

The Power of AWS Tags

Two-person approval and scalable ABAC on AWS

Who are we?



Yoav Yanilov
Solutions Architect
Axiom Security



Itamar Bareket
CTO
MobiMatter

Agenda

1. realize ABAC is broken
2. fix ABAC
3. profit

Agenda

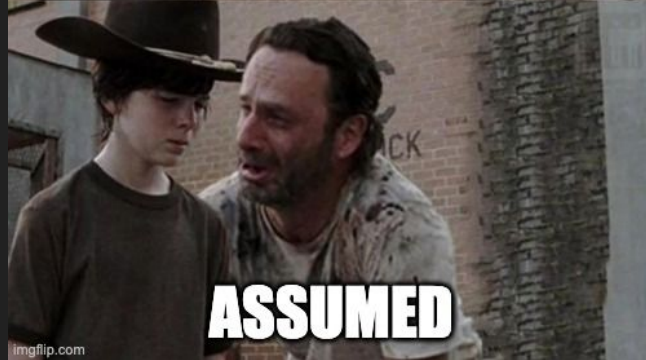
(for real)

Who can delete an RDS cluster in your organization?

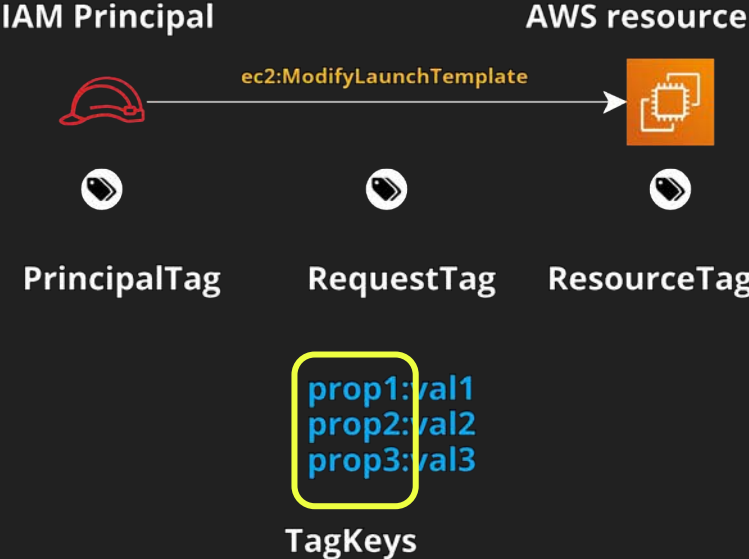
Seriously now, who?

Oh, An admin user.

How are you securing this user?



IAM condition keys refresher

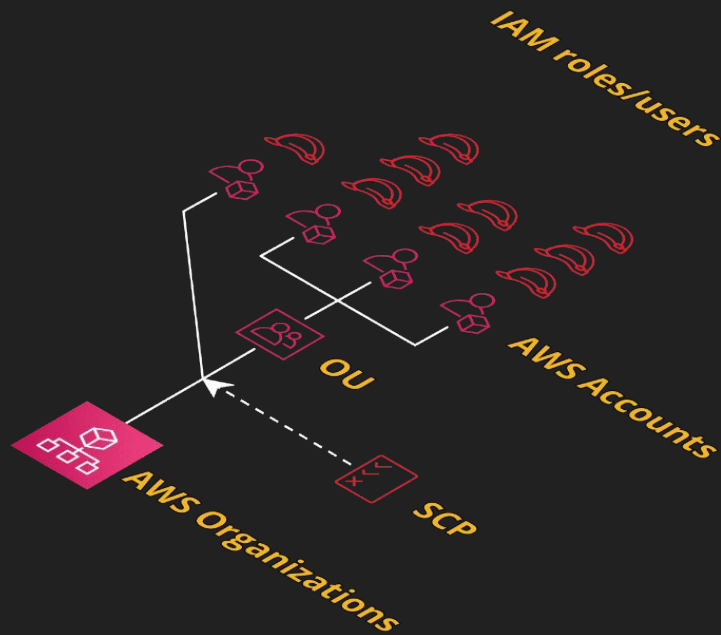


First things first

Enforcing AWS permissions at the Account-level

Centralized enforcement - SCP

- Just an IAM policy.
- Attached to AWS account(s).
- Applies to every User/Role
In that account.



