Overview

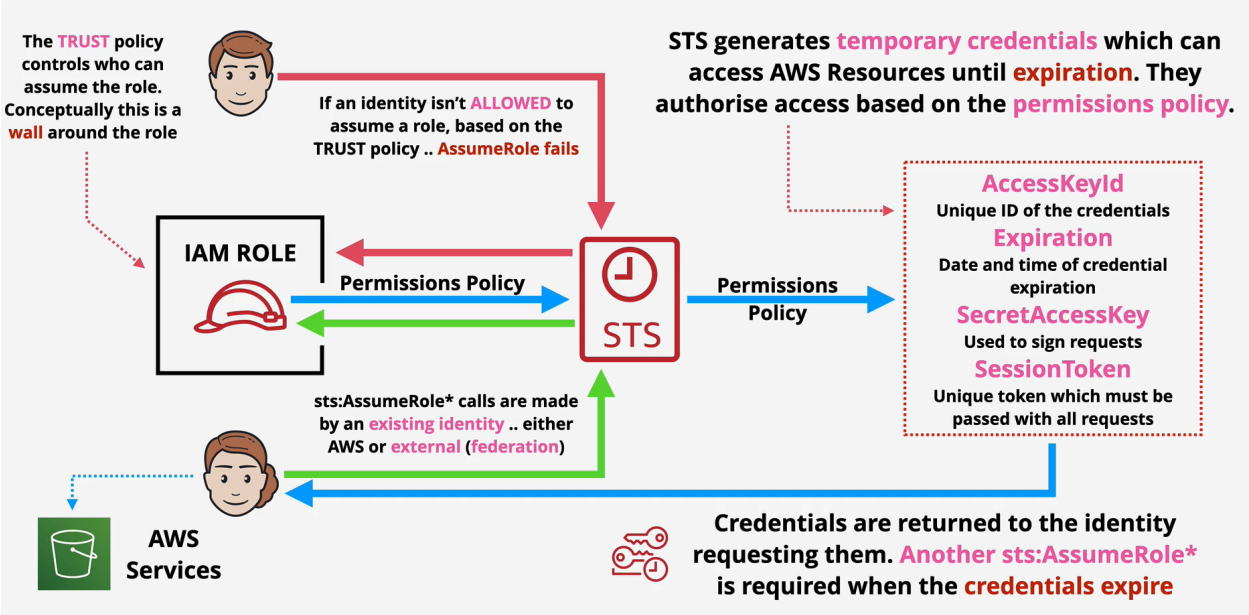
1. Generates **temporary credentials** when you use **sts:AssumeRole**

2. Similar to access keys

3. They expired and don't belong to the identity

4. Requested by an Identity (AWS or External)

STS Workflow



Step 1: Bob and Julie are trying to assume a Role. So, they send STS:AssumeRole to STS Service

Step 2: First STS will check the TRUST POLICY in the IAM Role to verify the given identity has allowed permission to assume the role or not.

Step 3: In the above case, Bob is not allowed to assume the role based on the Trust Policy. So, Assume Role fails. However, Julie is allowed to assume the role based on Trust Policy. So, she can assume Role.

Step 4: Once the Identity is allowed then STS will assume the Permission Policy from the given Role to check what are actions that given Identity can perform on the specific resource

Step 5: Based on the permission policy the STS will generate temporary credentials which can access AWS Resources until expiration. They authorize actions based on the permission policy.

Step 6: Credentials are returned to the identity requesting them. Another STS:AssumeRole is required when the credentials expire

