Access Undenied

Automatically explaining Access Denied errors in AWS

Noam Dahan, Ermetic, @NoamDahan

ermetic

A day in the life

- A developer creates a lambda that needs to access a bucket.
- [ERROR] ClientError: An error occurred (AccessDenied) when calling the ListObjectsV2 operation: Access Denied
- Why isn't this working??? 😭
- It's you and your least-privilege again!!! 😠
- Please fix this 🙏

Access: Denied

- AWS errors are mostly opaque
- They could come from various sources:
  - Identity policy
  - Resource policy (same/cross account)
  - Service Control Policy
  - Permission boundary
  - VPC endpoint policy
  - Session policy
  - Tag policy

Problematic solutions

- Pressure to restore/create usability
- Permission drift
- “Don’t touch this, it might break”

The Galápagos of error messages

- An error occurred (AccessDenied) when calling the **<action>** operation: Access Denied
- User: **<principal>** is not authorized to perform: **<service>:<action>** on the specified resource
- User: **<principal>** is not authorized to perform: **<service>:<action>** with an **explicit deny**
- User: **<principal>** is not authorized to perform: **<service>:<action>** on resource: **<resource>**
- You are not authorized to perform this operation. Encoded authorization failure message: **<encoded_message>** (Client.UnauthorizedOperation)
- You are not authorized to perform this operation.
- **And many more…** 😱

AWS has been getting better

- But not a full solution

```
→  ~ aws --region us-west-1 codecommit get-repository --repository-name test-repository

An error occurred (AccessDeniedException) when calling the GetRepository operation:
User: arn:aws:sts::123456789012:assumed-role/AWSReservedSSO_DevAccess_obfuscated_/michael
is not authorized to perform: codecommit:GetRepository on resource:
arn:aws:codecommit:us-west-1:123456789012:test-repository with an explicit deny in a
service control policy
```

- Principal, Action, and resource
- Reason for deny + policy type

The new error messages:

- Really awesome!
  - Consistent and predictable error messages are 🔥 🔥 🔥
- **Note:** only gives the type of policy causing the error
  - So still some work to do after that
  - For security reasons – this may be as much as we'll get
  - Manual – not for scale
- Service coverage gaps: S3, SQS, EKS, SSO, EFS, Batch, etc.

Access: Undenied!

- Get an AccessDenied CloudTrail event(s)
- Gives an answer for why access was denied, and what steps should be taken to facilitate least-privilege access
- Scans permissions landscape ad-hoc
- Supports: SCPs, resource policies, permission boundaries, identity policies, conditions
- Open source 😀

Why CloudTrail?

- Pros:
  - (almost) always available
  - All originators, including service originators
  - Scalable
  - Relatively detailed
- Cons:
  - Inconsistent format
  - Sometimes access denied doesn’t log details
  - Sometimes access denied isn’t logged (aws pls)

How it works

- Parse CloudTrail event
- Extract PARCs
- Gather all relevant policies
- Manipulate inputs as needed
- iam:GetContextKeysForCustomPolicy -> Generate context
- iam:SimulateCustomPolicy
- Create actionable output

How it works

- Parse CloudTrail event
- Extract PARCs
- Gather all relevant policies
- Manipulate inputs as needed
- iam:GetContextKeysForCustomPolicy -> Generate context
- iam:SimulateCustomPolicy
- Create actionable output

Challenge #1: PARCs and recreation

- We have to identify the Principal, Action (permission), Resource (+ resource account) and Context of the request
- Challenges:
  - Action ≠ permission
  - Dependent actions (PassRole, tagging)
  - Extracting the resource and resource account (error message, resources key, request parameters)?
  - Multiple resources

Challenge #2: SimulateCustomPolicy

- Pros:
  - Universally available, self-contained
  - Gives detailed output
- Cons:
  - Does not support IAM roles as callerArn or IAM Roles+resource policies
  - Doesn't fully support SCPs
- **Useful tool, but you have to manipulate its inputs**
- Looking forward: an open-source SimulateCustomPolicy?

SimulateCustomPolicy: The SCP problem

- The SCP evaluation rule:
  - Action has to be allowed **at every level of the hierarchy** (root, OUs, account)
- SimulateCustomPolicy handles boundaries well, but only lets you use **one** boundary per call, doesn't handle SCPs well
- For each org hierarchy
  - Take all attached SCPs
  - Merge into one policy
  - Check with that policy as a boundary
  - Translate back into original names

Demo: Access Undenied

- Previously on...
  - First action: Implicit deny
  - Second action: Implicit deny in resource policy (cross-account)
  - Third action: Explicit deny in a Service Control Policy

Scaling up

- Some interesting possibilities!
- Aggregating similar errors
- Filtering out routine denials

Roadmap going forward

- Support for more resources with resource policies
- Service-specific permission mechanisms (KMS grants)
- VPC endpoint policies
- Session policies
- Multiple replies

Acknowledgements

- Ian Mckay's IAM dataset: https://github.com/iann0036/iam-dataset/
- Ben Kehoe's aws-error-utils: https://github.com/benkehoe/aws-error-utils

Q&A

- Thank you for listening!
- noam@ermetic.com
- Twitter: @NoamDahan
- Github: noamsdahan