ACCESS CONTROL MANAGEMENT: RBAC AND OPA

Rupesh Juyal[1], **Pratibha Pal**[2]

[1]*PG Student, Department of MCA, Vivekanand Education Society's Institute of Technology, Mumbai, India*
[2]*PG Student, Department of MCA, Vivekanand Education Society's Institute of Technology, Mumbai, India*

---***---

Abstract: *In the current world of technology, security is at its top. May it be a web-based or mobile-based application there are hackers everywhere and to prevent it, the security of the application is increasing rapidly. This research paper is about Access Control Management. Every application has a login page and the purpose of the login page is to distinguish existing users(customers) from the unknowns. The other use of the login is to protect the data within our database. Access Control Management is used for the same purpose. It is to identify which part of the information is important and which user can access the information. In this research paper, we will talk about RBAC aka Role-Based Access Control, OPA aka Open Policy Agent, and also their difference and ways to improve them.*

Keywords: Access Control, RBAC, ABAC, OPA, Pre-evaluation OPA.

INTRODUCTION

Security can be achieved in two ways authentication and authorization. In this paper we will talk about the way we can log in, that is how can one know whether the user is genuine and the later one talks more about once the user is logged-in, which service a user can get access to. What are the permissions? What authority does the user have? Access Control talks about the later part that is an authorization.

Role-Based Access Control (RBAC) is a way to stop users to access files that are not allowed. It is used by the majority of organizations currently in the market[3]. Role-Based Access Control (RBAC) is an access control mechanism defined around rules and privileges. There are various components in RBAC like user-role, role-permission which makes it easier to understand and implement this approach. A study by NIST has demonstrated that RBAC addresses many needs of economic and government organizations [5].

The Open Policy Agent (OPA) is an open-source, policy engine that joins policy enforcement across the stack.[4] OPA is hosted by the Cloud Native Computing Foundation (CNCF) as an incubating-level project.[6] As the name suggests OPA is a policy engine i.e. every organization has its own set of policies, rules, do's, and don'ts OPA helps them to apply those rules in a general form that makes sure everything is happening within the boundaries. OPA has its own language to use. This is a high-level declarative language called REGO (pronounced as "ray-go").[4] REGO lets you specify policy as code and simple APIs to offload policy decision-making from your software. Once the policies are written OPA will make sure all the policies specified are followed.

LITERATURE

Access Control

In industries whether large or small, there are some securities to be followed. Many such organizations are very much concerned with protecting confidentiality. This includes the protection of personnel data, marketing plans, formulas, and other techniques. Along with this, they have major concerns with integrity. Thus, a wide range of security policies is required in such an organization. It is difficult to define the meaning of security for any organization beforehand. Each organization is unique with its unique requirements. Which are difficult to be fulfilled by MAC and DAC controls.

DAC is an access control mechanism that allows system users to permit or prohibit other user's access to keep things under their control. The means of allowing or restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control).[7] DAC as the name implies, whether a user has access privilege to a particular topic is left to the discretion of an individual user. A DAC mechanism allows users to grant or revoke access to any object under their control without any interference from a system administrator. But the issue is, in most of the organizations the end-user who has access is not the owner of the particular information. The organization has ownership of most of the information.

MAC refers to a kind of access control during which OS constraints the power of a topic or initiator to access or generally perform some kind of operation on an object or target. In practice, a subject is usually a process or a thread; objects are constructs such as files, directories, TCP/UDP ports, shared memory segments, IO devices, etc. In MAC the security policy is centrally controlled by a security administrator, the user does have the ability to change or override the policy. This can only be done by the security administrator. This allows security administrators to define a central policy that's guaranteed (in principle) to be enforced for all users.

Role-Based Access Control (RBAC)

In organizations access control is taken very seriously, every user is assigned a task to complete and have many responsibilities. Access control decisions are often determined by the roles these individual users take on as a part of the organization.[1] Role-Based Access Control (RBAC) policy bases access control decisions based on what responsibilities are assigned to that particular user. Every organization has employees with a particular role, RBAC uses this concept. In RBAC every user is assigned a specific role and instead of giving permissions to a particular user, access is given to a role. In this way, every user having the same role has the same sets of permission. The user cannot pass access permissions on to the other user at their will. Thus, the disadvantage of

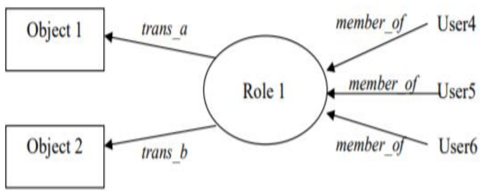


Figure 1. Role-Based Access Control[1]

Figure **1** can help us to understand the working of RBAC more clearly. As we can see above figure user4, user5, user6….userN are assigned to Role 1. Now the administration just has to take care of Role 1 decide which access is allowed for that role, which access is granted and which is rejected as we can see we have permission for transactions on Object 1 and Object 2 for Role 1.