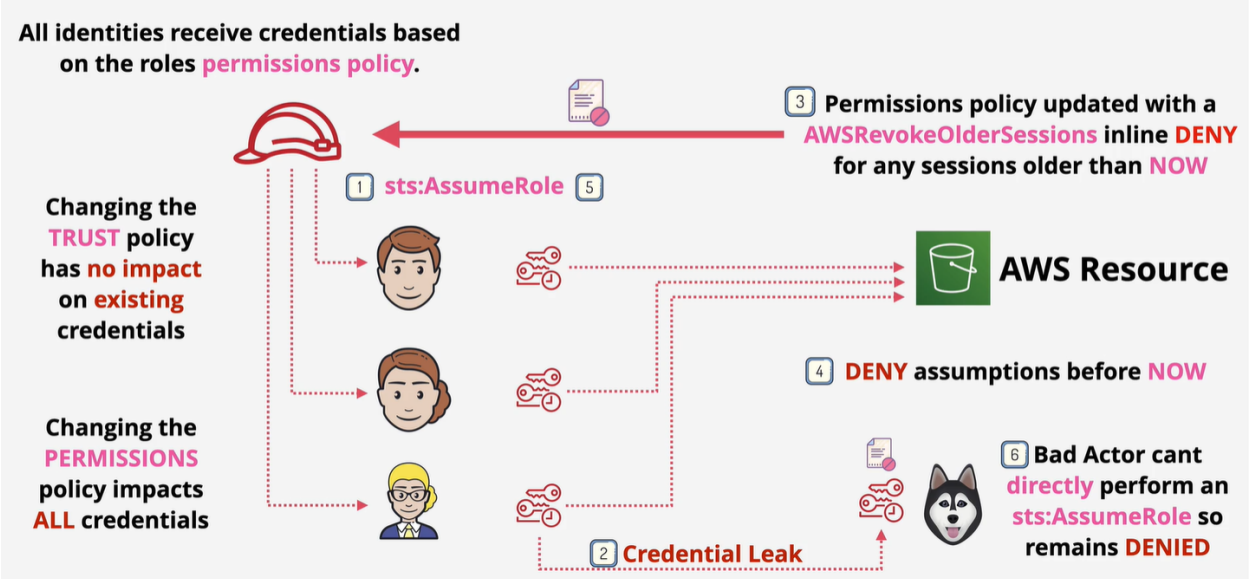
Revoking Temporary Credentials

Roles can be assumed by many identities

Who all get the same permissions

You can NOT manually invalidate temporary credentials, it will only get expired

So, let's see we have the following 3 users and Bob, Julile, and Emy (the one with glass). So, all 3 have assumed a role and accidentally Emy committed her temporary credentials in github. And that data is consumed by a bad guy woofy. Now, with the temporary credentials he can access the AWS Resources. How can you implement a design that will not affect the other users access, but deny the woofy user?



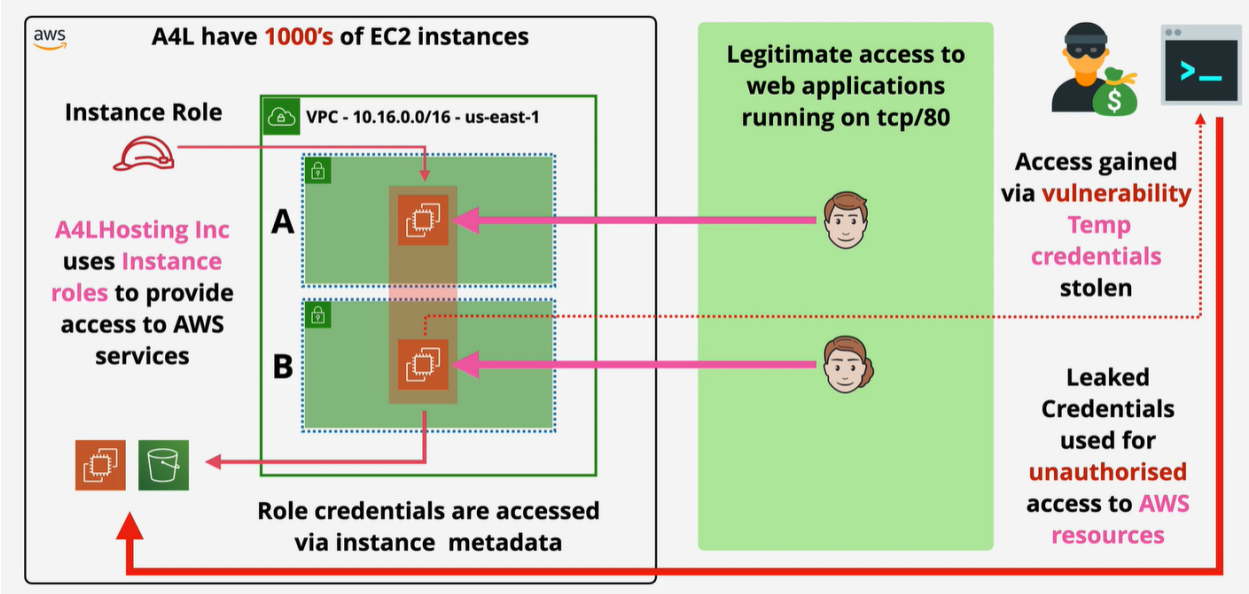
Step 1: Changing Trust Policy won't make any difference since woofy already has the temporary security credentials. So, he doesn't need to get validated again with Trust Policy to get the temporary credentials

