re:Invent Builder Session

Permission Boundaries

Greg McConnel

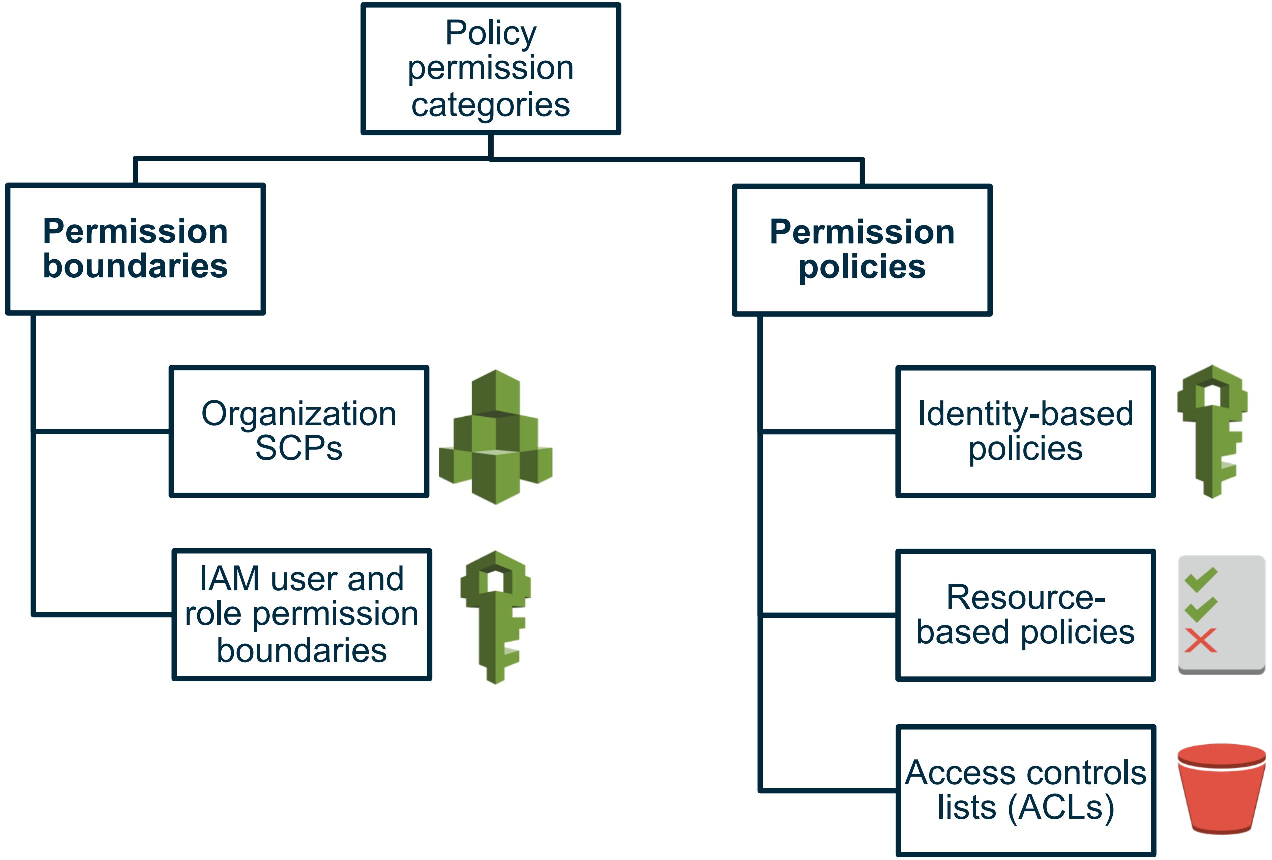
Solutions Architect
Amazon Web Services (AWS)

Agenda

- Policy categories
- Permission boundary basics
- Resource restrictions

https://awssecworkshops.com/builder-sessions/

Policy permission categories



Permission boundary basics

Before and After Permission Boundaries

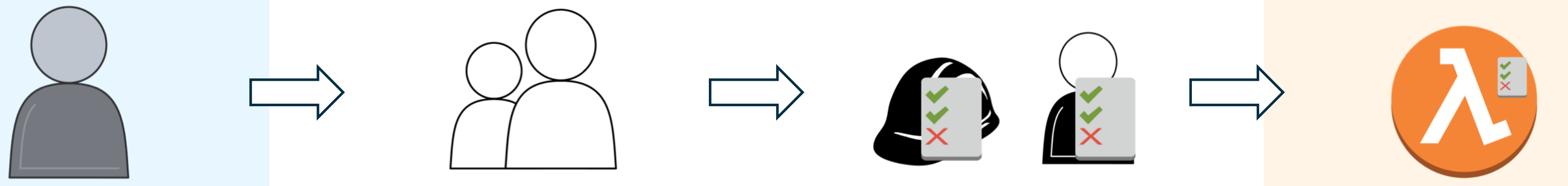
Before

- Certain IAM policy actions (e.g. PutUserPolicy, AttachRolePolicy) were essentially god-like permissions.
- Doing any form of self-service permissions management was non-trivial.

Now

- Administrator can grant previously god-like permissions, but specify a "permissions boundary."
- Allow developers to create principals for their applications and attach policies, but only within the boundary.

Permission Boundaries – mechanism



Admins

Create delegated admins

Admins create delegated admins with permissions such that users and roles created must have a permission boundary

Delegated admins

Create “bound” user & roles

Delegated admins create users and roles that have permission boundaries attached

“Bound” IAM users and roles

Permission boundary restricts the users and roles

Delegated admins attach the IAM roles to resources (or use the IAM user)

Restricted resources

Resource permissions restricted

Effective permissions of resources like Lambda functions are limited by permission boundary

A condition

```
"Condition": {"StringEquals":
    {"iam:PermissionsBoundary":
    "arn:aws:iam::ACCOUNT_ID:policy/permissionboundary"
    }
}
```

A condition applied to principal creation actions (users and roles)

```
"Effect": "Allow",
"Action": ["iam:CreateRole"],
"Resource": ["arn:aws:iam::ACCOUNT_ID:role/path/"],
"Condition": {"StringEquals":
    {"iam:PermissionsBoundary":
    "arn:aws:iam::ACCOUNT_ID:policy/permissionboundary"
    }
}
```

Mechanism

App developer creates role with delegated permissions

# Step 1: Create role

```
$ aws iam create-role –role-name MyTestAppRole
–assume-role-policy-document file://Role_Trust_Policy_Text.json
–permissions-boundary arn:aws:iam::<ACCOUNT_NUMBER>:policy/DynamoDB_Boundary_Frankfurt
```