Let's talk about ABAC

(tag-based access control)

Just use ABAC!

What could go wrong?

```
# Protect-Rds-Delete SCP
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyUnlessAdmin",
      "Effect": "Deny",
      "Action": "rds:Delete*",
      "Resource": "*",
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:PrincipalTag/is_admin": "true"
        }
      }
    }
  ]
}
```

“Who will guard the guards?”

– Juvenalis, Satire VI, circa 110 AD

```
# Protect-Rds-Delete SCP
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyUnlessAdmin",
      "Effect": "Deny",
      "Action": "rds:Delete*",
      "Resource": "*",
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:PrincipalTag/is_admin": "true"
        }
      }
    },
    {
      "Sid": "DenyTaggingAdminTrueUnlessAdminTrue",
      "Effect": "Deny",
      "Action": ["iam:Tag*", "iam:Untag*"],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/is_admin": "true"
        },
        "StringNotEqualsIfExists": {
          "aws:PrincipalTag/is_admin": "true"
        }
      }
    }
  ]
}
```

“Who will scale the.. uh.. Scallions?”

– Yoav, Conf42, circa now

```
# Protect-Rds-Delete SCP
```

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "DenyUnlessAdmin",  
      "Effect": "Deny",  
      "Action": "rds:Delete*",  
      "Resource": "*",  
      "Condition": {  
        "StringNotEqualsIfExists": {  
          "aws:PrincipalTag/is_admin": "true"  
        }  
      }  
    },  
    {  
      "Sid": "DenyTaggingAdminTrueUnlessAdminTrue",  
      "Effect": "Deny",  
      "Action": ["iam:Tag*", "iam:Untag*"],  
      "Resource": "*",  
      "Condition": {  
        "StringEquals": {  
          "aws:RequestTag/is_admin": "true"  
        },  
        "StringNotEqualsIfExists": {  
          "aws:PrincipalTag/is_admin": "true"  
        }  
      }  
    }  
  ]  
}
```

Single Responsibility

```
# Protect-Rds-Delete SCP
```

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "DenyUnlessAdmin",  
      "Effect": "Deny",  
      "Action": "rds:Delete*",  
      "Resource": "*",  
      "Condition": {  
        "StringNotEqualsIfExists": {  
          "aws:PrincipalType": "user",  
          "aws:PrincipalTag/is_admin": "true"  
        }  
      }  
    },  
    {  
      "Sid": "DenyTaggingAdminTrueUnlessAdminTrue",  
      "Effect": "Deny",  
      "Action": ["iam:Tag*", "iam:Untag*"],  
      "Resource": "*",  
      "Condition": {  
        "StringEquals": {  
          "aws:PrincipalType": "admin",  
          "aws:PrincipalTag/is_admin": "true"  
        }  
      },  
      "StringNotEqualsIfExists": {  
        "aws:PrincipalTag/is_admin": "true"  
      }  
    }  
  ]  
}
```

Data Plane

(distributed, specific)

Control Plane

(centralized, generic)

Separation of concerns!

```
# Protect-Rds-Delete SCP
```

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "DenyUnlessAdmin",  
      "Effect": "Deny",  
      "Action": "rds:Delete*",  
      "Resource": "*",  
      "Condition": {  
        "StringNotEqualsIfExists": {  
          "aws:PrincipalTag/is_admin": "true"  
        }  
      }  
    },  
    {  
      "Sid": "DenyTaggingAdminTrueUnlessAdminTrue",  
      "Effect": "Deny",  
      "Action": ["iam:Tag*", "iam:Untag*"],  
      "Resource": "*",  
      "Condition": {  
        "StringNotEqualsIfExists": {  
          "aws:RequestTag/is_admin": "true"  
        },  
        "StringNotEqualsIfExists": {  
          "aws:PrincipalTag/is_admin": "true"  
        }  
      }  
    }  
  ]  
}
```

Access control logic

Tagging Integrity logic

How do we scale
tagging integrity?

Path-based approach to integrity scaling

Alice is **granted** write access to `/home/alice`

Her **grant area contains** subfolders, e.g.

`/home/alice`

`/home/alice/git`

`/home/alice/memes`

But **doesn't contain**:

`/home/bob`

`/home/bob/git`

`/system`

Translated into IAM:

Alice's **grant area** is `/home/alice`,

She **can** add or remove any tag key matching

`/home/alice/*`

Chicken and Egg: granting access

All **grants** are stored in the `/meta/grant_path` file.

No one can write to `/meta/grant_path`