Step 2: Changing Permissions Policy seems to be solving an issue. However, it will impact all the other existing users. We don't want that based on the given scenario

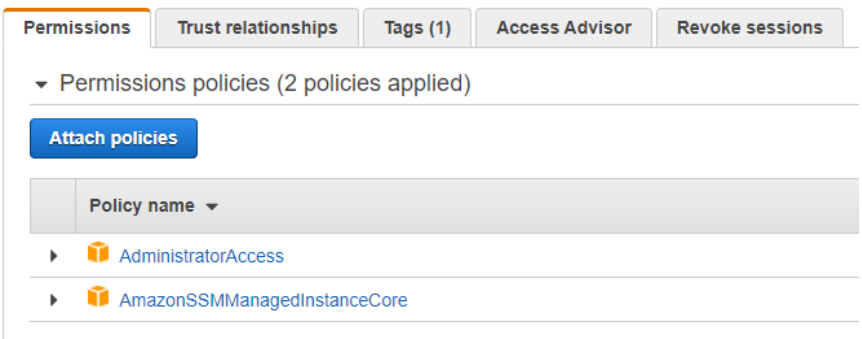
Step 3: If we can update the Permission policy with **AWSRevokeOlderSessions** inline **DENY** for any sessions older than **NOW** will resolve the issue.

Step 4: Now woffy can't use the existing credentials to access the resource due to explicit deny also he can't assume a new Role since he is not under Trust Policy. So, he will be loosing the access to access any AWS Resources

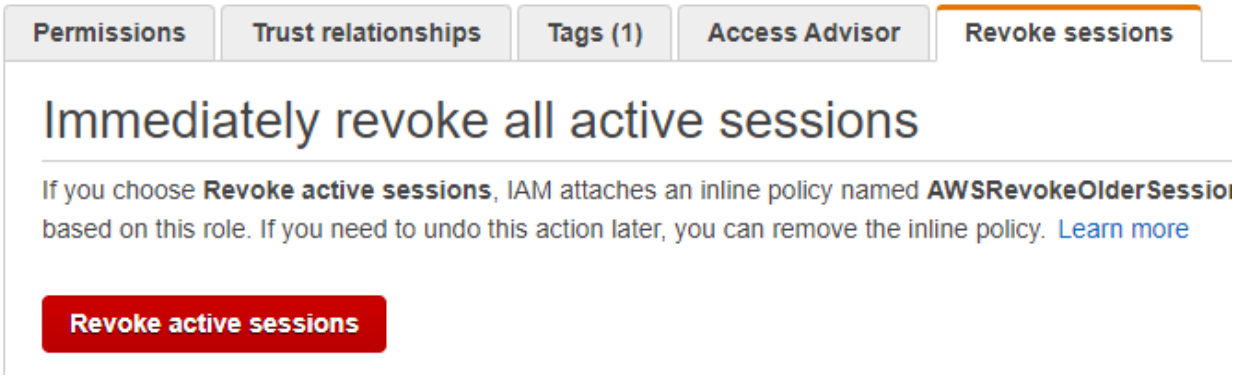
Demo:



