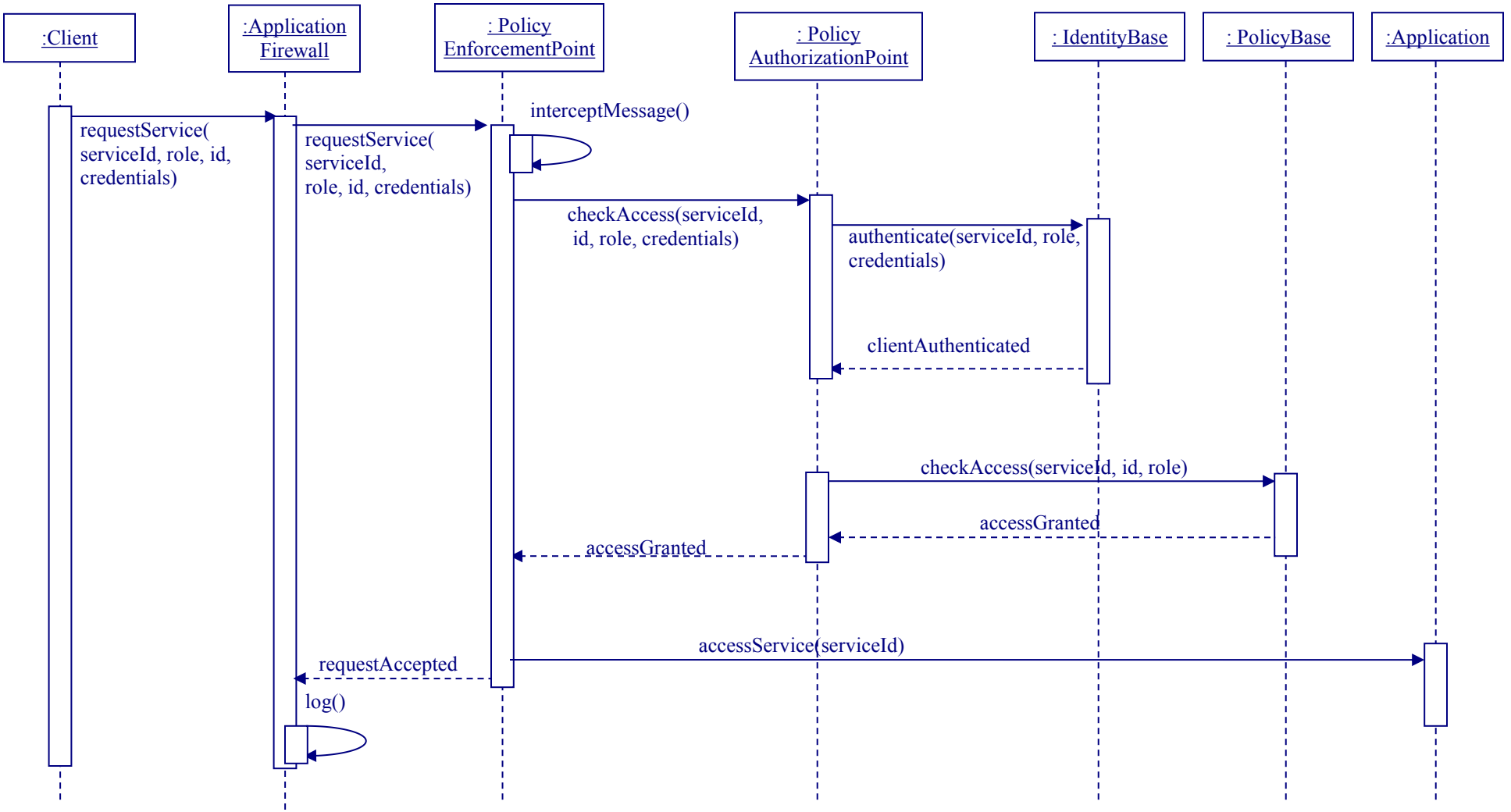

Figure 3: Sequence Diagram for filtering a Client's request with Authentication (case when the policies are described through roles)

Adding a new policy:

Summary: The security administrator intends to add a new policy to the set of policies. Before adding it, the firewall checks whether the new policy to be added does not already exist in the rule set. Figure 4 illustrates this use case.
Actors: Administrator.
Precondition: The administrator must have authorization to add rules.
Description:
a. The administrator initiates the addition of a new rule.
b. If the rule does not already exist in the rule set then it is added.
c. The firewall acknowledges the addition of the new rule.
Alternate Flow: The rule is not added because it already exists in the rule set.
Postcondition: A new rule is added to the rule set of the firewall.

