# Remote Authenticator /Authorizer

Eduardo B. Fernandez and Reghu Warrier
Dept. of Computer Science and Eng.
Florida Atlantic University
Boca Raton, FL, USA
ed@cse.fau.edu, reghu@dcr-inc.com

**Abstract**

Many distributed systems include a variety of nodes with different computational devices that need to access shared resources. A secure and easily manageable authentication and authorization mechanism is necessary in such an environment. We present here a pattern to achieve this objective using a remote authentication protocol. This is a composite pattern consisting of two known patterns: Proxy and Role-Based Access Control.

## Intent

Provide facilities for authentication and authorization when accessing shared resources in a loosely-coupled distributed system.

## Example

A multinational corporation may have employees, say in the US and Brazil. The user authentication and authorization information necessary to support an employee in the US is stored in the US servers and the information to support that of a Brazilian Employee is stored in Brazil servers. Now assume that an employee from the US is traveling to Brazil and has the need to access some data from the Brazilian database servers.

There are two possible ways to achieve this

1. Replicate the user information of the employee in the Brazilian Server and give her the proper authorizations to access the data.
2. Borrow the username of an employee in Brazil who has similar rights and use that username to access the required information.

Both of these solutions have their disadvantages. The system administrators will be faced with creating and managing user accounts within each of the multiple systems to be accessed in a coordinated manner in order to maintain the consistency of the security policy enforcement. If the username of another employee is borrowed, accountability is compromised

## Context

Loosely-coupled distributed systems such as the Internet, that consist of a variety of computational nodes, and where some nodes need to share resources. For example, a company with several divisions in different countries.

## Problem

How can we provide authentication and authorization in a distributed environment without the need for redundant user login information?

In the past few years, telecommuting, the Internet, and electronic commerce have developed from an alternative means of doing business to become increasingly mainstream consumer activities. The concern for corporate data security has grown tremendously and the need for single user sign on to multiple domains and multiple services is becoming more of a necessity than a luxury. A system with a centralized sign-on can provide easy management, more accountability and secure authentication.

## Forces

- Storing user authentication and authorization information at multiple locations makes them redundant, difficult to administer, and prone to inconsistencies.
- Although the authentication information may be stored anywhere, this location should be transparent to the users.
- Users typically work in the context of some role and these roles should be standard across a variety of domains, at least within a company or institution.
- Borrowing the login rights of a local user makes it impossible to make the user accountable, we need a way to keep the user id when he is accessing resources anywhere.

## Solution

Set up a single entry point that can transparently redirect the user to the correct server where his user login and access information can be validated.

We can achieve this redirection by using a specialized authentication/authorization server. This server is used for embedded network devices such as routers, modem servers, switches, etc. The authentication servers are responsible for receiving user connection requests, authenticating the user, and then returning all configuration information necessary for the client to deliver service to the user. Figure 1 shows this approach. The **Client** makes a request for a service through a **Proxy Server** that represents the actual server that contains the user login information. The request is routed to the **Remote Server,** which validates it, based on the **Role** of the **Subject** of the request and the **Rights** of this role with respect to the **Protection Object.**

## Dynamics

Typical systems use the following types of messages:

**Access-Request.** Sent by a client to request authentication and authorization for a network access connection attempt.
**Access-Accept.** Sent by a server in response to an Access-Request message. This message informs the client that the connection attempt is authenticated and authorized.
**Access-Reject.** Sent by a server in response to an Access-Request message. This message informs the client that the connection attempt is rejected. A server sends this message if either the credentials are not authentic or the connection attempt is not authorized.

**Access-Challenge.** Sent by a server in response to an Access-Request message. This message is a challenge to the client that requires a response.
**Accounting-Request.** Sent by a client to specify accounting information for a connection that was accepted.
**Accounting-Response.** Sent by the server in response to the Accounting-Request message. This message acknowledges the successful receipt and processing of the Accounting-Request message.

A message consists of a header and attributes. Each attribute specifies a piece of information about the connection attempt.