# Step 2: Create policy

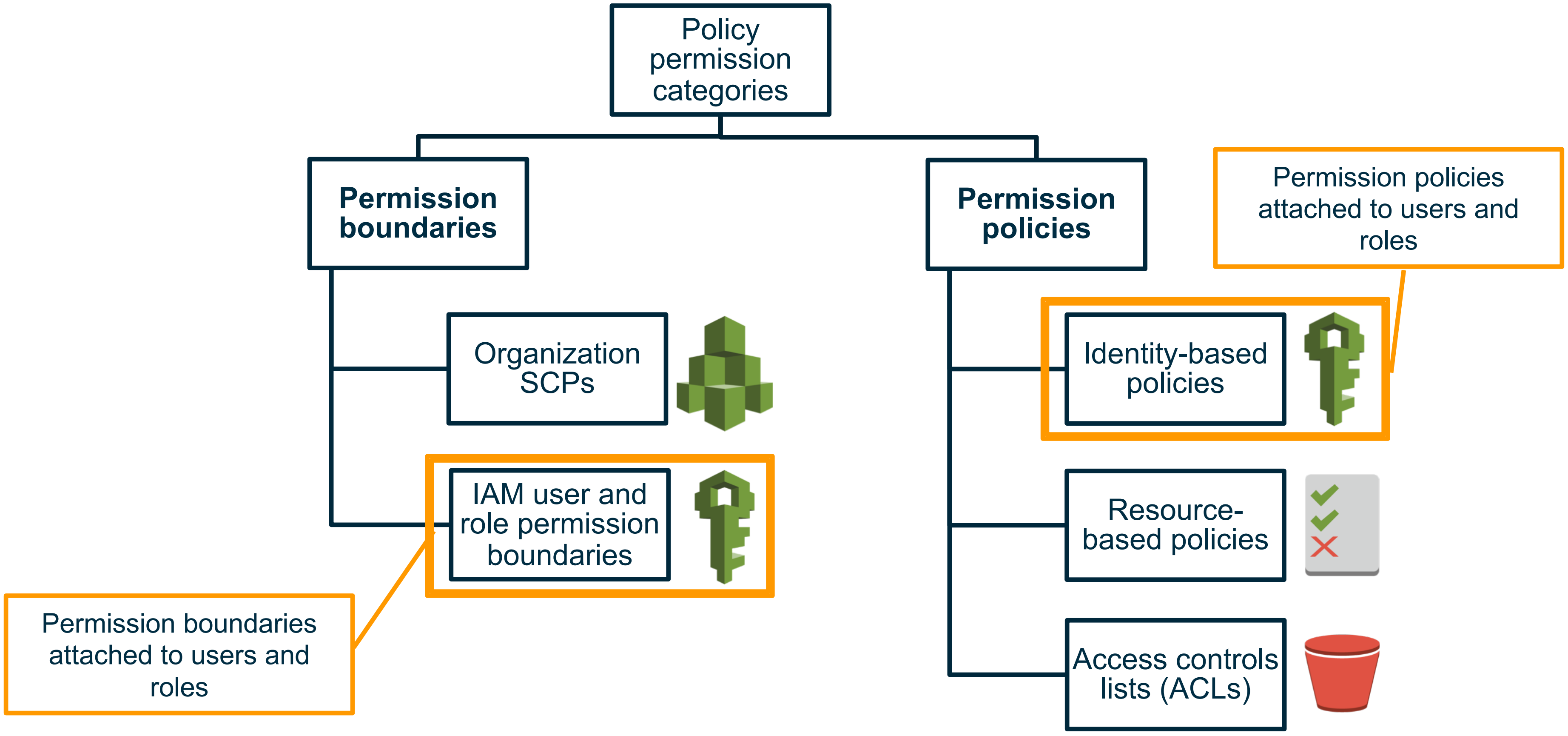
```
No change
```

# Step 3: Attach policy

```
No change
```

Permission boundary mechanisms

Policy permission categories


Policy permission categories
Permission boundaries
Permission policies
Permission policies attached to users and roles
Organization SCPs
IAM user and role permission boundaries
Identity-based policies
Resource-based policies
Access controls lists (ACLs)
Permission boundaries attached to users and roles