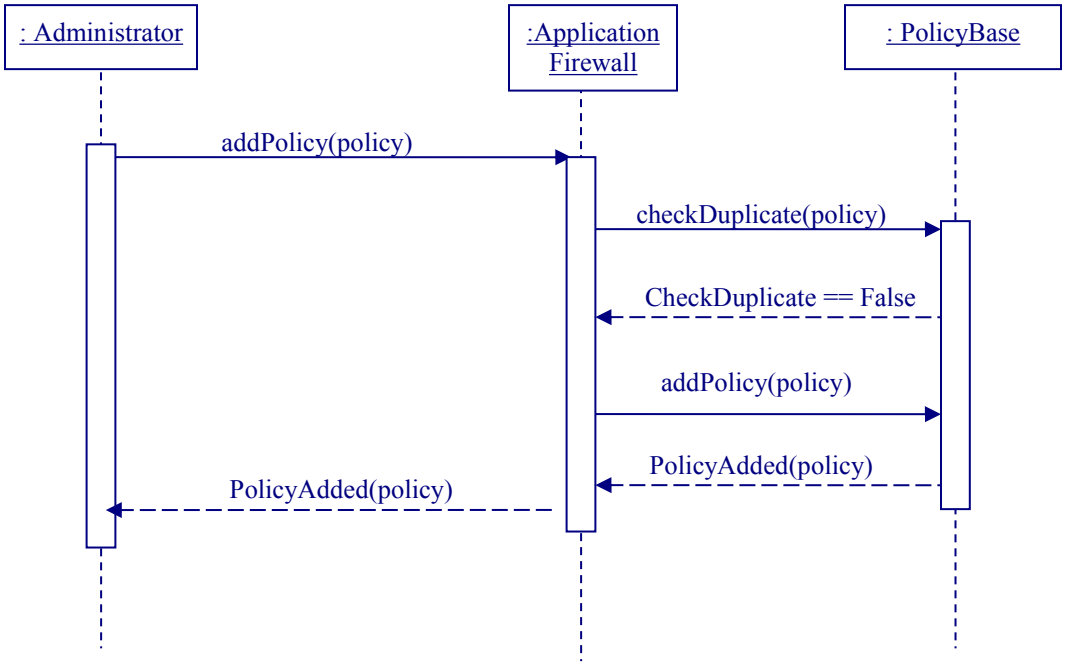

Figure 4: Sequence Diagram for defining a new Policy

Consequences

This pattern presents the following advantages:

- The institution policies to control access are easily defined and administered, as the policies have centralized administration. This makes the whole system less complex, and thus more secure.

- This firewall could be combined with an Intrusion Detection System to facilitate the prevention of some attacks.
- The firewall lends itself to a systematic logging of incoming and outgoing messages.
- As authentication of Clients is performed, users can be held responsible for their actions.
- New applications are easily integrated into the system by adding their specific policies.
- New clients can be accommodated by adding new policies to the policy base of an application.
- Because of their separation, the application and the filtering policies can evolve independently

The pattern also has some (possible) liabilities:

- The application could affect the performance of the protected system as it is a bottleneck in the network. This can be improved by considering the firewall a virtual concept and using several machines for implementation.
- The solution is intrusive for existing applications that already implement their own access control.
- The application itself must be built in a secure way or normal access to commands could allow attacks through the requests.
- We still need the operating system and the network infrastructure to be secure.

Implementation

To implement the Application Firewall, the following tasks need to be done:

1. Define users and their roles.
2. Define role rights and implement them as policies (Use Case 2).
3. Add/Remove policies when needed.

