Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 1, 2024

Compare and Analysis of Different Access Controlling Mechanisms in OAuth 2.0

Anri Morchiladze

Affiliated Associate Professor, International Black Sea University

amorchiladze@ibsu.edu.ge

Abstract

This article analyses the basic concepts associated with authorization methods and how existing solutions face the common problems in the modern world. Different possible methods are introduced to solve such kind of problems. This paper proposes a model for attribute-based access control for cross-domain sources using APIs. The model includes basic architectural decisions and principles of ABAC (attribute based access control), RBAC (role based access control) and OAuth. Within the capabilities of OAuth 2.0 and ABAC will allow you to implement an end-to-end security model that can protect the privacy of customers and employees, the most important transactions for the financial sectors, business, and in general the most sensitive data over the API gateway. It is also possible to filter the response message, which is very important for customers.

Keywords: oauth, sso, authorization, authentication, role management, permissions

Introduction

Authorization and access control are one of the most important parts of modern automated systems, as they directly affect the security and control of access to certain parts of the system for different groups of users. This is especially true in the growing trend of finding access control vulnerabilities against the most important security risks of web applications [1]. Organizations rely heavily on their information and the technologies that protect and use it. Organizations' dependence on information and technology makes them vulnerable to cyber threats: therefore, managers of these organizations must take proper precautions and implement appropriate best practices to reduce the risk of insider threats and other cyber threats. Employees shouldn't have full access, especially if they only need a basic account to perform a specific task. Giving users network-wide privileges can have unfortunate consequences, as disgruntled employees can become insider threats and use poor security policies to bypass all layers of security in place in the organization. Deploying a strong security policy can be an overlooked process and is often not addressed until it is too late. So, this is quiet important.

Access Control Mechanisms

Access control is a process that uses mechanisms to grant access to certain resources, applications, or a system. Computer

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 1, 2024

security architects and administrators deploy Access Control Mechanisms to meet their security requirements when processing subject requests. These access control models provide a structure and a set of conditions upon which objects, subjects, operations, and rules can be combined to make and enforce an access control decision. Each model has its own advantages and limitations, and currently the two main access control models are the most popular: the role-based access control (RBAC) model and the attribute-based access control (ABAC) model.

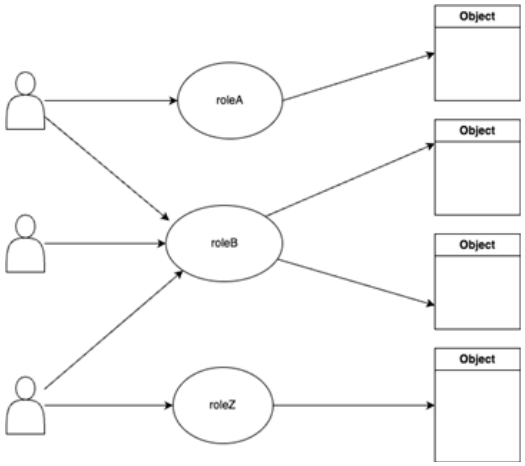

Figure1. Architecture of RBAC

RBAC is the most preferred access control model for the local domain. In this case, the user's access rights are determined by the roles specified for that user.

Roles are nothing more than abstractions of user behavior and assigned responsibilities. To provide access control and security in specific software systems, it is beneficial to use the concept of roles. It also reduces the overhead of privilege management [2].

The common disadvantages of the role based system are the huge number of roles in organization and it can't support restrictions based on current environment settings.

ABAC is a next-generation authorization model that provides dynamic, context-sensitive access control. In ABAC, permissions to access objects are not granted directly to the subject; this is done using two key elements: attributes and deployment architecture. Attributes are the basis for ABAC. In other words, attributes are key-value pairs, where the key represents the identifier of the attribute. Attributes can have multiple values. Attributes alone are not enough to express authorization logic. They must be linked to each other using policies. The key difference between AVAS is the ability to define a complex logical set of policies, with the help of which many different attributes will be assessed [3].

Natural Language Processing is set of high-level requirements that define how access to information is controlled and who can access what information under what circumstances. NLPs are expressed

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 1, 2024

in human-readable terms and may not be directly implementable in ACM. NLP can simultaneously be specific to a particular application and therefore must be taken into account by the application provider, and NLP can also describe possible actions of the subject in the context of corporate policies. To improve operational efficiency and simplify the specification, it may be necessary to decompose NLP and translate it into different digital policy versions that correspond to the infrastructure of the operating units of the enterprise.