```
$ ls -l /meta/grant_path
-r--r--r--
$ cat /meta/grant_path
alice:/home/alice
bob:/home/bob
```

Translated into IAM:

No centralized grants file – Each IAM principal is tagged with its own grant.

The tag key `/meta/grant_path` denotes the principal's entry in **grant path** "file".

This tag's value acts as a "pointer", and defines the allowed **grant area**.

No one has a grant to add or remove tags under `/meta/*`

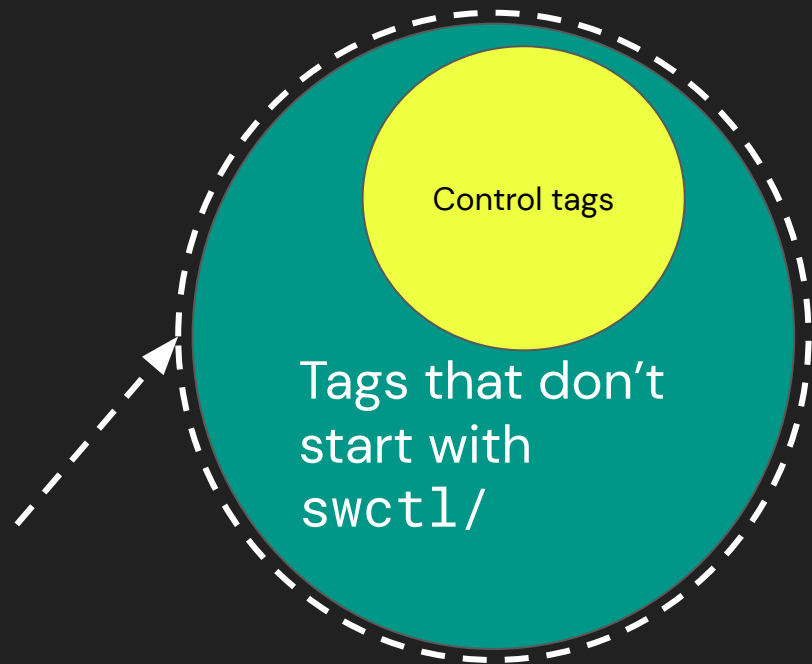
Introducing Control Tags

A scalable tag-based control plane for tagging operations

Definition:

Control tags
start with
swctl/

All Tags



```
[
{
  "Sid": "CtlTaggingWithoutGrantPath",
  "Effect": "Deny",
  "Action": "*",
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringLike": {
      "aws:TagKeys": "swctl/*"
    }
  },
  "Null": {
    "aws:PrincipalTag/swctl/v1/meta/grant_path": "true"
  }
}
},
{
  "Sid": "CtlTaggingOutsideGrantArea",
  "Action": "*",
  "Effect": "Deny",
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringLike": {
      "aws:TagKeys": "swctl/*"
    },
    "ForAnyValue:StringNotLike": {
      "aws:TagKeys": [
        "${aws:PrincipalTag/swctl/v1/meta/grant_path}",
        "${aws:PrincipalTag/swctl/v1/meta/grant_path}/*",
        "team",
        "role",
        "environment",
        "info/*"
      ]
    }
  }
}
}
]
```

designated prefix

Ensuring grant path exists when tagging with control-tags

```
    "aws:TagKeys": [
      "${aws:PrincipalTag/swctl/v1/meta/grant_path}",
      "${aws:PrincipalTag/swctl/v1/meta/grant_path}/*",
      "team",
      "role",
      "environment",
      "info/*"
    ]
  }
}
}
```

Well-known (legacy) tags
Extensible info (future) tags

Ensuring grant path
“permits” the tags when tagging with control-tags

Scaling ABAC with control tags – A recipe

1. Attach the ControlTags SCP to the target account.
2. Create an SCP (e.g. Protect-Rds-Delete) that relies on control tags.
3. Attach your SCP to the target account.

Protect-Rds-Delete SCP



Data plane

ControlTags SCP



Control plane



Revising the bad example

```
# Protect-Rds-Delete SCP
```

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "DenyUnlessAdmin",  
      "Effect": "Deny",  
      "Action": "rds:Delete*",  
      "Resource": "*",  
      "Condition": {  
        "StringNotEqualsIfExists": {  
          "aws:PrincipalTag/swctl/v1/admin/rds_deleter": "true"  
        }  
      }  
    }  
  ]  
}
```

Requires a grant of either:

```
swctl/v1/admin/rds_deleter  
swctl/v1/admin  
swctl/v1  
swctl/
```

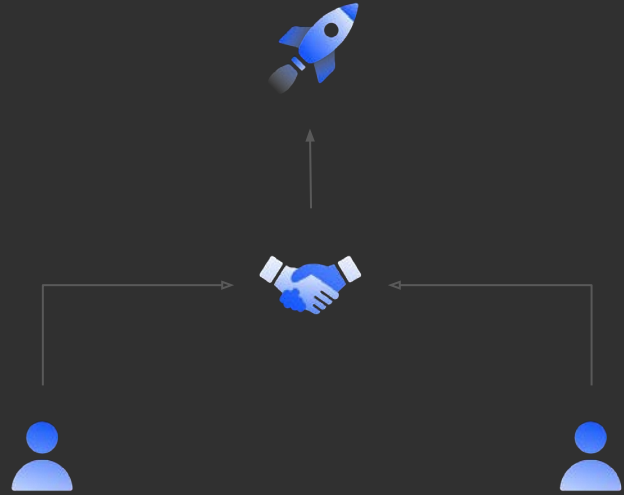
...but can we make it more secure?

What if only **two**
collaborating admins could
delete RDS resources?

Approvals in the cloud

Requiring the consent of two people in order to complete a process.

a.k.a 2PA



Approvals in the cloud

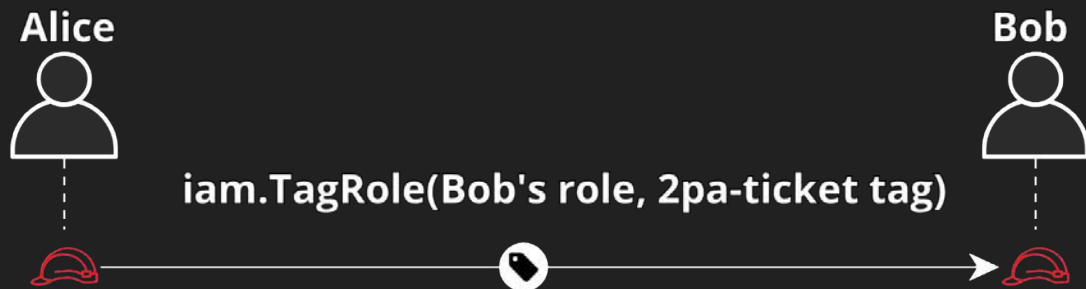
Use case #1

Preventing specific actions
Without approval



Demo Time!

What just happened



Anatomy of a 2PA ticket

Must be an admin to tag

swctl/v1/admin/2pa/ticket

=

by/alice/SOME_PAYLOAD/for/bob

Both giver and receiver must have
an `aws:SourceIdentity`

Payload

The 2PA Control Plane SCP

(hold your breath and count to 3)

2PA SCP (pt.1) – identity integrity

Limit who can set source identity to trusted parties, e.g.

- OIDC/SAML providers
- Principals acting as an identity brokers

Limit 2PA to principals with identity

```
{
  "Sid": "UnauthorizedSetIdentity",
  "Effect": "Deny",
  "Action": "sts:SetSourceIdentity",
  "Resource": "*",
  "Condition": {
    "StringNotEqualsIfExists": {
      "aws:PrincipalTag/swctl/v1/meta/identity_broker": "true"
    }
  }
},
{
  "Sid": "ApprovalWithoutIdentity",
  "Effect": "Deny",
  "Action": "iam:Tag*",
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringLike": {
      "aws:TagKeys": "swctl/v1/admin/2pa/*"
    },
    "Null": {
      "aws:SourceIdentity": "true"
    }
  }
}
}
```