Moreover, two architectural configurations are possible and shown below.

Reverse proxy

With the reverse proxy implementation, the input flow is intercepted on a single point. There is only one PolicyEnforcementPoint, and all the flow should go through it [Som03].

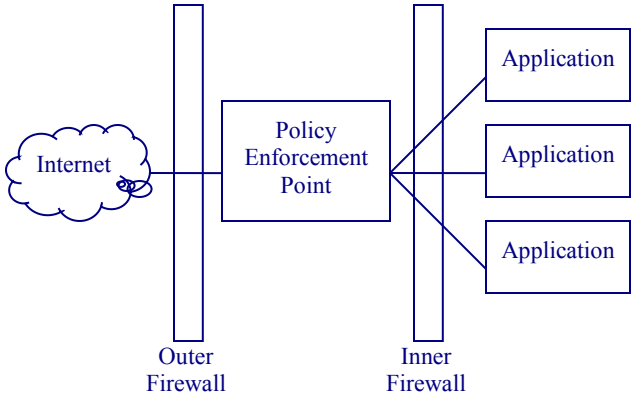

Figure 5: Reverse Proxy configuration

Multiple agents

With this implementation, several PolicyEnforcementPoints are distributed on the network, close to the different applications that have to be controlled. These enforcement points intercept every request to the application. It is also possible to control access for requests coming from internal networks.

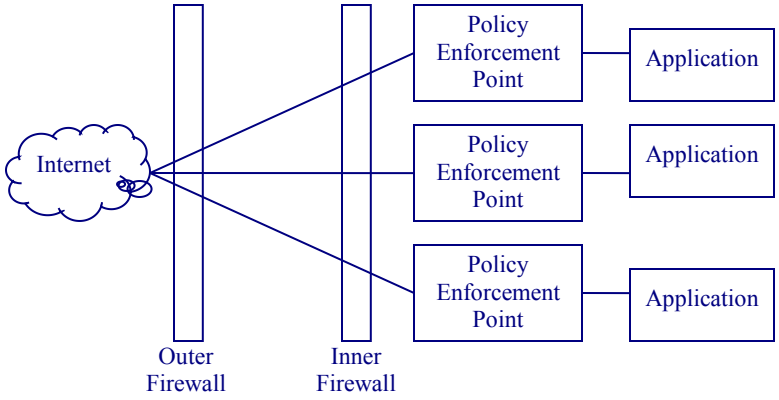

Figure 6: Multiple agents configuration

Known Uses

This pattern is used in several commercial products, such as Cerebit InnerGuard [Cer03], or Netegrity SiteMinder [Net03]. This model is also used as an underlying architecture for XML Application Firewalls (see next section). There are also some products called Application security gateways that incorporate these functions plus others.

Example Resolved

Application firewalls allow separation of the access control code from the application code. This allows reuse of the basic access control code in different applications. For example, in the example discussed earlier, the bulk of the access control code will be common to both medical and scheduling applications.

When application firewalls are used, all accesses to applications (medical or scheduling) have to pass through these firewalls. The application firewall ensures that the users are properly authenticated, and have privileges to the service that they are accessing based on configurable policies.

Related Patterns

The Authorization pattern [Fer01a] defines the security model for the Application Firewall. The Role-Based Access Control pattern, a specialization of the authorization pattern, is applicable if the business policies are respectively defined in terms of roles and rights [Fer01a]. The Application Firewall pattern is a special case of the Single-Point of-Access [Yod97]. The Reverse Proxy Pattern [Som03] defines a possible architecture to use this pattern. The PolicyEnforcementPoint is a special case of a Reference Monitor [Fer02].

XML Firewall

Intent

To filter XML messages to/from enterprise applications, based on business access control policies and the content of the message.

Context

Enterprise applications executing in distributed systems accessed through a local network, from the Internet, or from external networks. These applications communicate through XML messages and could be applications using web services. The messages can contain a remote procedure call or a document.

Problem

Some enterprise applications use tunneling into authorized flows (HTTP, SMTP,…) to communicate with the outside. They use higher level protocols such as SOAP and communicate through XML documents or XML-wrapped remote procedure calls. The

XML content of these messages can contain harmful data and can be used to perform attacks against applications.