## Class Diagram

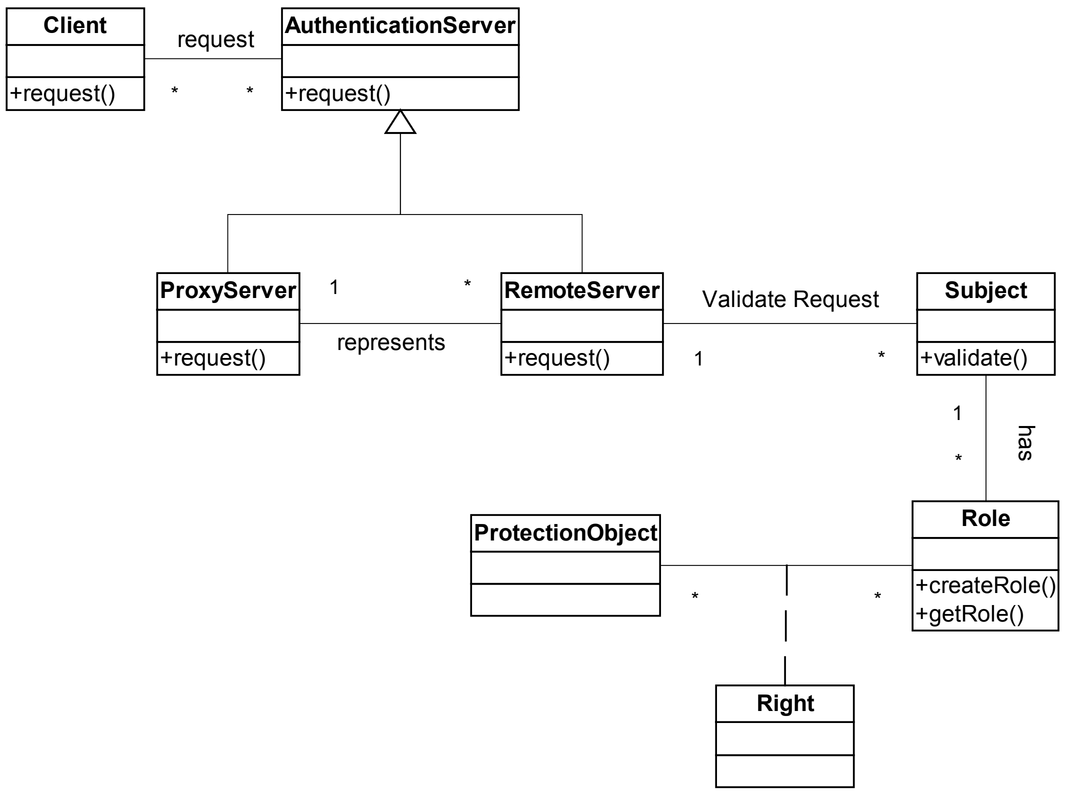

Fig 1 - Class Diagram for the Remote Authenticator/Authorizer pattern

The following scenario (Figure 2) illustrates a proxy-based communication between a client and the forwarding and remote servers:

1. A client sends its access-request to the forwarding server.
2. The forwarding server forwards the access-request to the remote server.
3. The remote server sends access-challenge back to the forwarding server.

4. The forwarding server sends the access-challenge to the client.
5. The client calculates a response for the challenge and forwards it to the forwarding server via a second Access-Request.
6. The forwarding server forwards the access-request to the remote server.
7. If the response matches the expected response the remote server replies with an Access-Accept, otherwise an Access-Reject.
8. The forwarding server sends the access-accept to the client.