MAC overcomes in RBAC. In this way every user has a role, a user at any particular time can have more than one role assigned. Since users aren't assigned permissions directly, but only assigned to them through their role (or roles), managing individual users is simple as we only have to assign a proper role to the user; this simplifies common operations, like adding a user or changing a user's department. RBAC is simple and easy to manage as many users having the same privilege can be managed by assigning them to a role.

There are three rules to RBAC:

1. A user can perform any task only when the user has been assigned a Role.
2. The Current Role of the user must be authorized as mentioned by Rule 1. This helps to make sure only valid Roles are assigned to users.
3. A user can perform a particular task only if its Role is assigned the permission to do so. This Rule along with the other two makes sure the user only uses the access given to its Role.

In addition to these roles can be combined in a hierarchy manner where the higher-level role has all the permissions that its sub-role has. RBAC is flexible access control technology which is flexible around to implement MAC or DAC

Open Policy Agent (OPA)

Open Policy Agent is an open-source policy engine that makes sure all the policies defined by the organization are followed. OPA is a new standard engine that came into the market in 2018 as Cloud Native Computing Foundation Sandbox which is an open-source, general-purpose policy engine that can be used to enable context-aware policy enforcement across the entire organization.[6] As we have seen that every organization has its own unique set of policies to follow. These policies are important for companies' long-term success. Often organizations can depend on normal methods for there security but those are prone to have few disadvantages or the policies are too complex to be implemented using those techniques. In OPA policy decisions are made by executing queries. OPA then evaluates policies and data to produce results for the requested query.

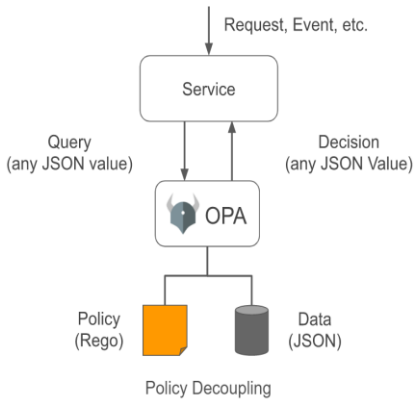


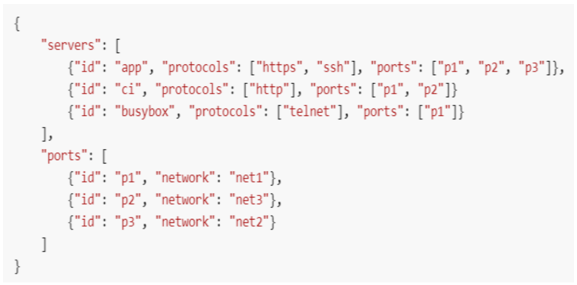
Figure 2. Working of OPA

OPA is a domain-agnostic which is helpful to execute any type of query.

- Which users can access which resources.
- Which subnets egress traffic is allowed to.
- Which clusters a workload must be deployed to.
- Which registries binaries can be downloaded from.
- Which OS capabilities a container can execute with.
- Which times of day the system can be accessed.

These policies are not limited to yes or no results they can give output as structured data like a .json file.

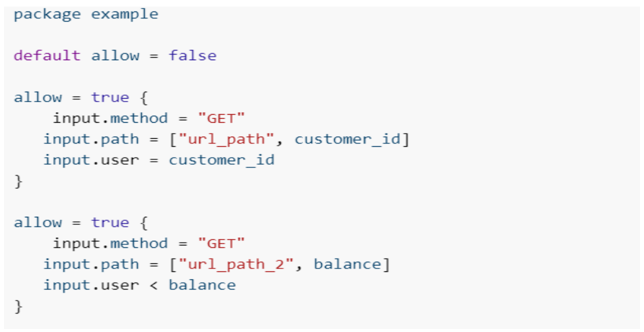
OPA is in more demand from the past two years as it has found its home at CNCF. OPA works perfectly along-side a Kubernetes cluster so this can be managed at many different levels as desired. OPA has a few basic components included within them. We have a data file, this is an optional file that can be in a JSON format. It stores all the variables that can be used to write the policies. This includes but is not limited to the type of user (role), routers, servers, ports, O.S, software's, etc. Below is a small example of the same.



```
{
    "servers": [
        {"id": "app", "protocols": ["https", "ssh"], "ports": ["p1", "p2", "p3"]},
        {"id": "ci", "protocols": ["http"], "ports": ["p1", "p2"]}
        {"id": "busybox", "protocols": ["telnet"], "ports": ["p1"]}
    ],
    "ports": [
        {"id": "p1", "network": "net1"},
        {"id": "p2", "network": "net3"},
        {"id": "p3", "network": "net2"}
    ]
}
```