Network firewalls provide infrastructure security but become useless when these high level protocols and formats are used.

Forces

- Document or remote procedure calls formats are subject to change, some new ones may appear (XML dialects); the firewall must adapt easily to these changes.
- New types of harmful data may be used by attackers, the firewall must adapt easily to these new types of attacks.
- There are many ways to filter, we need to separate the filtering code from the application code.
- There may be numerous applications that may require different levels of security.
- New applications may be integrated into the system after the firewall has been put into operation. This integration should not require significant additional costs.
- Network firewalls cannot understand the contents of XML messages or application semantics and do not stop potentially harmful messages.

Solution

Use a firewall that intercepts XML messages and can understand their contents. A client can access a service of an application only if a specific policy authorizes it to do so and if the content of the message is considered to be safe for the application. Policies for each application are centralized in the XML Firewall and they are accessed through a PolicyAuthorizationPoint. Each application is accessed by a client through a PolicyEnforcementPoint that enforces access control for the applications. The authorization decision may include authenticating the client through its identity data stored in the IdentityBase. It also includes looking for a matching policy for the request in the PolicyBase and checking the content of the message. First, its structure is validated through a list of valid XML schemas, and the data it conveys is checked through a HarmfulDataDetector.

Class diagram

Figure 7 shows the class diagram for this pattern. Some of the classes are similar to those of Figure 2.

They include an **IdentityBase,** a collection of the Client identities registered in the system. A **PolicyBase** stores authorization policies that define the rights of those users. A **PolicyAuthorizationPoint** collects both identity and authorization information. A **PolicyEnforcementPoint** performs access control checks. The new classes include the **ContentInspector**, which checks the content of the XML messages sent from/to the applications.