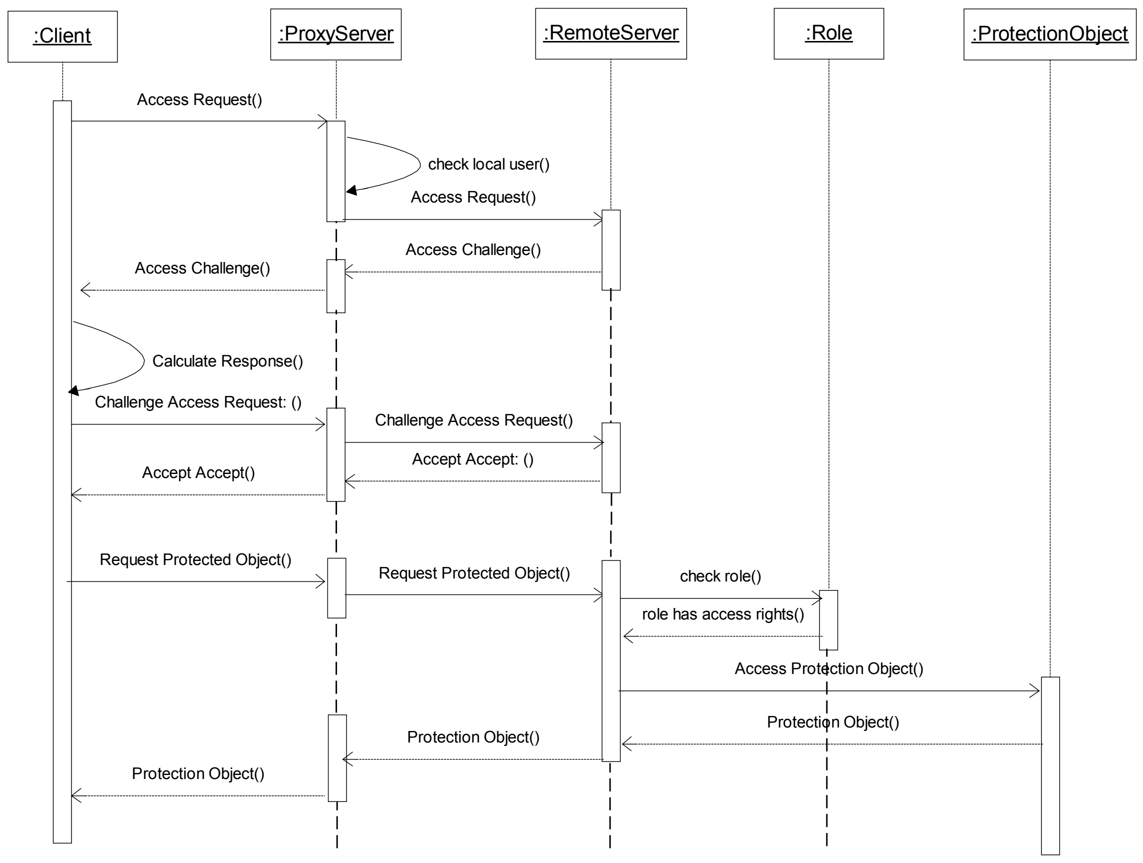

Fig 2 – Sequence Diagram for Client Authentication

## Implementation

An authentication server can function as both a forwarding server and a remote server, serving as a forwarding server for some realms and a remote server for other realms. One forwarding server can act as a forwarder for any number of remote servers. A remote server can have any number of servers forwarding to it and can provide authentication for any number of realms. One forwarding server can forward to another forwarding server to create a chain of proxies. A lookup service is necessary to find the remote server.

**Consequences**

This pattern has the following advantages:

- Roaming permits two or more administrative entities to allow each other's users to dial in to either entity's network for service.
- Storing the user login and access rights at a single location makes it more secure and easy to maintain.
- The user's login ID, password etc. are stored in the internal radius database or can be accessed from an SQL Database.
- The location where the user information is stored is transparent to the user.
- Roles and access rights have to be standard across locations.
- Both servers and clients should support the base  protocol.
- Units such as active cards [ACS] allow  complex request/challenge interactions.

There are also some liabilities:

- The additional messages used increase overhead, thus reducing performance for simple requests.
- The system is more complex than a system that directly validates clients.

**Example resolved**

When the US employee travels to Brazil he logs in a Remote Authenticator/Authorizer which reroutes her requests to the US server that stores her login information.

**Known Uses**

Remote Authentication Dial-In User Service (RADIUS) is a widely deployed IETF protocol enabling centralized authentication, authorization, and accounting for network access [Has02, Rig00]. Originally developed for dial-up remote access, RADIUS is now supported by virtual private network (VPN) servers, wireless access points, authenticating Ethernet switches, Digital Subscriber Line (DSL) access, and other network access types [Hil]. Figure 3 shows the typical authentication sequence of a client in a RADIUS server using a challenge response approach..

With proxy RADIUS, one RADIUS server receives an authentication (or accounting) request from a RADIUS client (such as a NAS), forwards the request to a remote RADIUS server, receives the reply from the remote server, and sends that reply to the client.  A common use for proxy RADIUS is roaming.  Roaming permits two or more administrative entities to allow each other's users to dial in to either entity's network for service.