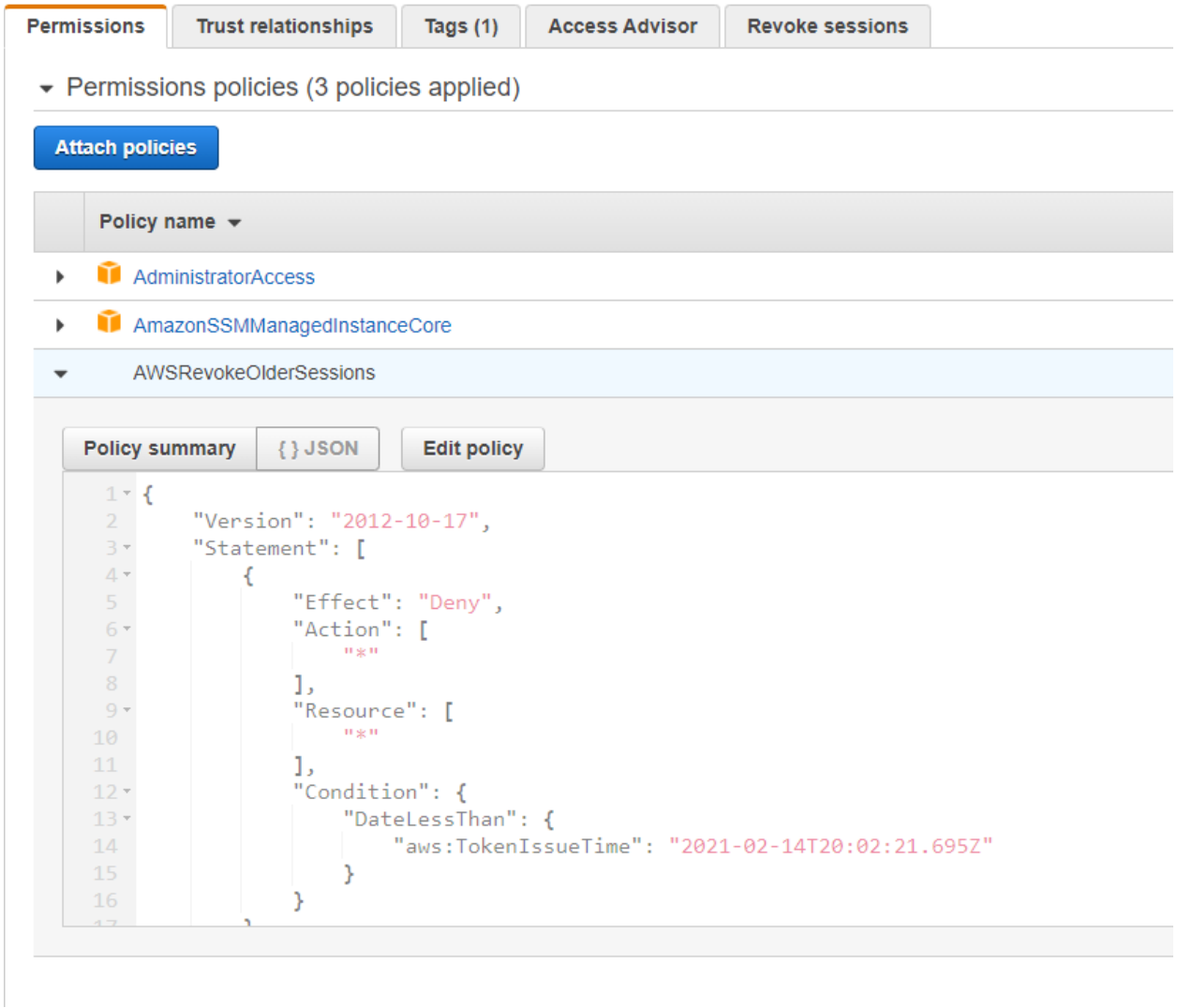
Points to remember: Before fix



Solution: Attache Revoke Session



After fix:



Implement the following: