Protecting your ABAC security model through SCP guardrails

Michael Chan
Sr. Solutions Architect
Identity Solutions

Jonathan VanKim
Sr. Solutions Architect
Security Specialist
Agenda

Overview of ABAC for AWS

High-level guidance

Enforcement examples and strategy

Questions
Overview of ABAC for AWS
ABAC for AWS

- Attribute-based access control (ABAC)
  - Authorization based on principal attributes and resource attributes
  - Attributes – part of the session
  - Attributes on resources are tags
Do you use ABAC?

Do you currently use it?

Why are you using it?

Do you want to use it?
What reasons should you consider ABAC?

• Single account but require fine-grained controls
  ▪ IAM pathing of roles/resources might be a better fit if the services support it

• Associate access with employee attributes

• Fine-grained control within a specific service or set of resources
High-level guidance for ABAC
High-level guidance

- ABAC complements RBAC rather than replaces it
- ABAC relies on mutable tags – key to success is tag governance
- Consider ABAC when a multi-account strategy along with RBAC does not meet your requirements
- Adopt ABAC on a resource-by-resource basis
- Consider human versus application scenarios – applications deployed using pipelines can have built-in guardrails
Implementing ABAC: How do we secure it?

What is a tag?

Tags are multipurpose attributes applied to principals and resources
What ABAC looks like
A quick ABAC refresher

A role's permissions policy, granting the **DeleteSecret** permission

```json

Admin
role

Identity
policy

A simple IAM policy statement

```json
{
  "Effect": "Allow",
  "Action": [ "secretsmanager:DeleteSecret" ],
  "Resource": [ "arn:aws:secretsmanager:*::*:secret:*" ],
  "Condition": { }
}
```
Role with a principal tag

Our role with a tag applied

tag key: is_admin
tag value: true
ABAC using a principal tag

This policy allows us to use our principal’s tag for authorization

```json
{
    "Sid": "AllowDeletionForAdmins",
    "Effect": "Allow",
    "Action": [ "secretsmanager:DeleteSecret" ],
    "Resource": [ "arn:aws:secretsmanager:*::*:secret:*" ],
    "Condition": {
        "StringEquals": {
            "aws:PrincipalTag/is_admin": "true"
        }
    }
}
```
What ABAC guardrails look like
Where do guardrails come in for tags?

- Tags are mutable

- Tags are used for many things besides authorization
Where do guardrails come in for tags?

- Tags are mutable
  - Tags used for ABAC need to be protected

- Tags are used for things besides authorization
  - Tags not used for ABAC need to be modifiable
What’s a service control policy (SCP)?

- Manages permissions across your organization
- Specifies maximum available permissions
- Controls maximum available permissions, centrally
- Does not grant permissions
Using SCPs: Protect our ABAC principal tags!

```json
{
    "Sid": "DenyModifyingAdminTag",
    "Effect": "Deny",
    "Action": [
        "iam:TagRole",
        "iam:TagUser",
        "iam:UntagRole",
        "iam:UntagUser"
    ],
    "Resource": "*",
    "Condition": {
        "StringNotEqualsIfExists": {
            "aws:PrincipalTag/is_admin": "true"
        },
        "Null": {
            "aws:RequestTag/is_admin": "false"
        },
        "StringNotLike": {
            "aws:PrincipalArn": "arn:aws:::iam::*:role/admin/iam/*"
        }
    }
}
```

Deny tagging operations if the principal is not tagged as an admin

Deny tagging if the tag to create/update/delete is the `is_admin` tag

Only an IAM admin is able to set the `is_admin` tag
Using SCPs: Enforce attribute-based access!

