



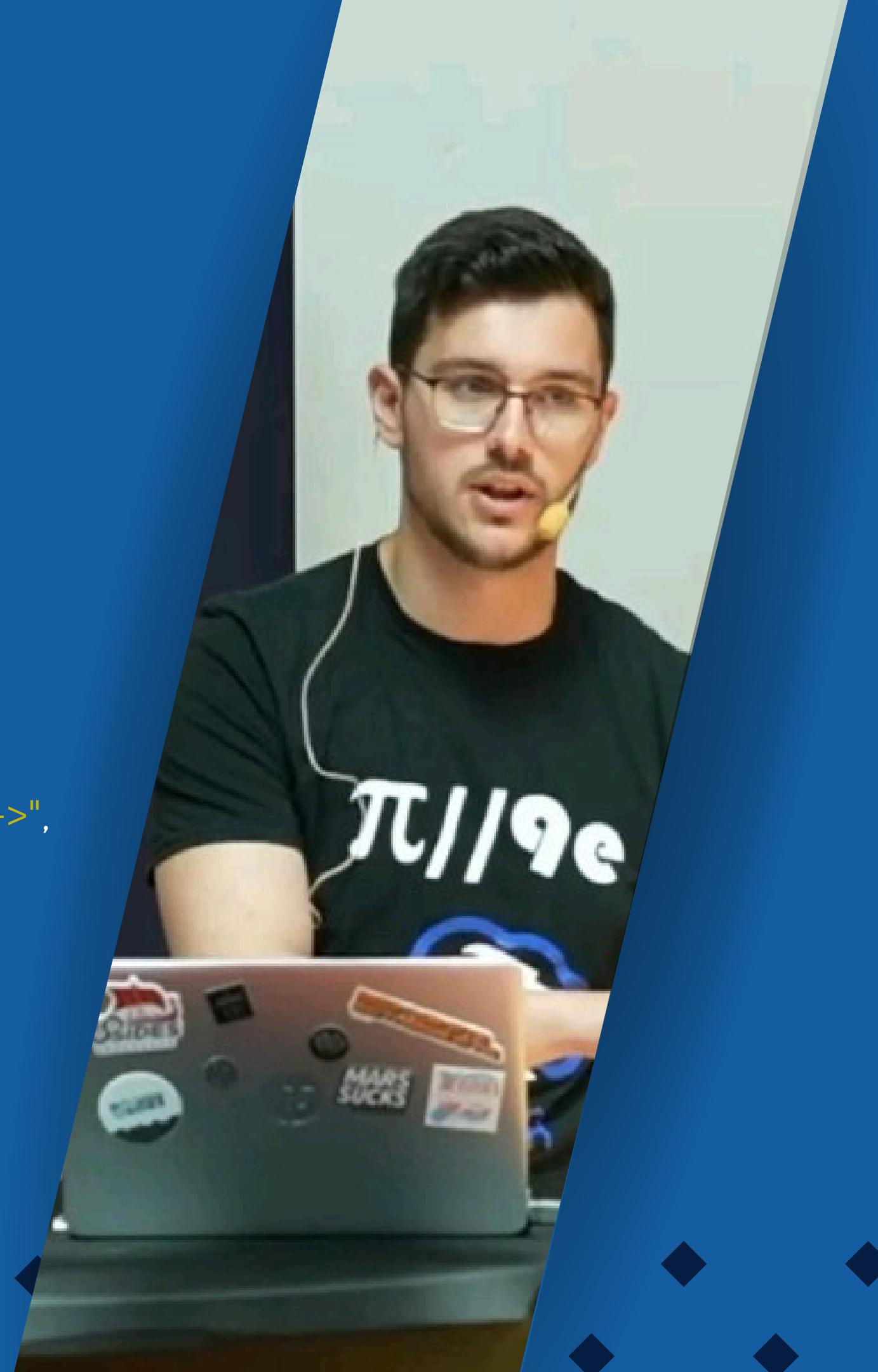
# ENCRYPTING BUCKETS FOR COMPLIANCE AND RANSOM HOW ATTACKERS CAN USE KMS TO RANSOMWARE S3 BUCKETS



```
(blackhat())(Nebula) >>> use credentials gl4ssesb01  
(blackhat())(Nebula) >>> getuid
```

```
-----  
UserName: gl4ssesb01  
-----
```

```
{  
    "UserName": "gl4ssesb01",  
    "UserInfo": {  
        "UserName": "gl4ssesb01",  
        "Name": "Bleon Proko",  
        "Description": "This guy ----->",  
        "Position": "Cloud Researcher @ Permiso\"",  
        "Email": "bleon.proko@protonmail.com",  
        "Blog": "https://blog.pepperclipp.com/"  
    }  
}
```