Everything after authentication

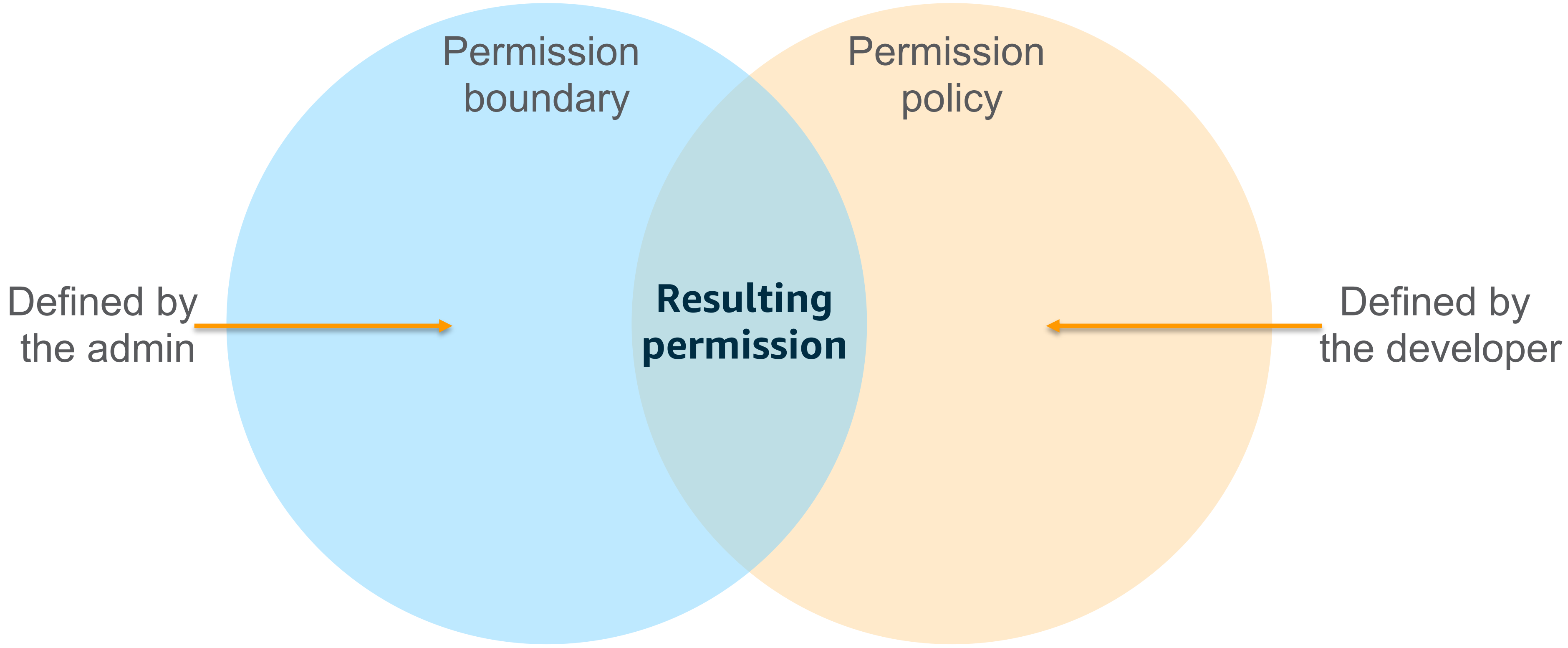
1. **Authenticate** the principal

2. Determine which **policies** apply to the request

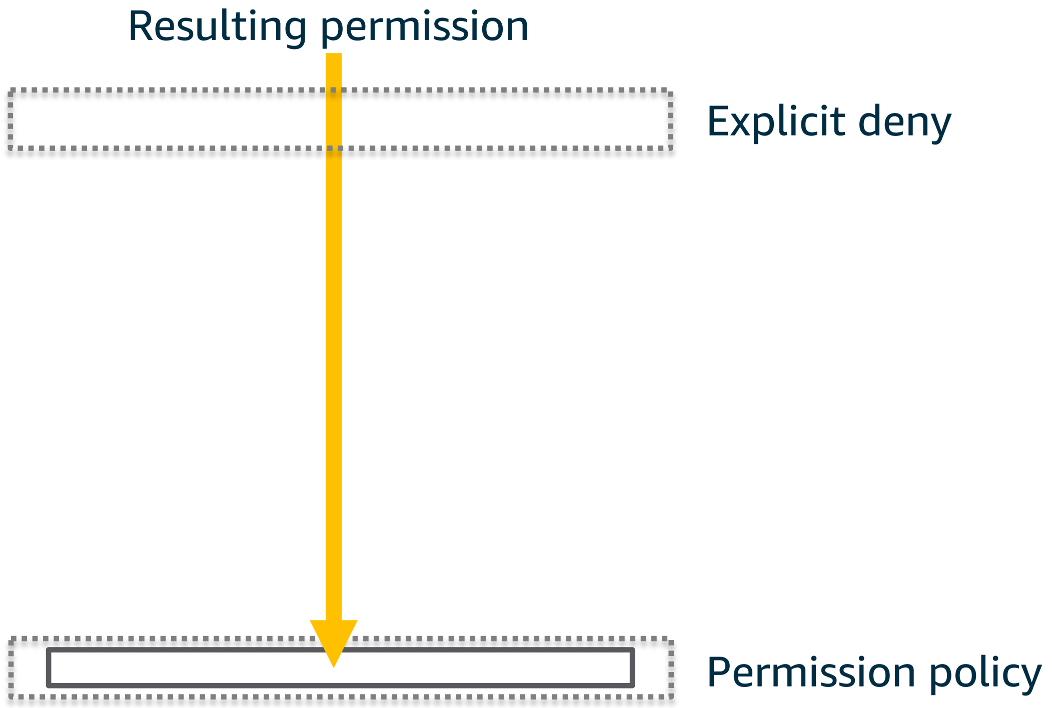
3. **Evaluate** the different policy types that apply which affect the order in which they are evaluated.