2PA SCP (pt.2) – anti-reflexive integrity

Limit the “receiver section” of the tag’s value to **anything but** the current principal’s identity.

```
{
  "Sid": "ApprovingForSelf",
  "Effect": "Deny",
  "Action": [
    "iam:TagUser",
    "iam:TagRole",
    "iam:CreateUser",
    "iam:CreateRole"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "aws:RequestTag/swctl/v1/admin/2pa/ticket": [
        "*/for/${aws:SourceIdentity}"
      ]
    }
  }
}
```


2PA SCP (pt.3) – anti-forgery integrity

Limit the “giver section” of the tag’s value to **just** the current principal’s identity.

```
{
  "Sid": "ApprovingOnBehalfOfAnother",
  "Effect": "Deny",
  "Action": [
    "iam:TagUser",
    "iam:TagRole",
    "iam:CreateUser",
    "iam:CreateRole"
  ],
  "Resource": "*",
  "Condition": {
    "Null": {
      "aws:RequestTag/swctl/v1/admin/2pa/ticket": [
        "false"
      ]
    },
    "StringNotLike": {
      "aws:RequestTag/swctl/v1/admin/2pa/ticket": [
        "by/${aws:SourceIdentity}/*"
      ]
    }
  }
}
```

2PA ticket summary

Key guarantees:

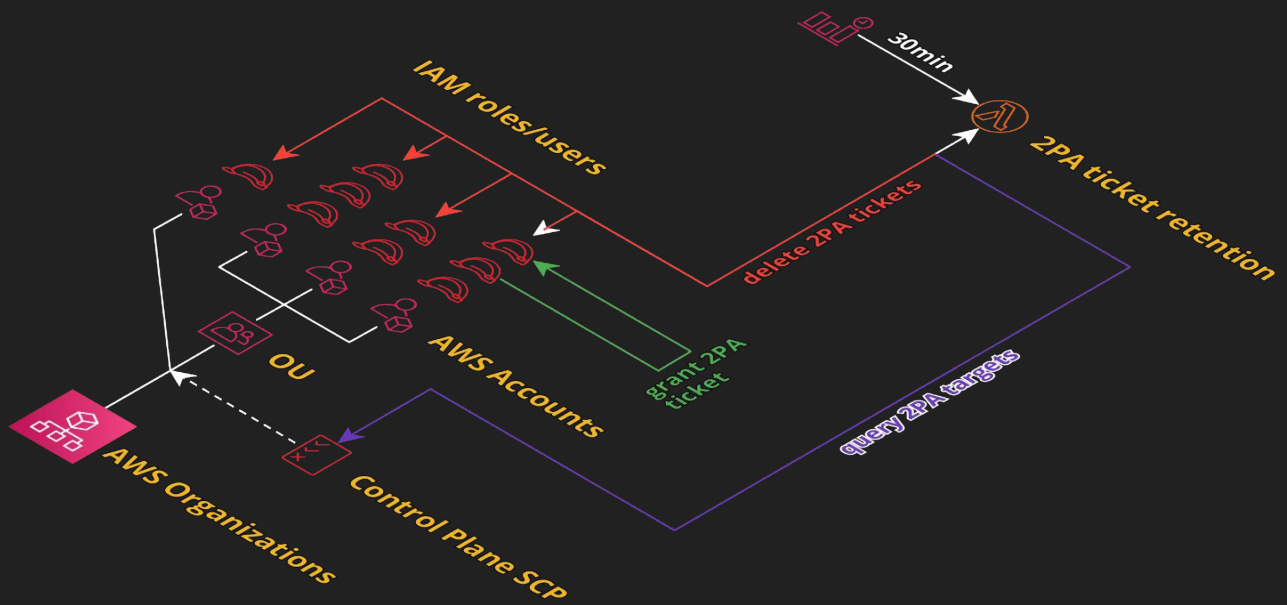
- Admin-only feature (swctl/v1/admin prefix)

Value guarantees:

- Identity integrity: only trusted parties may issue identities
- Anti-reflexive: can approve anyone **but** self
- Anti-forgery: can approve **only** on behalf of self
- Payload-bearing: specifying TTL for an external ticket retention process

2PA ticket retention

(turns out it can't all be JSON)



Data plane

defining Guarded Action SCP



2PA-guarded actions SCP

Eureka! Now NO ONE can delete an s3 bucket or an RDS instance/cluster when acting alone.



