S3 Bucket Policy

AWS Policies are JSON-based documents that define permissions for what actions can be performed on AWS resources and under what conditions. They are a key element of AWS security model, allowing fine-grained control over access to various services and resources. Each policy consists of statements that specify which principals (IAM users or AWS services) can perform certain actions (e.g., `s3:GetObject`) on specific resources (e.g., an S3 bucket) under particular conditions.

There are two subcategories of AWS policies in particular, which are crucial for managing access control in AWS, though they both serve different purposes and scopes:

1. **S3 Bucket Policies:** These are attached directly to S3 buckets and define permissions for all objects within the bucket. Bucket policies are written in JSON and can specify access permissions based on IP address, AWS account, or other conditions.
2. **IAM Policies:** These are attached to IAM identities (users, groups, roles) and specify what actions they can perform on AWS resources and are also written in JSON but are more granular, allowing fine-tuned control over specific AWS services and resources. They define permissions based on actions, resources, and conditions.

Basically, S3 bucket policies control access to S3 buckets and their contents based on conditions like IP address or AWS account, while IAM policies control access related to actions, resources, and identities within an AWS account. A detailed example of a S3 bucket policy is presented in the next page, with IAM policies receiving a similar treatment in later sections.

S3 Bucket Policy Example and Breakdown:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::example-bucket/*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "203.0.113.0/24"
                }
            }
        }
    ]
}
```

Note: If you are not someone from a technical background, do not be scared by the curly braces, the JSON code you see above is extremely simple to understand and write.

Short for JavaScript Object Notation, JSON code is structured as a series of keys, (some of which may be nested within one another) with every key having a unique value associated with it. Let us examine the code one line at a time, to better understand this structure and the purpose of the code.

"Version": "2012-10-17"

This line specifies that the 2012-10-17 version of the AWS Policy Language is being used.

"Statement": [ ]

The core of our policy, this line defines the statement key, which acts as a container for any number of different individual permissions (often times also referred to as “Statements”). Consider each individual statement to be like a rule that dictates who can do what with your resources on AWS.

"Effect": "Allow"

The “Effect” field determines whether an action is allowed or denied. In our case, the specified value is "Allow", meaning that the policy grants permission to perform the mentioned action. If the specified value were to be "Deny" instead, then it would explicitly block access to the desired action.

"Principal": “*”

The “Principal” element specifies the users, accounts and/or services which the policy applies to. In our case, the asterisk (*), also known as the wildcard character, is set as the element value. This makes the policy applicable to all users, accounts and/or services (basically everyone). You could replace this with a specific AWS account or user if you wanted to limit access to said agent.

"Action": “s3:GetObject”

Used to specify the operations or API calls that are to be allowed or denied. Each AWS service has its own associated set of actions that describe the tasks that can be performed with said service. For example, the above statement deals with whether the policy allows the associated user to retrieve objects from the S3 Object Storage service.

"Resource": “arn:aws:s3:::example-bucket/*”

A self-explanatory field, the “Resource” element specifies the AWS resources (like an S3 bucket, EC2 instance, lambda function, etc) that the policy applies to. It allows us to ensure that the policy only applies to the objects we deem necessary, with the AWS resources being identified by their ARN (Amazon Resource Name).

For example, in our case, the policy applies to all objects within the S3 Bucket named `example-bucket` as specified by the ARN (*arn:aws:s3:::example-bucket/**)

"Condition": { }

An optional field, the “Condition” element lets us specify the circumstances in which the policy would be effective, allowing us to define additional constraints that must be met for the policy to apply. Mentioned conditions are written as a series of key-value pairs, for example in the mentioned snippet, the “IpAddress” condition key is used to limit the effects of the policy to requests coming from the very specific IP range of `192.0.2.0/24`, adding an extra layer of security to the S3 Buckets.

TLDR;

In our example policy, the following fields are used to:

Statement: Define the individual permissions.

- **Effect**: Specify that the purpose of the policy is to grant permission/"Allow" to perform specific actions.
- **Principal**: Make the policy apply universally to all users/accounts, through use of the wildcard character `"*"`.
- **Action**: `s3:GetObject` allows users to retrieve objects from the S3 bucket.
- **Resource**: Specifies the S3 bucket (`example-bucket`) and all objects inside (`/*`).
    - **Condition**: Limits access based on IP address. In this case, only users coming from the IP range `203.0.113.0/24` are allowed access.

Therefore, the purpose of the policy is to grant read-only access to all objects in the `example-bucket` , provided the users are accessing it from a specific IP range.