4. **Allow or Deny** the request

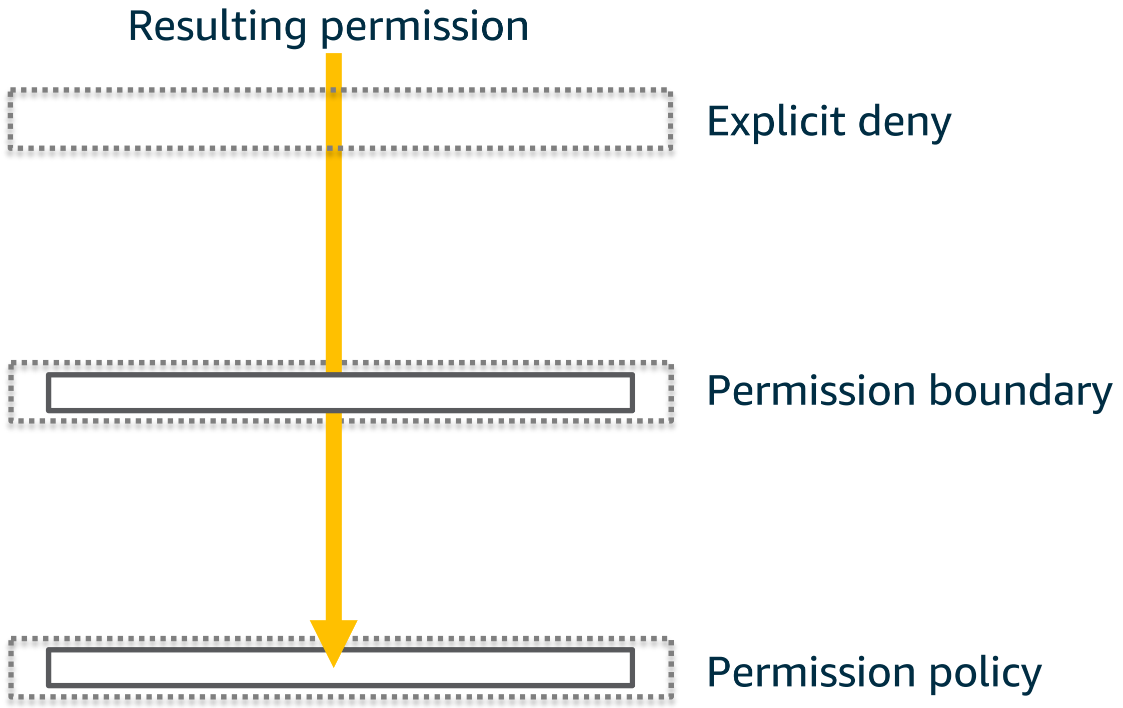
Effective Permissions - intersection



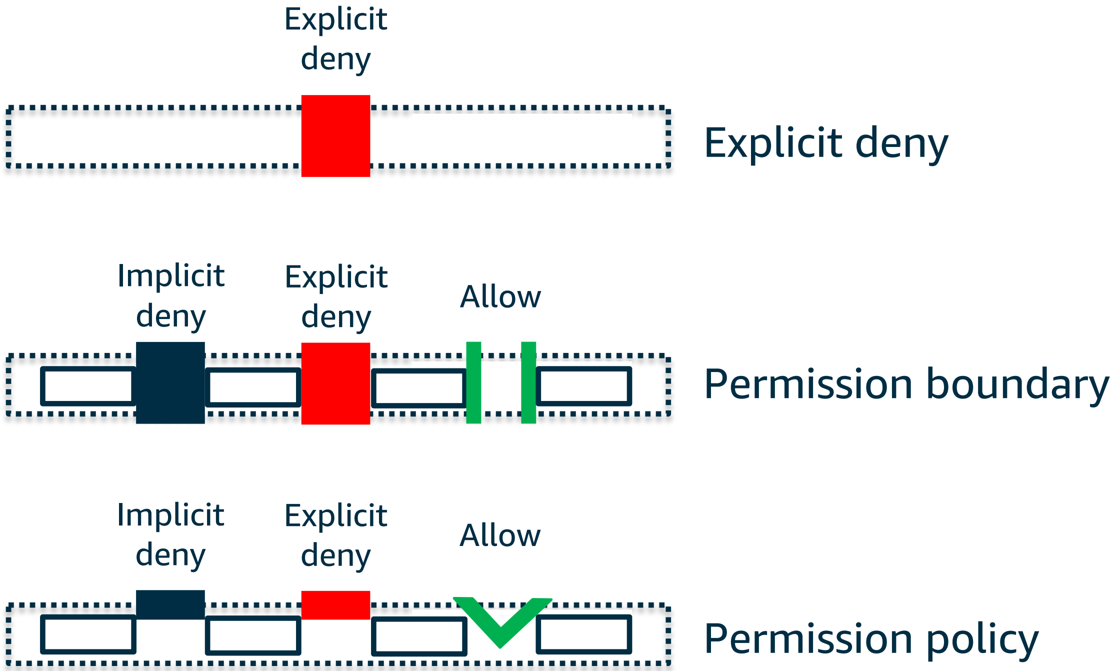
Effective permissions – mechanism



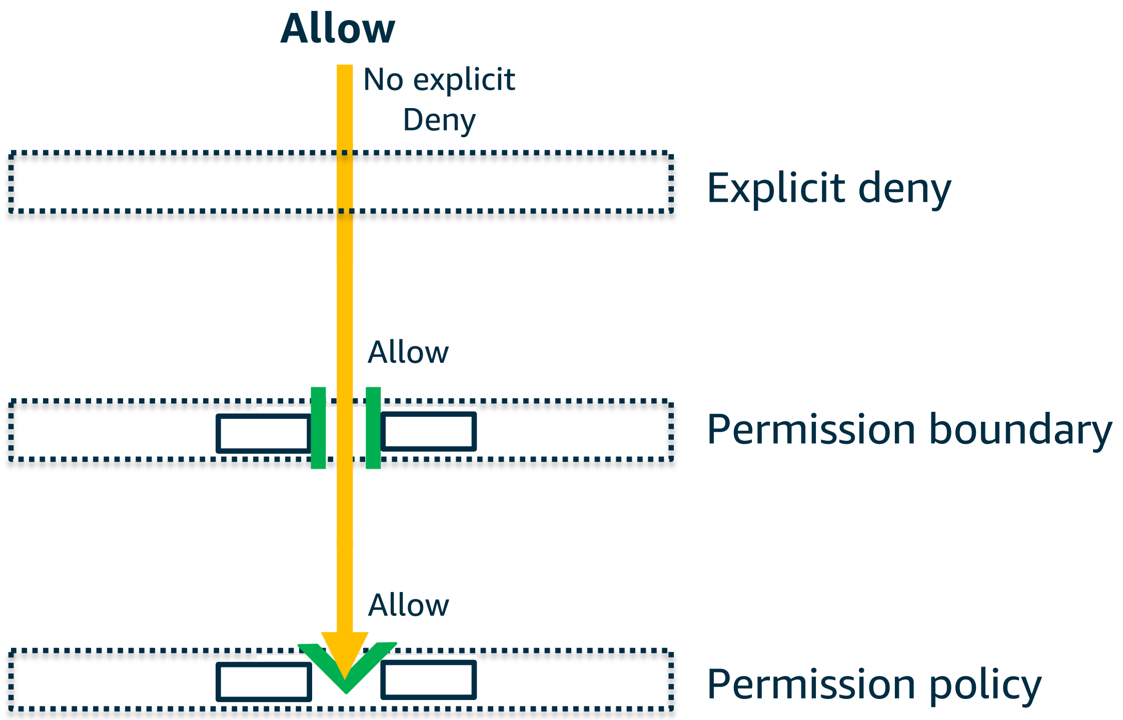
Effective permissions – mechanism



Effective permissions – mechanism



Effective permissions – mechanism