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GuardedActionWithoutApproval",
      "Effect": "Deny",
      "Action": [
        "s3:DeleteBucket",
        "rds:DeleteDBInstance",
        "rds:DeleteDBCluster"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotLikeIfExists": {
          "aws:PrincipalTag/swctl/v1/admin/2pa/ticket": [
            "*/for/${aws:SourceIdentity}"
          ]
        }
      }
    }
  ]
}
```

Escape deletion guard
using an approval ticket!

 The cherry on top 
(which also solves the chicken and the egg)

Escaping Control Tags, using 2PA Tags!

Use 2PA to escape the hierarchical integrity set by the control tags themselves.

Note: The initiating principal must be permitted to a superpath of a ticket (e.g. `swctl/v1/admin`)

```
{
  "Sid": "CtlTaggingOutsideGrantArea",
  "Effect": "Deny",
  "Action": "*",
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringLike": {
      "aws:TagKeys": [
        "swctl/*"
      ]
    },
    "ForAnyValue:StringNotLike": {
      "aws:TagKeys": [
        "team",
        "role",
        "environment",
        "info/*",
        "${aws:PrincipalTag/swctl/v1/meta/grant_path}",
        "${aws:PrincipalTag/swctl/v1/meta/grant_path}/?*")
      ]
    },
    "StringNotLikeIfExists": {
      "aws:PrincipalTag/swctl/v1/admin/2pa/ticket": [
        "*/for/${aws:SourceIdentity}"
      ]
    }
  }
}
```

Escape meta grant limitation
using an approval ticket!

Approvals in the cloud

Use case #2

Preventing specific actions

On specific resources

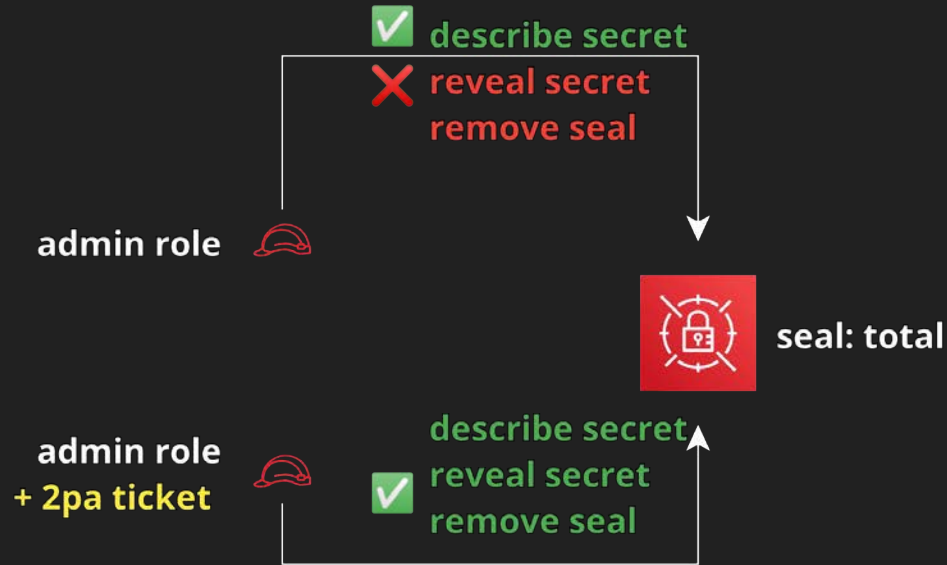
Without approval



2PA Seals 🐻🐻

2PA Seals 

2PA seals – workflow



Reflecting on 2PA seals

By sealing a resource we “freeze” specific functionality

- Sealing an AWS Secrets Manager secret to any action:
useful for storing manual recovery keys
- Sealing trust-relationship policy on IAM role:
Useful against privilege escalation

Exploring

Trust-relationship seals

2PA-Seal (pt. 1) – seal bypass

Only a principal carrying a 2pa ticket may execute

`iam:UpdateAssumeRolePolicy`

when the 2pa seal is set to
`deny_trust_update`