Figure 3. Data.json

The second important file in OPA is writing the policies. OPA is written in a language called REGO. REGO is a high-level declarative language. As REGO is a declarative language it does not involve the extra code which makes it slow. It is fast and clean which executes the query in a short time. We will take an example and then understand it.

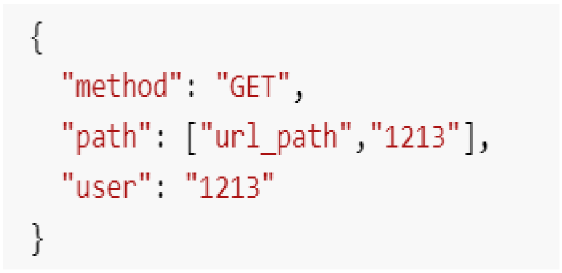


```
package example

default allow = false

allow = true {
    input.method = "GET"
   input.path = ["url_path", customer_id]
   input.user = customer_id
}

allow = true {
    input.method = "GET"
   input.path = ["url_path_2", balance]
   input.user < balance
}
```

Figure 4. Example of policy

A policy written in OPA looks likes the image above. We can import the data file as a package that can later be used in policies. When we write allow = true that means the specific decision would be allowed if the following statement is evaluated as true. If all the statement following are true only then access is given. Even if one of the statements results as false the access would be revoked for that set. If there are more than one condition set for a particular asset the statements within performs AND operation. So if all are TRUE only then access is granted, but for two different conditions for the same asset, we perform OR operation that is even if it's allowed in one condition access will be granted. We can get the inputs like method, path, user, id, etc from input.? Which is commonly used like input.method, input.path, etc.

Now once we are set, we can execute the query and get the result. This is the last component that is given by the user. A query is stored on a file that is fired or it can be directly executed. The result of the query decides whether we are allowed or not.



```
{
  "method": "GET",
  "path": ["url_path","1213"],
  "user": "1213"
}
```

Figure 5. Example of query file

COMPARISON

RBAC is settled in the market and used widely. It is basic and is perfect for security to some extent. RBAC only takes into the matter about the role of the user. This is the major drawback of RBAC. There can be a user having access for a particular asset and so it's in that ROLE but we have to make sure access of the user is revoked after working hours. Any organization has many such requirements so RBAC cannot satisfy all the needs. There is an upgrade of this access control Attribute Base Access Control (ABAC). This considers all the other attribute which are not considered by ABAC. On the other hand, OPA also solves all the major drawbacks. Unlike other access control management tools OPA does not tries to define but it gives the user a platform to create any time of policy that the organization can ever imagine. The language provided REGO is declarative so authors can focus on the results they should expect instead of worrying how to execute the query. If we compare OPA it can easily perform tasks of the RBAC model along with the concept of Separation of Duty (SOD) as in RBAC. SOD makes sure that certain combinations of access should never be assigned to a user at the same time. Like raising salary hike requests and also having the right to approve such kind of requests. Not only RBAC but OPA can also perform all the ABAC operations, you can do all the operations considering the attributes of the user.

PROPOSED FEATURE

As we learned OPA is an open-policy engine that enforces the rules given by the system on every user. OPA is a better replacement for access control management than RBAC or ABAC. We have seen how policy is written in OPA. We make a function which returns whether permission is granted or revoked. A simple OPA policy can be as:

```
allow = true {
 net := input.networks #assigns input.networks to net
 internal_net = net #checks if internal_net is equal to net
} #if all the statements are true allow will be true
```

A typical organization has many such policies written which are evaluated every time a query is executed in OPA. Here we would like to introduce a new feature in OPA called pre-evaluation which will have many benefits to the existing application.

Pre-evaluation

As we have found earlier RBAC is widely used among all the organization even is has a basic model. The reason is, it's based on Roles, even if we require a more secured system RBAC covers most of the basic security functionality. In OPA if we simply follow the basic rule used in RBAC with few modifications we can implement it in OPA. OPA at the start of the session, while login will evaluate the policy with only checking the known factors after login like user-id and roles assigned to it. If the user does not match any criteria those policies will be discarded for the session and only the remaining policy will be evaluated further on.