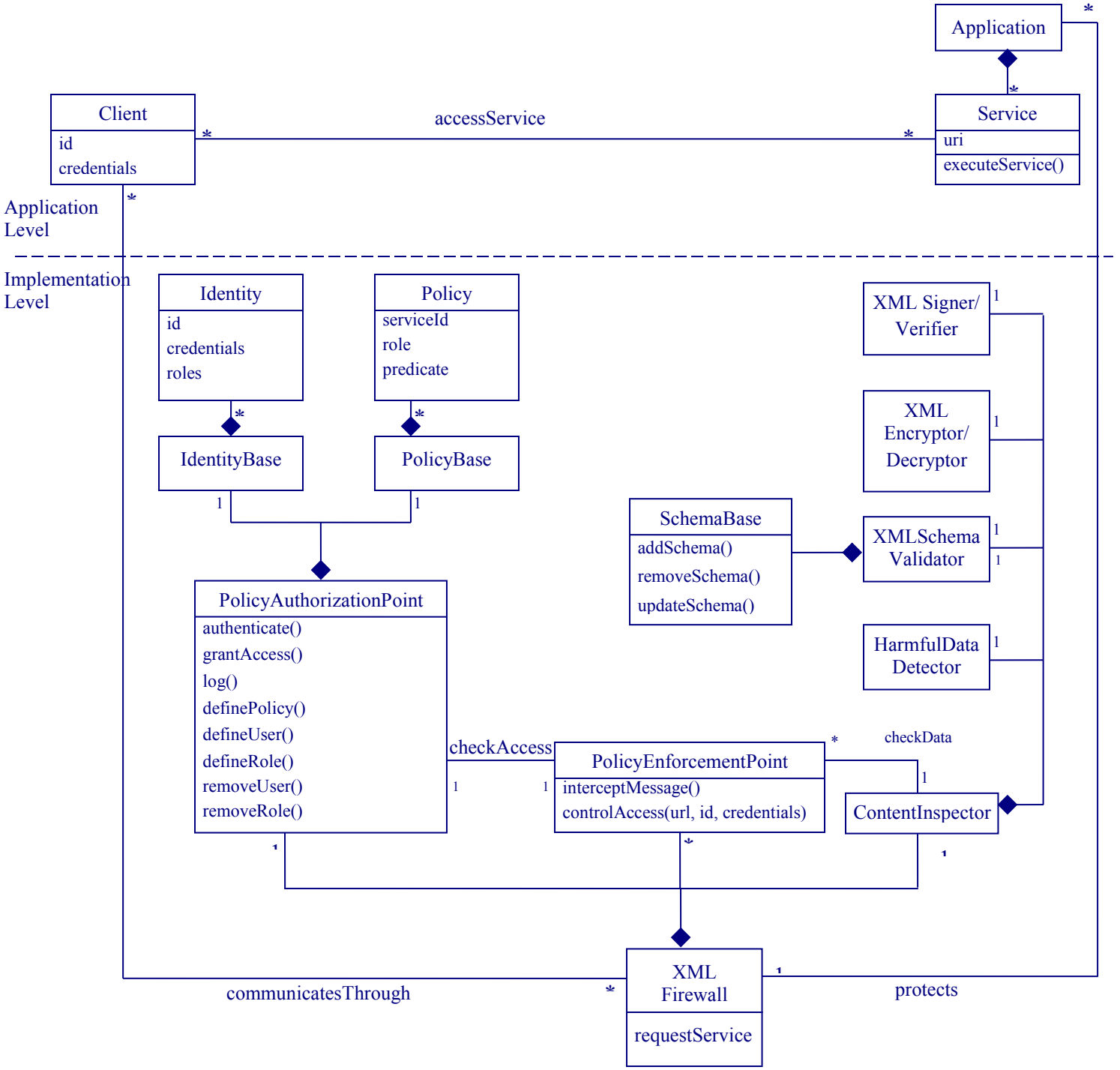

Figure 7: Class diagram for the XML Firewall

The ContentInspector consists of a **HarmfulDataDetector**, a **XMLSchemaValidator,** a **XMLSigner/Verifier**, and a **XMLEncryptor/Decryptor**. The HarmfulDataDetector perform checks for harmful data embedded in the content of the message. The XMLSchemaValidator checks the validity of the XML documents sent to the application. The XMLSigner/Verifier and XMLEncryptor/Decryptor respectively sign/verify, and encrypt/decrypt XML messages that access the Firewall, in accordance with the XML Digital Signature and XML Encryption standards proposed by W3C [W3C]. These mechanisms are used to guarantee confidentiality, data authenticity and integrity of the XML documents, as well as non-repudiation.

Dynamics

Figure 8 shows the dynamic aspects of the XML Firewall using a sequence diagram. It corresponds to a use case where the XML message is encrypted and signed, and whose user needs to be authenticated. A more basic use case would be obtained by removing some of these calls.

Filtering an encrypted and signed Client's Request, with user authentication:

Summary: A Client requests access to a service of an application to either transfer or retrieve information, through an XML message. First, the content of the message is checked so that only harmless messages are given access to the applications. Then, the access request goes through the PolicyEnforcementPoint, which accesses the PolicyAuthorizationPoint to determines whether to accept or deny the request.
Actors: External Client
Precondition: Existing IdentityBase and PolicyBase must be in place in the firewall. The IdentityDatabase contains the data necessary to authenticate a Client. The PolicyDatabase contains specific policies defined by the organization. An existing XML Schema database contains the XML Schemas trusted by the organization.
Description:

a. A Client requests access to an application.
b. An XML Firewall, through its ContentInspector, checks the validity of the XML message and decrypts it.
c. The PolicyEnforcementPoint intercepts the request and relay it to the PolicyAuthorizationPoint