```
{
  "Sid": "BypassSealwithout2PATicket",
  "Effect": "Deny",
  "Action": "iam:UpdateAssumeRolePolicy",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:ResourceTag/swctl/v1/admin/2pa/seal": [
        "deny_trust_update"
      ]
    },
    "StringNotLikeIfExists": {
      "aws:PrincipalTag/swctl/v1/admin/2pa/ticket": [
        "*/for/${aws:SourceIdentity}"
      ]
    }
  }
}
```

2PA-Seal (pt. 2) – seal life-cycle

Only a principal carrying a 2pa ticket may change the seal tag.

Only a principal with an identity can attempt to bypass the seal

```
{
  "Sid": "ChangeSealWithout2PATicket",
  "Action": "*",
  "Effect": "Deny",
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringEquals": {
      "aws:TagKeys": "swctl/v1/admin/2pa/seal"
    },
    "StringNotLikeIfExists": {
      "aws:PrincipalTag/swctl/v1/admin/2pa/ticket": [
        "*/for/${aws:SourceIdentity}"
      ]
    }
  }
},
{
  "Sid": "BypassSealwithoutIdentity",
  "Effect": "Deny",
  "Action": "iam:UpdateAssumeRolePolicy",
  "Resource": "*",
  "Condition": {
    "Null": {
      "aws:SourceIdentity": "true"
    },
    "StringEquals": {
      "aws:ResourceTag/swctl/v1/admin/2pa/seal": [
        "deny_trust_update"
      ]
    }
  }
}
}
```


Fret not



The 2PA

Trust relay pattern

1. Deny assume-role without approval
2. Deny changes to trust policy without approval

```
data "aws_iam_policy_document" "trust" {
  statement {
    effect = "Allow"
    actions = ["sts:AssumeRole"]
    principals {
      type = "AWS"
      identifiers = ["arn:aws:iam::12345678:role/admin"]
    }
  }
  statement {
    effect = "Deny"
    actions = ["*"]
    principals {
      type = "AWS"
      identifiers = ["*"]
    }
    condition {
      test = "StringNotLikeIfExists"
      variable = "aws:PrincipalTag/swctl/v1/admin/2pa/ticket"
      values = ["*/for/${aws:SourceIdentity}"]
    }
  }
}

resource "aws_iam_role" "superadmin" {
  name = "super_admin"
  assume_role_policy = data.aws_iam_policy_document.json
  tags = {
    "swctl/v1/2pa/seal" = "deny_trust_update"
  }
}
```

Extending 2PA to AWS EKS

Goal: login to EKS Kubernetes clusters with a highly-privileged user only when two admins are collaborating

- Map the `super_admin` IAM role to a user with `system:masters` permissions in the EKS cluster's `aws-auth` ConfigMap.
- Apply a 2PA Seal tag on the `super_admin` IAM role, to prevent any single IAM principal from assuming it.

2pa trust relay pattern

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - rolearn: arn:aws:iam::12345678:role/super_admin
      username: superadmin
      groups:
        - system:masters
    - rolearn: arn:aws:iam::12345678:role/admin
      username: admin
      groups:
        - myorg_rnd
```

```
resource "aws_iam_role" "superadmin" {
  name = "super_admin"
  assume_role_policy = #require 2pa ticket to assume
  tags = {
    "swctl/v1/2pa/seal" = "deny_trust_update"
  }
}
```

A game of cat and mouse

```
s3.GetObject  
s3.PutBucketPolicy
```

Denied by bucket policy

```
iam.AssumeRole  
iam.PassRole
```

Denied by trust policy

```
iam.PutAssumeRolePolicy
```

Denied by 2pa seal

```
kubectl exec  
kubectl edit cm/aws-auth  
kubectl create rolebinding  
kubectl create clusterrolebinding
```

Unauthorized

```
iam.AssumeRole(superadmin) -> kubectl
```

Unauthorized by 2pa in trust policy

```
iam.create_saml_provider  
-> iam.PutAssumeRolePolicy(admin)  
-> create 2 users mapped to admin role  
-> have user1 approve user2  
-> have user2 assume superadmin
```