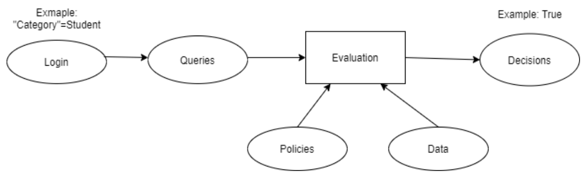


Figure 6. Pre-Evaluation at Login

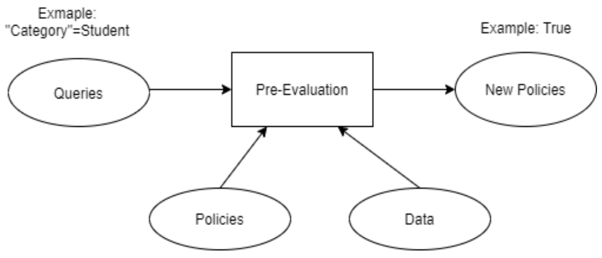
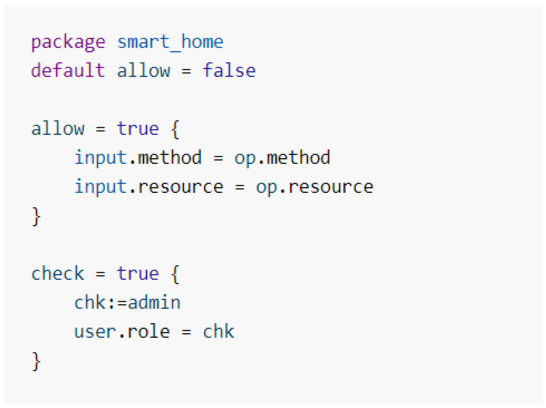


Figure 7. Evaluation based on new policy

By using pre-evaluating policies, OPA can compute all the policies once at login instead of evaluating all the policies every time.

In many cases, most of the policy depends upon which user is being logged in instead of the user inputs. Pre-evaluating the policies reduces the number of policies to be evaluated in the entire session. Evaluating policies results in a decision like yes/no or a list but pre-evaluation results in a new policy with a fewer number of polices in them. This results greatly in reduced time in evaluating the policies for the entire session of the user logged in.

We will discuss using the following example



```
package smart_home
default allow = false

allow = true {
    input.method = op.method
    input.resource = op.resource
}

check = true {
    chk:=admin
    user.role = chk
}
```

Figure 8. Example of policy

To use this policy, the user will pass the query and input.method and input.resource will be provided at run time. Along with that, there are few policies that do not requires user input, once the user is logged in we can check basic information about the user. In this case, if the logged-in user is admin this query will be kept on the other hand if a user logs in which is not admin the new policy created by pre-evaluating will not have a check policy as it would never be true. Once after login OPA will create a new policy file, OPA will then automatically check in this new policy file instead of the old policy.

CONCLUSION

Access Control Management is an important aspect in security of data. We have seen in this paper that there are various methods to implement access control in organization. Requirement for security changes from organization to organization. RBAC is perfect fit if there policy and rules are not strict. As we get more complex rules to be followed OPA is the best policy-engine to look for. Using a declarative language to write policy, OPA is quick and strong engine to ensure all the rules and policies are been followed. Even though OPA evaluates were quickly 10;s of 1000 of policy can be slow. Pre-evaluating the policy can decrease the number of policies to be searched at every request.

References

1. Role-Based Access Controls (15th National Computer Security Conference (1992) Baltimore, Oct 13-16, 1992. pp. 554 - 563)

2. https://www.openpolicyagent.org/docs/latest/

3. A.C. O'Connor & R.J. Loomis (March 2002). Economic Analysis of Role-Based Access Control (PDF). Research Triangle Institute. p. 145.

4. https://www.openpolicyagent.org/docs/latest/

5. Gilbert MD, Lynch N, Ferraiolo FD (1995). "An examination of federal and commercial access control policy needs". National Computer Security Conference, 1993 (16th) Proceedings: Information Systems Security: User Choices. DIANE Publishing. p. 107.

6. https://www.cncf.io/blog/2019/04/02/toc-votes-to-move-opa-into-cncf-incubator/

7. Trusted Computer Security Evaluation Criteria, DOD 5200.28-STD. Department of Defense, 1985.-