# Our TAs



Financially motivated attackers

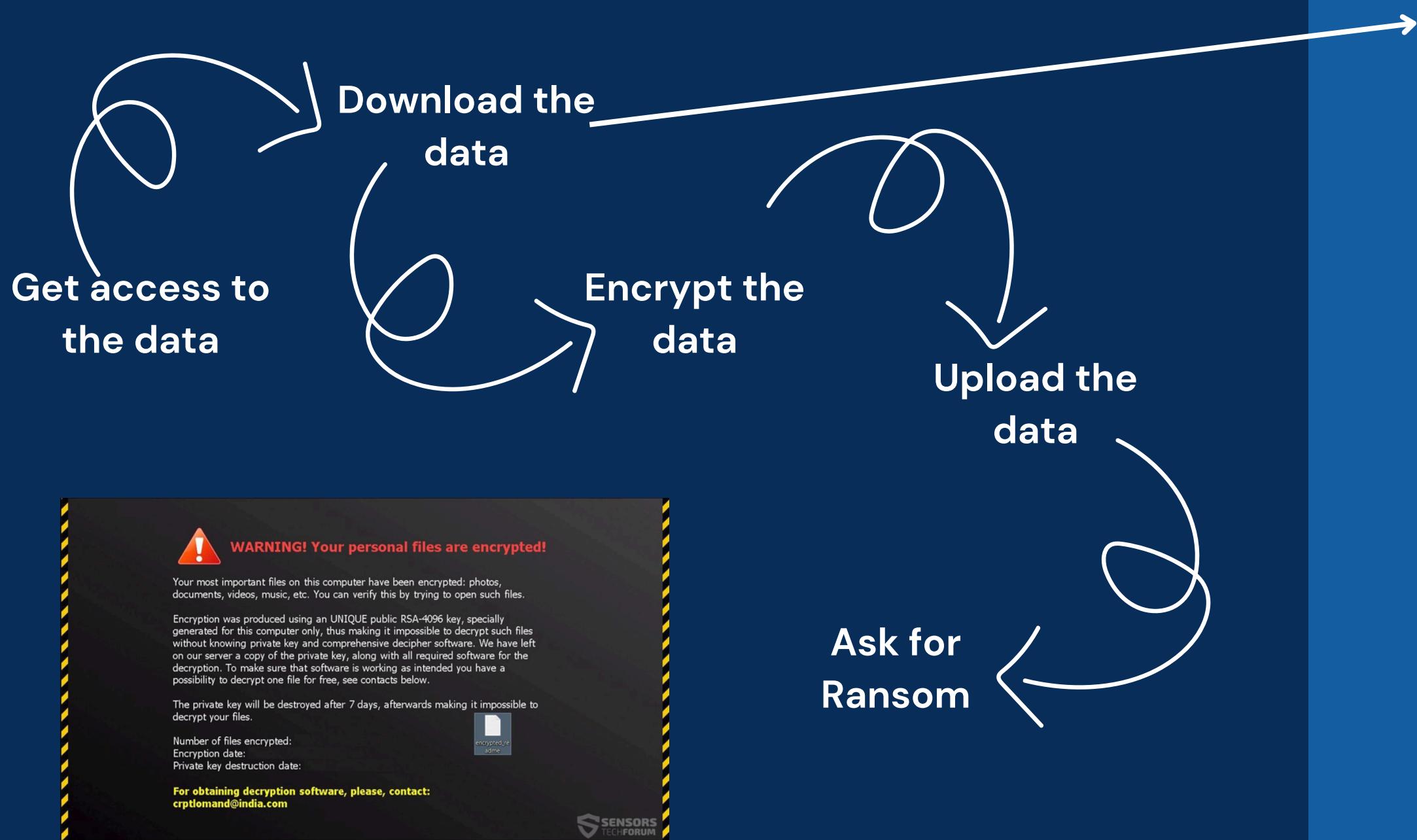
Not Nation Supported

Relatively low budget

Ransomware is a goal to them, since it does not require a large infrastructure, not large funds and it's very profitable.