Denied by identity integrity in 2PA SCP

mgmt account



Control plane
SCP

member account



EKS Cluster



EKS Pod

TARGET: access
sensitive data

sensitive bucket
2pa ticket guard



sensitive role
2pa seal



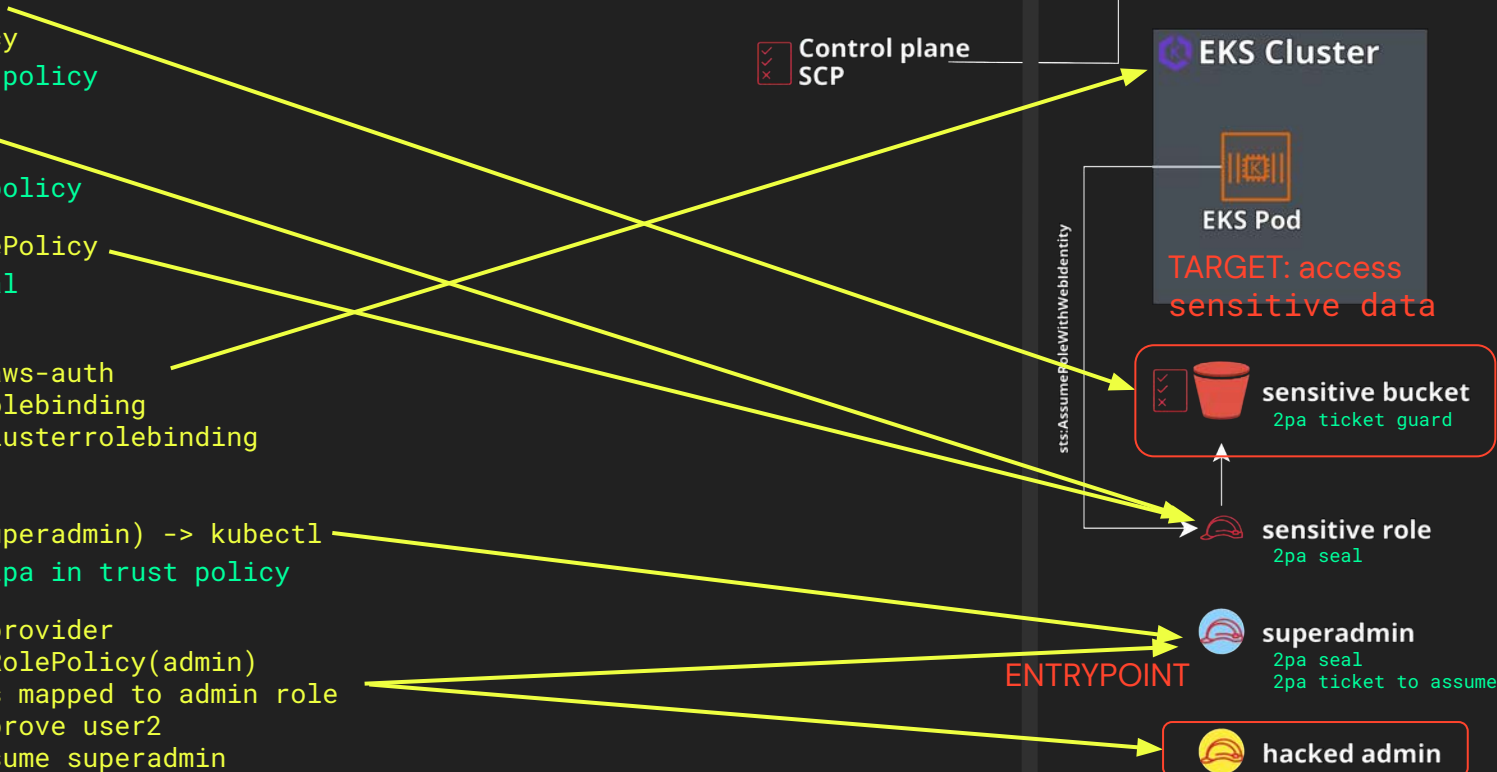
superadmin
2pa seal
2pa ticket to assume

ENTRYPOINT



hacked admin

sts:AssumeRoleWithWebIdentity



Always has been.

Wait, it's all control tags?



Thank you!
Questions?

Yoav



Itamar