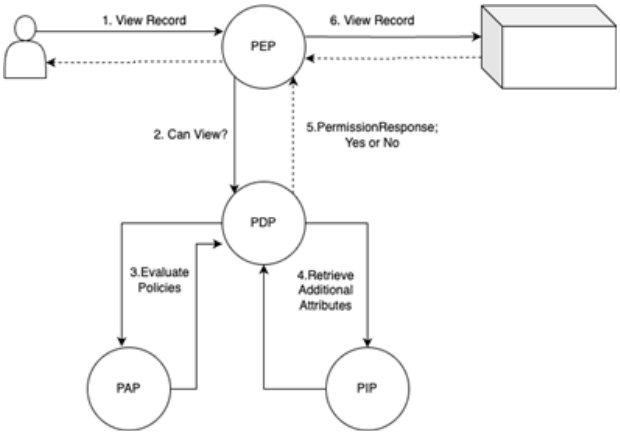

Figure 2. Architecture of ABAC

Finally, let's look at the distribution and management of multiple access control mechanisms. Depending on user needs, enterprise size, resource allocation, and privacy levels of the objects that need to be accessed or shared, the use of multiple ACMs can be critical to the success of an ABAC implementation. ACM functional components can be physically and logically separated and distributed within the enterprise system. Within ACM, there are several functional "points" that are the basis for policy retrieval and management, along with some logical components for handling the context or process of policy retrieval, various attributes, and evaluation of results. The main functional points are: Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Information Point (PIP), and Policy Administration Point (PAP). When these components are in the same environment, they must function in concert to make access control decisions. The PDP evaluates the DP and multiple MPs to make access control decisions.

PDP and PEP can be physically and logically separated within an organization. For example, an organization could create a centrally managed enterprise decision service that evaluates attributes and policies and makes decisions that are then passed to the PEP as approvals. This allows centralized management of subject attributes and policy, but provides partial control of object access by the local object owner. In order for PDPs and PEPs to perform their roles, they must be able to have information about the attributes and policies that must be applied. These functions are performed by PIP.

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 1, 2024

Let describe the working algorithm of the current model. When an access request is sent from a subject to perform a specific action on an object, it is intercepted by the PEP, which converts the application request into an authorization request and sends it to the PDP, which is an authorization decision engine that uses the information provided in the request and policies to decide whether the request should be allowed or not. PDP uses PIP to find attributes that are referenced by policies and therefore needed to make a decision on an authorization request. A Policy Administration Point (PAP), as the name suggests, is an architectural entity that is used to manage policies that are later evaluated by the PDP. It allows you to create, deploy and manage policy changes.

Traditionally, security administration for large systems has been simplified using a role-based access control approach that scales better than other previous models. By treating roles and identities as characteristics of a principal, attribute-based access control fully embraces the functionality of both ABAC and RBAC and can define permissions based on virtually any security-relevant characteristic.

Oauth 2.0

Today, more and more sensitive transactions and data are accessible through APIs, which have become the most commonly used method for allowing customers, employees, business partners and services to connect to internal processes, services and data. The gateway's basic security capabilities may be sufficient for simple or basic use cases, but organizations that need to securely access and share highly sensitive data will likely need additional capabilities. Most API gateways provide integration capabilities with security solutions that support a variety of standardized access control mechanisms, such as OAuth, for identity verification, token management, and other scenarios.

OAuth 2.0 is an open authorization protocol that allows you to grant one service (application) rights to access user resources on another service. The protocol eliminates the need to transfer a login and password to the application, and also allows you to issue a limited set of rights, and not all at once [3], by providing the service with an access token. In OAuth 2.0 exists four main parts: owner of the resource, clients, resource server and authorization server.