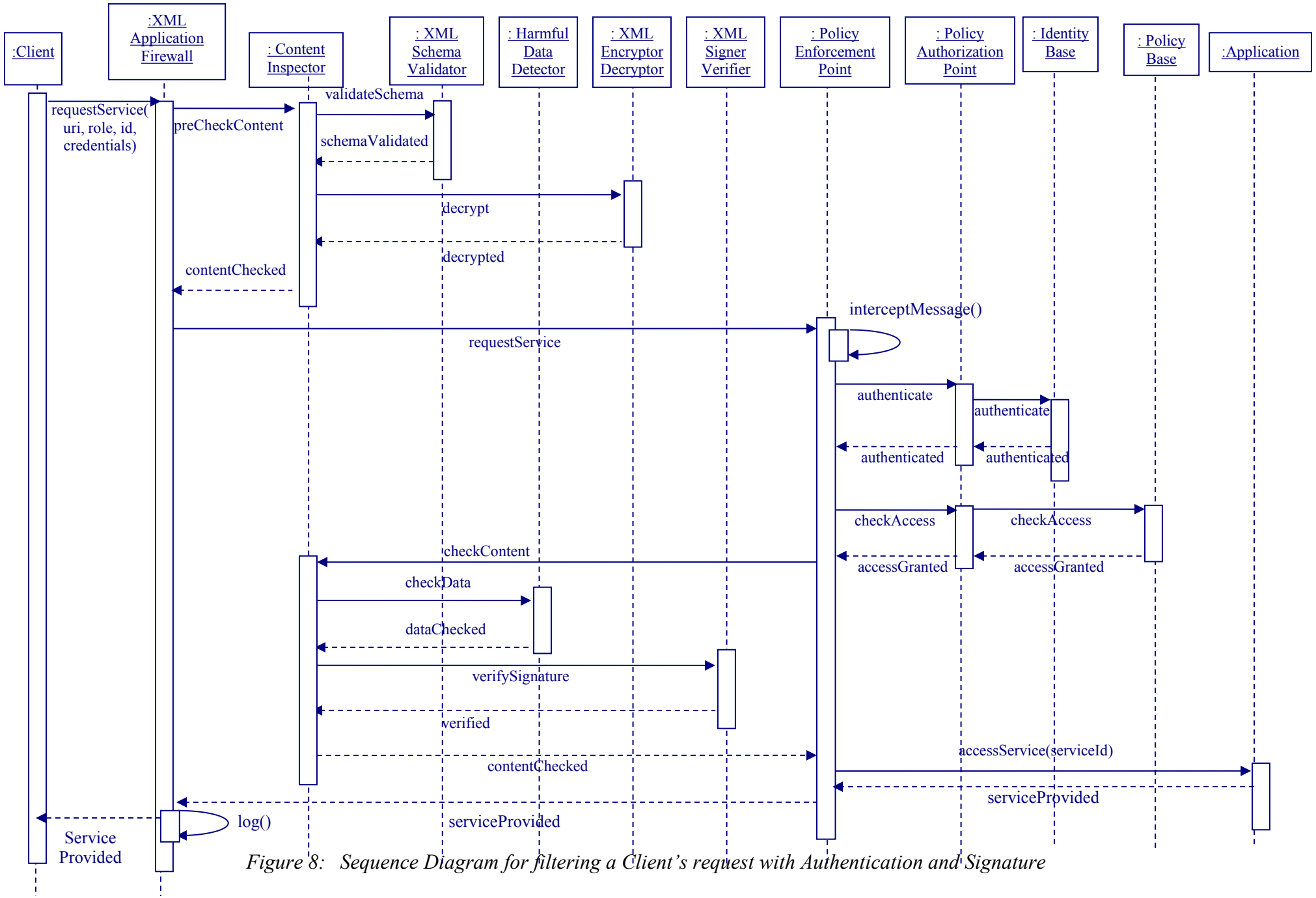

Figure 8: Sequence Diagram for filtering a Client's request with Authentication and Signature

d. The PolicyAuthorizationPoint authenticates the Client through its IdentityBase. This step may be avoided for each request through the use of a Session class.
e. Once the Client has been authenticated and identified, the PolicyAuthorizationPoint filters the request according to the PolicyBase. The request is accepted or denied according to the defined policies.
f. The contents of the message is checked . If the message contains harmful data, it is rejected.
g. The Signature of the XML Document is verified.
h. The firewall allows access to the service of the application and the access is logged into the XML Firewall.