Effective permissions – scenario 1

Request: s3:GetObject / bucket name: example1

Permission Boundary

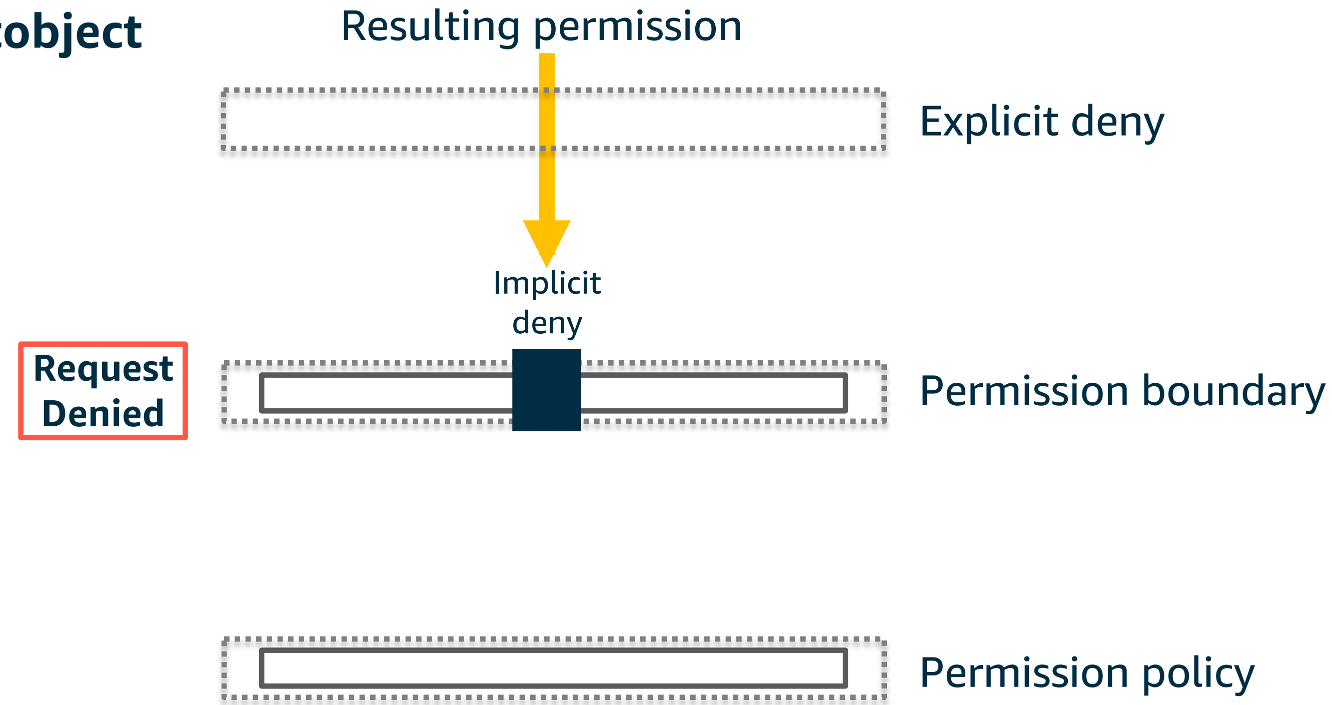
```
{
  "Version": "2012-10-17",
  "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "logs:CreateLogGroup",
         "logs:CreateLogStream”,
         "logs:PutLogEvents”
       ],
     "Resource": "arn:aws:logs:*:*:*"
     }
}
```