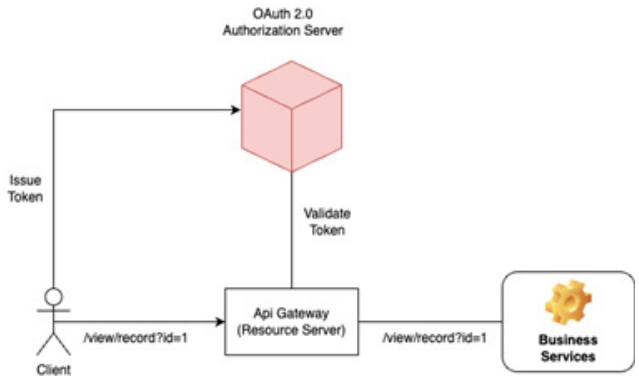

Figure 3. API Gateway interface with OAuth 2.0

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 1, 2024

The authorization server verifies the authenticity of the information provided by the client and then creates authorization tokens for the application, with which the client will access the resource server. An authorization grant is a credential issued by the owner of a resource for access to protected resources, and is used by the client to obtain an access token. An access token is a credential used by a client to access protected resources. An access token is a string that represents the authorization issued to the client. The string is typically treated as unformatted data by the client. Tokens are intended for a specific area and a specific access method, issued by the owner of the resource and used (taken into account) by the resource server and the authorization server.

The OAuth 2.0 protocol is typically used for third party authorization to gain permission to access a resource server. The scope will then be placed on the generated token when authentication is done, from that point on the origin will only check the token and the scope embedded within it to check if the user is allowed to obtain the required source of information [4].

Attribute Based Access Protocol in OAuth 2.0

To ensure that the resources of those who can perform certain transactions or who can access and use certain data via an API are protected, an API gateway solution must be complemented and extended by a dynamic policy-based authorization solution so that organizations can provide access control to specific resources. This means that access control, for example, can be applied to individual documents, bank accounts (personal, shared or delegated), patient records and etc.

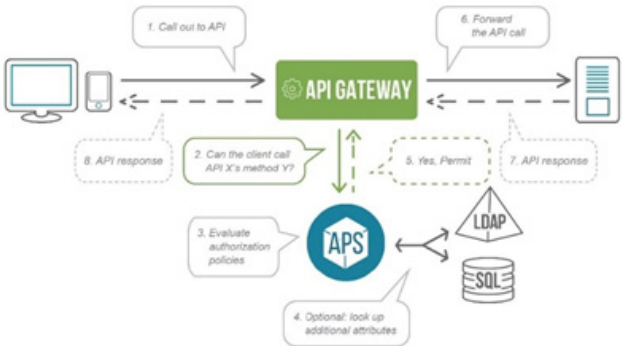

Figure 4. ABAC architecture with API Gateway as enforcement point

Consider the ABAC authorization service as a microservice or a set of microservices. This will ensure that the architecture and deployment are compatible with microservices-based applications. The ABAC service has the typical characteristics of a microservice.

- Does not maintain any state, processes requests without storing any state information
- Immutable
- Rest/JSON support
- Limited Context

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 1, 2024

Combining the capabilities of OAuth 2.0 and ABAC will allow you to implement an end-to-end security model that can protect the privacy of customers and employees, the most important transactions for the business, and the most sensitive data over the API gateway. However, OAuth scopes are fairly static and do not support any language for expressing authorization policies. Writing authorization policy logic using OAuth will result in organizations embedding authorization logic into APIs. This creates a tight coupling between API logic and authorization logic, making them difficult to manage and audit. Thus, OAuth scopes are mainly suitable for granular and functional access control, such as which users can perform payment transactions or view patient logs.

A typical deployment configuration consists of an OAuth 2.0 authorization server and an API gateway acting as a resource server. We will assume that this function is separate from the business service. You can think of this as the second and third stages of the authorization code grant flow, where the client is issued an access token that is used in subsequent calls to the business service. The API Gateway validates the access token before forwarding the request to the business service. OAuth itself does not provide a good policy enforcement mechanism for business services containing sensitive data, which is a common scenario in the industry.

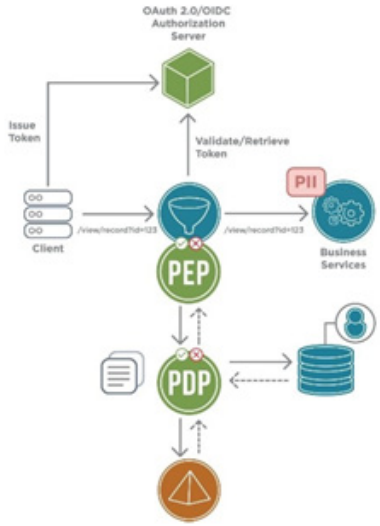

Figure 5. API Gateway mediates ABAC – OAuth data flows

The API Gateway plays a key role in the deployment of APIs/microservices, providing many security, management and operational capabilities. When advanced access control requirements must be met, the gateway can be easily configured to call the ABAC service to determine whether API access should be granted or denied, as shown in Figure 4. All access policies are centrally managed and enforced in the ABAC service instead hard coding this logic into the gateway - or worse, into the API itself. There can be

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 1, 2024

multiple business services protected by a single API gateway, and the policy applied can be dynamic and apply to many/all of them. API Gateway can be configured to call the authorization service in scenarios where additional policy evaluation is required. For these scenarios, the gateway creates a formatted JSON message with the corresponding data (SubjectId, method, authentication level, resource requested, etc.) and sends it to the authorization service via REST endpoint. During policy evaluation, the authorization service may include additional contextual information, such as determining whether the SubjectId is associated with the requested resource (if any), checking the client's current status, assessing current risk, and so on. The resulting solution is sent back to the gateway, where it is analyzed and executed. This level of integration requires no custom coding, just gateway configuration.

All ABAC and OAuth components come together in this configuration option. In Figure 5, the API Gateway has a lot of information at its disposal just before it makes an API call on behalf of the client. For example, the gateway has scope information and perhaps uses the OAuth userinfo endpoint to collect additional data about the logged-in user in a JSON Web Token (JWT). Attributes about the user, along with information about the API being called, are packaged into a message for the ABAC service to be evaluated at the enforcement point (PEP). The JWT token can also be forwarded to a policy decision point (PDP). The PDP then uses the Policy Information Point (PIP) to look up the attributes referenced by policies and the Policy Administration Point (PAP) to find the required policies to base its decision on granting the request.

As noted earlier, the ABAC service can access any additional context (risk assessment, device information, etc.) to make an authorization decision before sending the result back to the gateway for enforcement. API security is not a one-way process. All previous scenarios focus on applying authentication and access control to an incoming API call. It should also be possible to filter the response message. If the API call is to retrieve data records from a student, bank, or medical record, they may contain sensitive or private data elements that should be filtered based on the caller's credentials. Therefore, you must be able to configure the API Gateway to call the ABAC service to define any field filtering that should be applied before returning the record to the calling user or application. In this case, the gateway sends metadata (such as region, PII flag, etc.) from the data record to the ABAC service to decide whether to forward. The ABAC service returns a set of decisions or filtering patterns that the gateway applies to the data record.

Conclusion

Authorization and access control are one of the most important parts of modern automated systems, as they directly affect the security and control of access to certain parts of the system for different groups of users. This is especially true in the growing trend of finding access control vulnerabilities against the most important web application security risks.

Journal of Technical Science & Technologies; ISSN: 2298-0032; e-ISSN:2346-8270; Volume 8, issue 1, 2024

This article looked at the possibilities of using role-based access control (RBAC) and attribute-based access control (ABAC) together using the open authorization protocol OAuth 2.0. The basic concepts associated with authorization models, attributes, and problems encountered by existing authorization methods, as well as possible methods for solving them, are considered.

This paper proposes a model for attribute-based access control for cross-domain sources using APIs. The model includes the basic architectural solutions and principles of ABAC and OAuth, and contains all the necessary tools and infrastructure required to solve the problems of authentication, authorization and attribute-based access control.

References

OWASP Top Ten, “Top 10 Web Application Security Riskss”, https://owasp.org/www-project-top-ten/, accessed at 2023.

H. L. F. Ravi S. Sandhu, Edward J. Coyne, and C. E. Youman. Role-based access control models. IEEE Computer, 29(2): 38-47, February 1996.

H. Jin, R. Krishnan, and R. S. Sandhu. A unified attribute-based access control model covering DAC, MAC and RBAC. DBSec, 12:41-55, 2012.

Bilbie, A., “A Guide To OAuth 2.0 Grants,” https://alexbilbie.com/guide-to-oauth-2-grants/, accessed at November 2023.

D. Hardt, Ed, “The OAuth 2.0 Authorization Framework,” IETF, 2012.

David Brossard, Gerry Gebel, Mark Berg, “A Systematic Approach To Implementing ABAC”, 2022.