Alternate Flow:
If the XML message is invalid, or the XML message contains harmful data, or the Client is not authenticated or no policy allows the specific Client to access the specified service, the firewall rejects the access request.
If the user has already been authenticated, the Client may not be authenticated again (Single Sign-On use).
If the signature is not verified, the request may be relayed, depending on the existing policies.
Postcondition: The firewall has filtered the access of a Client to a service, based on the content of the message, the authentication of the Client, and the existence of a matching policy.

Consequences

The XML Firewall has the same advantages of the Application firewall and the following additional advantages:

- Provides a higher level of security than the Application Firewall for inputs which are XML documents or requests.

The XML Firewall has the following (possible) liabilities:

- The application could affect the performance of the protected system as it is a bottleneck in the network, and as the XML content checking may create a large overhead. This can be alleviated by using multiple a multiple-agents configuration.
- The solution is intrusive for existing applications that already implement their own access control or their own filtering.

Implementation

The same architectural structures used for the Application Firewall (Reverse Proxy, Multiple Agents) can be used to deploy XML Firewalls.

Known Uses

This model is used in several commercial products, such as Reactivity's XML Firewall [Rea03], Vordel's XML Security Server [Vor03], Westbridge's XML Message Server [Wes03], Netegrity's TransactionMinder [Net03], DataPower's Security Gateway[] and Forum Systems Xwall [For04]..

Related Patterns

This model is a specialization of the Application Firewall presented earlier and has the same related patterns.

Acknowledgements

We thank our shepherd Robert Hanmer for valuable comments that improved the quality of our paper.

References

[Cer03] Cerebit, Inc. "Safeguarding the Enterprise from the inside out." http://www.cerebit.com/download/Cerebit-EnterpriseApplicationSecurity.pdf .

[Dat04] DataPower, http:///www.datapower.com

[Fer01a] E. B. Fernandez and R. Pan. "A Pattern Language for Security Models." *In Proceedings of the PLoP Conference, 2001*. http://jerry.cs.uiuc.edu/~plop/plop2001/accepted_submissions/accepted-papers.html.

[Fer01b] E. B. Fernandez. "An Overview of Internet Security." *In Proceedings of the World's Internet and Electronic Cities Conference* (WIECC 2001), Kish Island, Iran, May 2001.

[Fer02] E. B. Fernandez. "Patterns for Operating Systems Access Control" *In Proceedings of the PLoP Conference, 2002* http://jerry.cs.uiuc.edu/~plop/plop2002/final/OSSecPatt7.doc

[Fer03] E. B. Fernandez, M. L. Petrie, N. Seliya, N. Delessy and A. Herzberg. "A Pattern Language for Firewalls." In *Proceedings of the PLoP Conference, 2003*. http://jerry.cs.uiuc.edu/~plop/plop2003/Papers/Fernandez-firewalls.pdf

[Fer04] E. B. Fernandez, M. L. Petrie, N. Seliya, Stateful Inspection Firewall Pattern.

[For04] Forum Systems Inc., http://www.forumsys.com/

[Net03] Netegrity, Inc. “A reference architecture.” http://www.netegrity.om/pdfs/refarch.pdf

[Rea03] Reactivity. http://www.reactivity.com

[San96] R.Sandhu et al., "Role-Based Access Control models", Computer , vol. 29, No2, February 1996, 38-47.

[Som03] P. Sommerlad “Reverse Proxy Patterns”, *Procs. of EuroPLoP 2003*.

[Vor03] Vordel Limited. “An in-depth description of Vordel’s innovative VordelSecure, the XML security server.” http://www.vordel.com/downloads/VS2.1_white_paper.pdf

[Wes03] Westbridge Technology. http://www.westbridgetech.com

[W3C] W3C-IETF, “XML-Signature Syntax and Processing”, http://www.w3.org/TR/xmldsig-core/, “XML Encryption” http://www.w3.org/TR/xmlenc-core/

[Yod97] J. Yoder and J. Barcalow, "Architectural patterns for enabling application security". Procs. PLOP’97, http://jerry.cs.uiuc.edu/~plop/plop97 \. Also, Chapter 15 *in Pattern Languages of Program Design*, vol. 4 (N. Harrison, B. Foote, and H. Rohnert, Eds.), Addison-Wesley, 2000.