Permission Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "logs:CreateLogGroup",
         "logs:CreateLogStream",
         "logs:PutLogEvents",
         "s3:*"
        ],
      "Resource": "*"
      }
    ]
}
```

Effective permissions – scenario 1

Request: s3:getobject



Effective permissions – scenario 1

Request: s3:GetObject / bucket name: example1

Permission Boundary

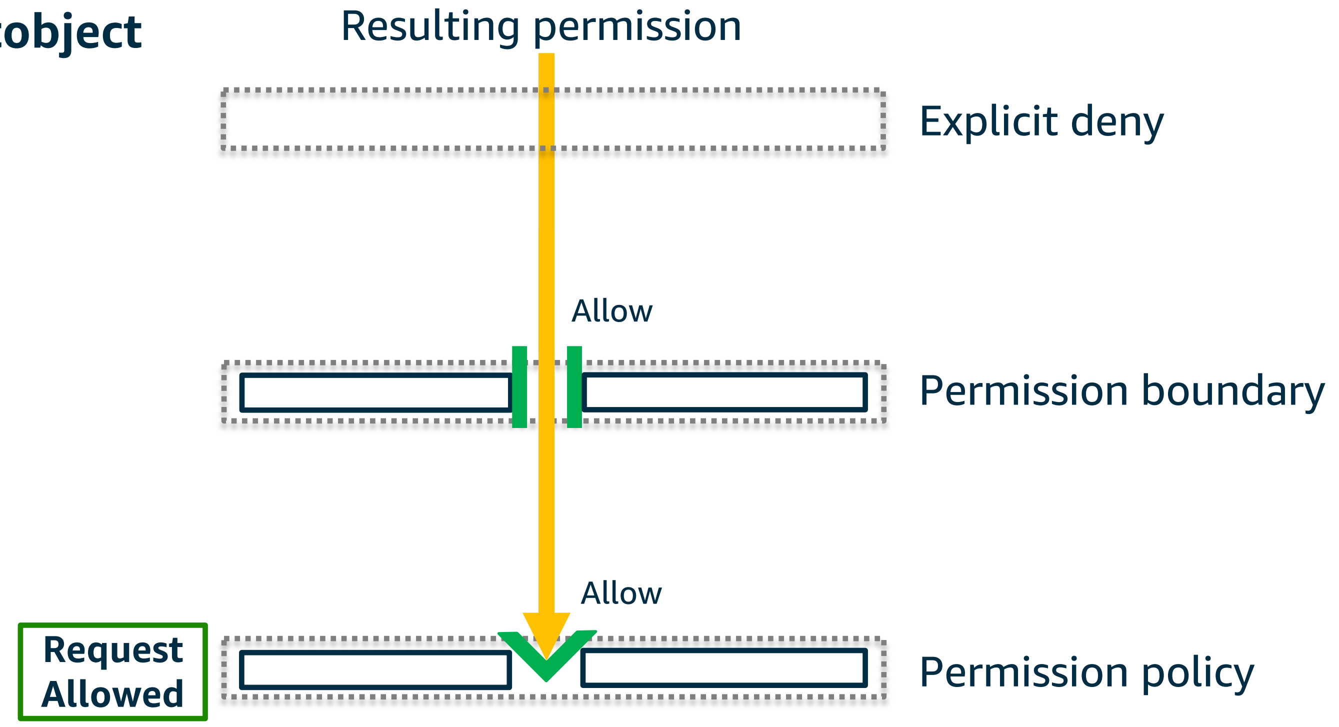
```
{
  "Version": "2012-10-17",
  "Statement": [
     {
       "Effect": "Allow",
       "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream”,
            "logs:PutLogEvents”
       ],
     "Resource": "arn:aws:logs:*:*:*"
     },
     {
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource”:"arn:aws:s3:::example1/*"
     }
}
```

Permission Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
       "Effect": "Allow",
       "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "s3:*"
        ],
      "Resource": "*"
     }
    ]
}
```

