AWS Roles and Policies

Steve Loughran

January 2018

Concepts

- **AWS account:** Root a/c. Unlimited rights. Can be granted access to other account's S3 data via ACLs
- **IAM account:** Non-root account. login as (ID, pass) or (ID, secret)
- **Federated Account:** a/c from external service, authenticated via SAML (OKTA, AD, ...)
- **ARN: Amazon Resource Name:** UUID of things in AWS (buckets, tables...)
- **Role:** Identity an IAM account, federated account or service can adopt
- **Service Role**: role of a service (EC2, AWS Lambda)
- **Principal**: entity within policies
- **Group**: Not a principal, just a way of organizing policies
- **Credentials**: secrets used to sign AWS API requests

Relationships

- Federated users are always assigned a role
- EC2 VMs are always deployed with a role (credentials served up over HTTP)
- IAM Users have `adoptRole(role, permissions)` call;
  credentials valid for 1-15 min
- IAM Users can be in Groups. This is for policy admin: they are not principals
- Role ARNs can be used in policies for all AWS services

Roles provide a principal for federated users, and a way of restricting access to different services; useful for testing, possibly delegation

HORTONWORKS

AWS Policies

- JSON Policy language
- Common to all AWS services
- Identity-Based Policies: User, groups, roles all have policies
- Resource-Based Policies: policy on AWS resource/service (S3, AWS Lambda, KMS) (not: DynamoDB)
- AWS standard policies "S3 Access", "Dynamo DB Access"
- Limit: 10KB per policy (some resources only take 2KB)
- Limit: 10 "managed" policies per: user, group and role.
- Policy variables: limited. e.g. `${aws:userid}` `${aws.username}` (these two do not resolve for roles/federated users)

HORTONWORKS

Read all bucket, write under one directory

```
{ "Version" : "2012-10-17",
"Statement" : [
{
  "Sid" : "4",
  "Effect" : "Allow",
  "Action" : [ "s3:ListBucket", "s3:GetObject", "s3:GetObjectTagging" ],
  "Resource" : "arn:aws:s3:::hwdev-steve-ireland-new/*"
}, {
  "Sid" : "5",
  "Effect" : "Allow",
  "Action" : [ "s3:*" ],
  "Resource" :
  [ "arn:aws:s3:::hwdev-steve-ireland-new/test/testRestrictedRenameSrc/*",
    "arn:aws:s3:::hwdev-steve-ireland-new/test/testRestrictedRenameSrc",
    "arn:aws:s3:::hwdev-steve-ireland-new/test/testRestrictedRenameSrc/" ]
} ] }
```


HORTONWORKS

S3A support for mixed permissions

- HADOOP-15141 Support IAM Assumed roles in S3A (done!)
- HADOOP-15176 Enhance IAM assumed role support in S3A client (mostly done)
  - Handle lack of write up the directory tree
  - Don't worry if empty directory markers can't be created after delete, rename
  - Add API for creating basic policies (mostly for testing)
  - Tests via assumed roles

Mapping Ranger to AWS policies

- Don't have room to play with
- Variables may permit /home/${aws.user} rules, but not in roles (including EC2)
- Testing "fun"

Space and general rules language are the troublespots

HORTONWORKS

Statements: (effect, action+, principal*, resource+, condition*)

HORTONWORKS

Actions: verbs to be managed

HORTONWORKS

Resource: ARN patterns to which actions will be applied

HORTONWORKS

Conditions: predicates, can act on HTTP headers

```
{ "Sid": "RequireEncryptionHeaderOnPut",
  "Effect": "Deny",
  "Principal": "*",
  "Action": [ "s3:PutObject" ],
  "Resource": "arn:aws:s3:::BUCKET/*",
  "Condition": { "Null": { "s3:x-amz-server-side-encryption": true } }
 },
{
  "Sid": "RequireKMSEncryptionOnPut",
  "Effect": "Deny",
  "Principal": "*",
  "Action": [ "s3:PutObject" ],
  "Resource": "arn:aws:s3:::BUCKET/*",
  "Condition": {
   "StringNotEquals": { "s3:x-amz-server-side-encryption": "SSE-KMS" } }
}
```


HORTONWORKS

Resolution

1. All policies of resource, IAM user, role, groups user/role member of aggregated
2. All "Deny" rules checked first
3. *Any* explicit deny matches (action, resource, principal, conditions)→ fail
4. All Allow rules checked next
5. Any explicit allow rules matches (action, resource, principal, conditions) → pass
6. No match → fail

order independent; deny wins

you cannot explicitly deny an action in a parent path and then allow it underneath

HORTONWORKS

Mixing permissions down the tree.

- New to S3A this week
- Trouble: mock directory marker addition after: `delete()`, `rename()`
- Trouble: mock directory marker deletion
- Trouble: `rename(read-only, writeable)`
- Doomed: Having write access up the tree but not to a path underneath

These are where the attempts to sustain the "its a filesystem" metaphor break

HORTONWORKS

S3Guard: High performance & consistent metadata for S3

- What we plan to test Hive with
- Uses DynamoDB for IO
- Tables are updated after successful writes
- Not yet tested with restricted permissions

For a user to have access to part of the S3Guard table, it needs access to it all.

HORTONWORKS


Questions?
HORTONWORKS
POWERING THE FUTURE OF DATA