```json
{
    "Sid": "DenyDeletionForNonAdmins",
    "Effect": "Deny",
    "Action": [ "secretsmanager:DeleteSecret" ],
    "Resource": [ "arn:aws:secretsmanager:*:*:secret:*" ],
    "Condition": {
        "StringNotEqualsIfExists": {
            "aws:PrincipalTag/is_admin": "true"
        }
    }
}
```
Session tags: First, a role with a standard tag

User

Role A

IAM console

is_admin: true

IAM console

Role A

Summary

Creation date
April 22, 2020, 07:29 (UTC-08:00)

ARN
aws:iam::123456789012:role/roleA

Last activity
None

Maximum session duration
1 hour

Permissions

Tags (1)

Key
is_admin

Value
true
Session tags: Overrides existing tags

Call **AssumeRole**, and set:
1) tag `is_admin:false`

Set `is_admin:true`
Transitive tags: Persist across role assumptions

User

Call AssumeRole, and set:
1) tag is_admin:false
2) set tag = transitive

Role A

AssumeRole (no tags explicitly passed)

Role B

is_admin:false

is_admin:false
Principal tag guardrails: Summary

Use SCPs to

- Ensure **only authorized users can assign** an ABAC tag
- Ensure **only those authorized can pass** transitive ABAC tags
- **Protect the tags** once assigned
- **Enforce the permissions** that use tags

Use transitive session tags to

- **Pass unmodifiable tags** to roles
Using resource tags with ABAC

Resource tags can be used with principal tags

Role: team:green

Secret: team:green
ABAC with resource tags

Granting access to resources if the principal has the same tag

```json
{
    "Sid": "AllowGetSecretWithMatchingTags",
    "Effect": "Allow",
    "Action": [ "secretsmanager:GetSecret" ],
    "Resource": [ "arn:aws:secretsmanager:*:*:secret:*" ],
    "Condition": {
        "StringEquals": {
            "aws:PrincipalTag/team": "${aws:ResourceTag/team}"}
    }
}
```
ABAC guardrails for resource tags

Guardrails to consider implementing

1. **During resource creation**, resources must have an ABAC tag applied (tag-on-create)

2. **During resource creation**, the provided ABAC tag key must be the same case as the principal’s
ABAC guardrails for resource tags

Guardrails to consider implementing

1. **During resource creation**, resources must have an ABAC tag applied (tag-on-create)
2. **During resource creation**, the provided ABAC tag key must be the same case as the principal’s
3. **After resource creation**, the ABAC tag cannot be modified
4. **After resource creation**, the ABAC tag cannot be deleted
ABAC guardrails for resource tags

Guardrails to consider implementing

1. **During resource creation**, resources must have an ABAC tag applied (tag-on-create)
2. **During resource creation**, the provided ABAC tag key must be the same case as the principal’s
3. **After resource creation**, the ABAC tag cannot be modified
4. **After resource creation**, the ABAC tag cannot be deleted
5. A principal **cannot modify any tags** on resources they didn’t create
6. A principal **cannot delete any tags** on resources they didn’t create
7. A principal **cannot do tagging operations** if its principal tag doesn’t exist
ABAC guardrails visual summary

Where you need to secure

1. AWS IAM Identity Center (formerly AWS SSO)
2. External identity provider
3. Permissions policy
4. Secret
ABAC guardrails summary

If you use principal tags or resource tags for ABAC, remember to

1. Ensure directories are passing ABAC tags securely to principals (roles)
2. Ensure only authorized users can assign an ABAC tag to principals
3. Enforce the permissions that use tags
4. Protect the tags on resources once assigned

Note: All guardrail policy samples are available in our gist repo:

Additional guidance

- Be mindful of SCP space – implement only needed guardrails
- Implement ABAC only for resources that need it
- Test your policies
- *AWS services that work with IAM* page lists high-level ABAC support

Learn in-demand AWS Cloud skills

**AWS Skill Builder**

Access **500+ free** digital courses and Learning Plans

Explore resources with a variety of skill levels and **16+** languages to meet your learning needs

Deepen your skills with digital learning on demand

Train now

---

**AWS Certifications**

Earn an industry-recognized credential

Receive Foundational, Associate, Professional, and Specialty certifications

Join the **AWS Certified community** and get exclusive benefits

Access **new** exam guides
Thank you!