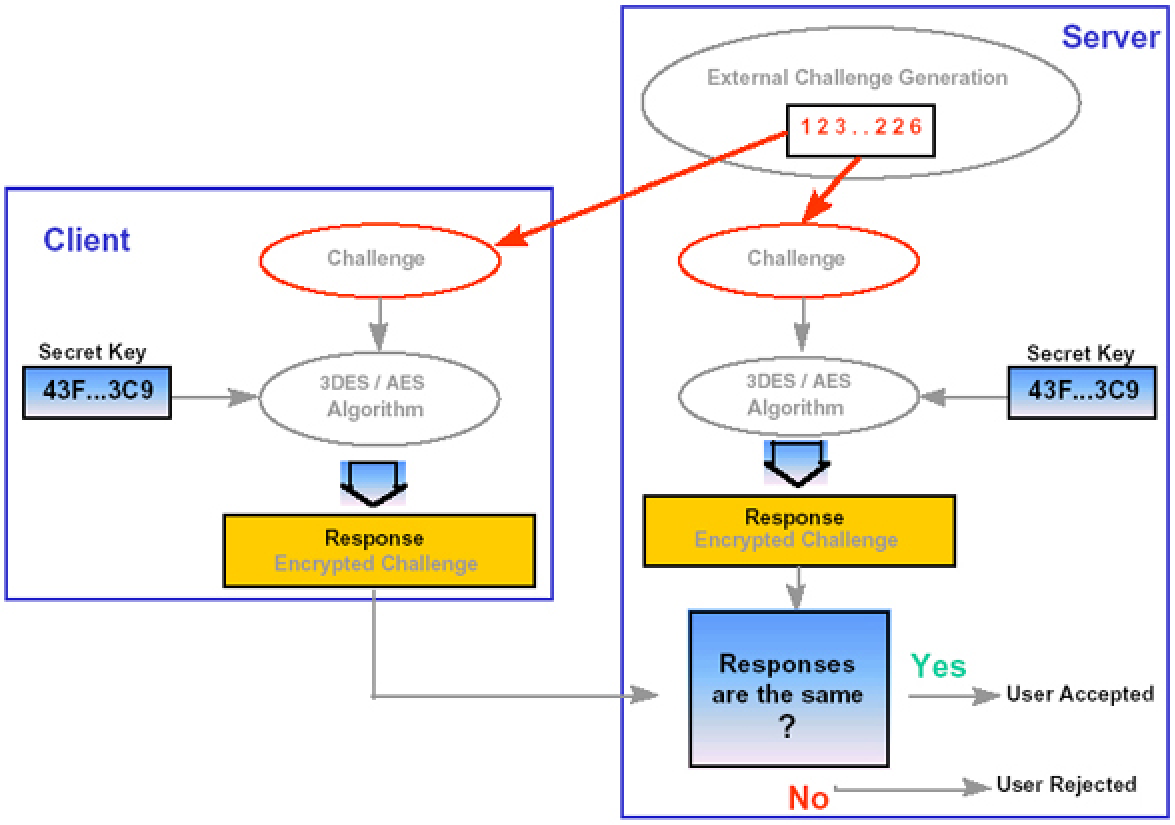

Figure 3 – RADIUS Challenge/Response authentication

There are many commercially available RADIUS Servers that are in use today. These include:

*FreeRADIUS*

FreeRADIUS Server [Fre] is a daemon for Unix operating systems which allows one to set up a radius protocol server, which is usually used for authentication and accounting of dial-up users. FreeRADIUS is an open-source product, and has all the benefits open-source provides.

*Steel-Belted Radius*

Steel-Belted Radius is a complete implementation of RADIUS. It provides full user authentication, authorization, and accounting capabilities.
Steel-Belted Radius fully supports proxy RADUS; it can:

- Forward proxy RADIUS requests to other RADIUS servers

- Act as a target server that processes requests from other RADIUS servers
- Pass accounting information to a target server, either the one performing the authentication or a different one.

*NavisRadius*

NavisRadius is an implementation of the RFC standard RADIUS protocol that provides Authentication, Authorization and Accounting (AAA) services. NavisRadius provides an integrated network-wide remote access security solution for service providers and carriers. NavisRadius supports the RADIUS standard as defined by the IETF RADIUS RFC 2865 (RADIUS Authentication) and 2866 (RADIUS accounting) and is used to provision a wide range of network services.

Earlier authentication servers were used in products of CKS, MyNet, and Security Dynamics [CTR96].

## Related Patterns

The whole architecture is an application of the single-point-of-check pattern [Yod97]. It uses the Proxy pattern [Gam95] as a fundamental component. Finally, user rights may be defined using a Role-Based Access Control model [Fer01, Yod97]. A pattern for authenticating distributed objects is given in [Bro99].

## References

[ACS] ActivCard Synchronous Authentication (http://www.activcard.com/activ/services/library/synchronous_authentication.pdf)

[Bro99] F.L. Brown and E.B. Fernandez, "The Authenticator pattern", *Procs. PLoP99.*

[CTR96] "Security: Resellers getting the advantage of growth", Computing Technology Review, Febraury 1996, 14-17.

[Fer01] E B. Fernandez and R.Y. Pan, "A pattern language for security models", *Procs. of PLoP 2001*, http://jerry.cs.uiuc.edu/~plop/plop2001/accepted_submissions/accepted-papers.html, http://www.cse.fau.edu/~ed/SecModels.pdf

[Fre] http://www.freeradius.org/mod_auth_radius/

[Gam95] E. Gamma, R. Helm,R. Johnson, and J. Vlissides, *Design patterns –Elements of reusable object-oriented software*, Addison-Wesley 1995.

[Has02] J. Hassell, *RADIUS*, O'Reilly, 2002.

[Hil] J. Hill, "An Analysis of the RADIUS Authentication Protocol", InfoGard Laboratories, http://www.untruth.org/~josh/security/radius

[Rig00] C. Rigney, S. Willens, A. Rubens, W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", June 2000. http://www.ietf.org/rfc/rfc2865.txt

[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.