Effective permissions – scenario 1

Request: s3:getobject



Effective permissions – scenario 1

Request: s3:GetObject / bucket name: example1

Permission Boundary

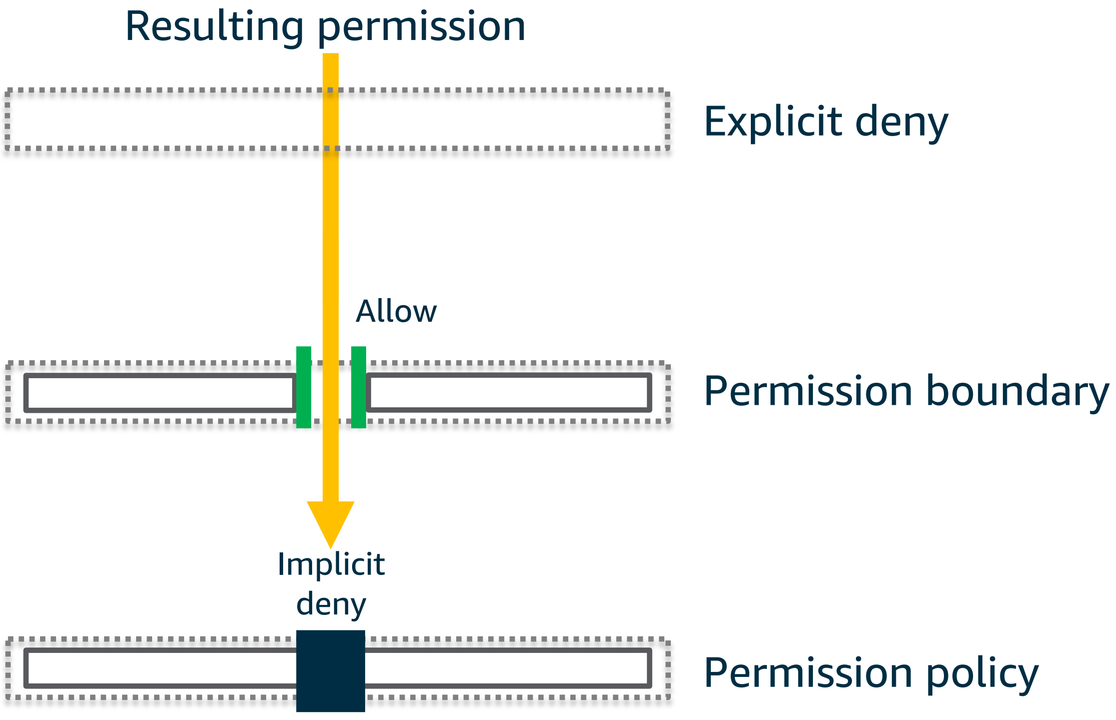
```
{
  "Version": "2012-10-17",
  "Statement": [
     {
       "Effect": "Allow",
       "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream”,
            "logs:PutLogEvents”
       ],
     "Resource": "arn:aws:logs:*:*:*"
     },
     {
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource”:"arn:aws:s3:::example1/*"
     }
}
```

Permission Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
       "Effect": "Allow",
       "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
         ],
      "Resource": "*"
      }
    ]
}
```

Effective permissions – scenario 1

Request: s3:getobject



Request Denied

Resource Restrictions

- Use of ARNs to specify individual resources in the policy
- Wild cards so that any names within that namespace can be used
- Can then use polices to restrict access based on name and/or path
- Primarily concerned with IAM roles, policies and users. Also could be useful for EC2 instances and Lambda functions

Resource Restrictions

- Not all actions support resource level permissions: https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-services-that-work-with-iam.html

- ARNs
    - arn:aws:iam::123456789012:role/example-path/*
    - arn:aws:iam::123456789012:policy/example-name*
    - arn:aws:iam::123456789012:user/example-path/example-name*

Resource Restrictions

- Goal: carving out a space for the delegated admins to be able to modify resources without impacting other resources. Yet they can still use other resources like AWS managed policies.

Resource Restrictions - policies

- Consider permissions assigned to a delegated admin to create policies
- If not restricted then delegated admins could modify existing customer managed policies.
- Complementary but not required for a permission boundary strategy

```
"Effect": "Allow",
"Action": [
      "iam:CreatePolicy",
      "iam:DeletePolicy",
      "iam:CreatePolicyVersion",
      "iam:DeletePolicyVersion",
      "iam:SetDefaultPolicyVersion"
      ],
"Resource": "*"
```

vs

```
"Effect": "Allow",
"Action": [
      "iam:CreatePolicy",
      "iam:DeletePolicy",
      "iam:CreatePolicyVersion",
      "iam:DeletePolicyVersion",
      "iam:SetDefaultPolicyVersion"
      ],
"Resource":
"arn:aws:iam::ACCOUNT_ID:policy/path/name*"
```

Resource Restrictions - roles

- Just like with policies we want to carve out a safe space for roles.
- Permission boundaries play a part here, but not all actions support the condition
- In addition different teams could be using the same permission boundaries

```
"Effect": "Allow",
"Action": [
      "iam:UpdateRole",
      "iam:DeleteRole”
      ],
"Resource": "*"
```

vs

```
"Effect": "Allow",
"Action": [
      "iam:UpdateRole",
      "iam:DeleteRole”
      ],
"Resource":
"arn:aws:iam::ACCOUNT_ID:role/path/name*"
```

Resource Restrictions - roles

- Here are the actions that the support the permission boundary condition:

  AttachRolePolicy
  AttachUserPolicy
  CreateRole
  CreateUser
  DeleteUserPermissionsBoundary
  DeleteUserPolicy
  DetachRolePolicy
  DetachUserPolicy
  PutRolePermissionsBoundary
  PutRolePolicy
  PutUserPermissionsBoundary

Resource Restrictions – other resources

- Where else would resource restrictions for a permission boundary strategy make sense?