# Ransomware

**Ransomware:** a type of malicious software designed to block access to a computer system until a sum of money is paid.



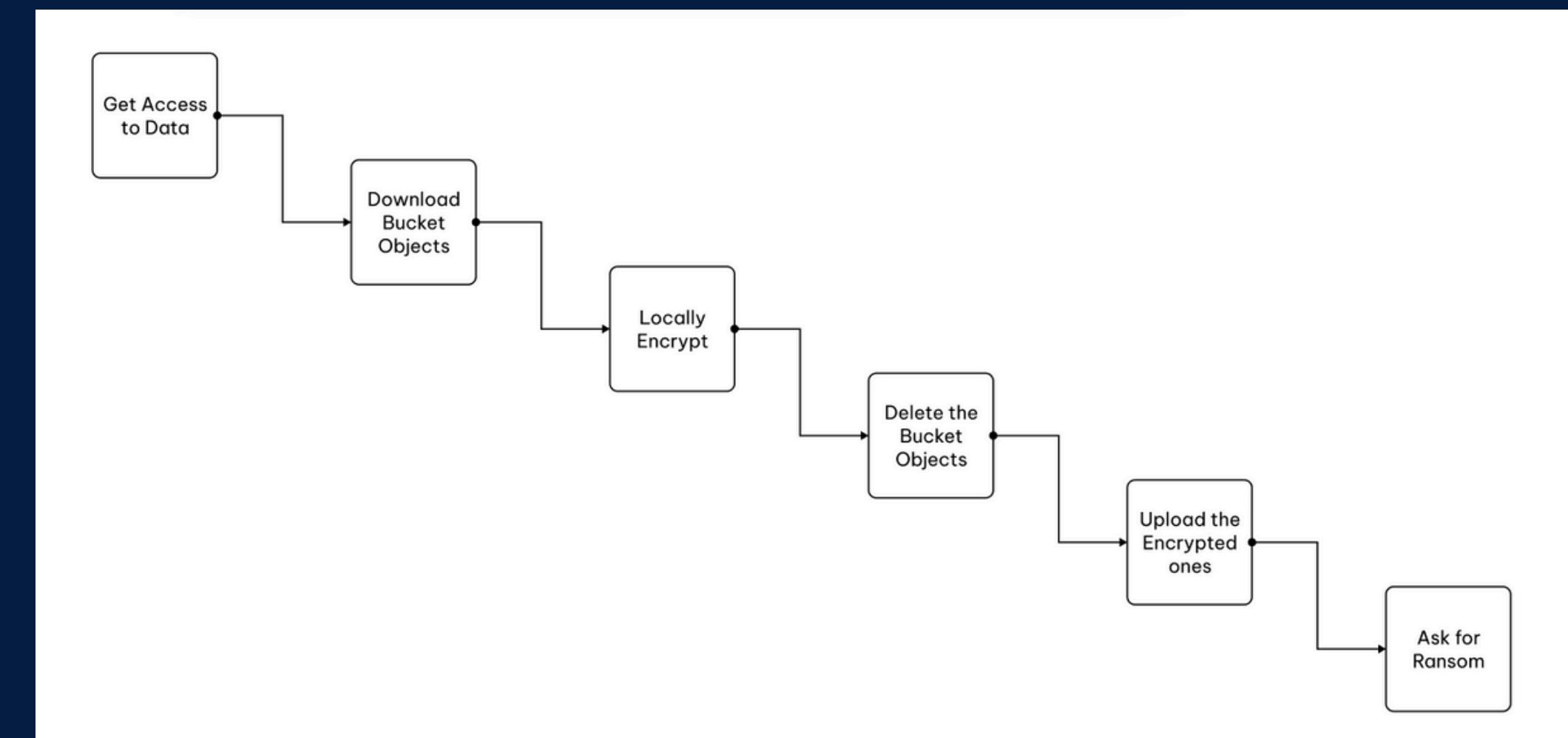
In case of not paying the ransom, the TA can release/sell the data.



# WHAT ABOUT AWS S3 DATA?

For AWS S3, we need to make sure you have access to get and put files on the bucket, as well as being able to bypass security limitations that prevent data loss.

- Bucket Access Privileges (`s3>ListObjects`, `s3:GetObject`, `s3:PutObject`)
- Data Loss Prevention (**Bucket Versioning** and **MFA Delete**)



# RANSOMWARE BOTTLENECK

When faced with Hundreds/Thousands of Terabytes of data, downloading them will have some issues:

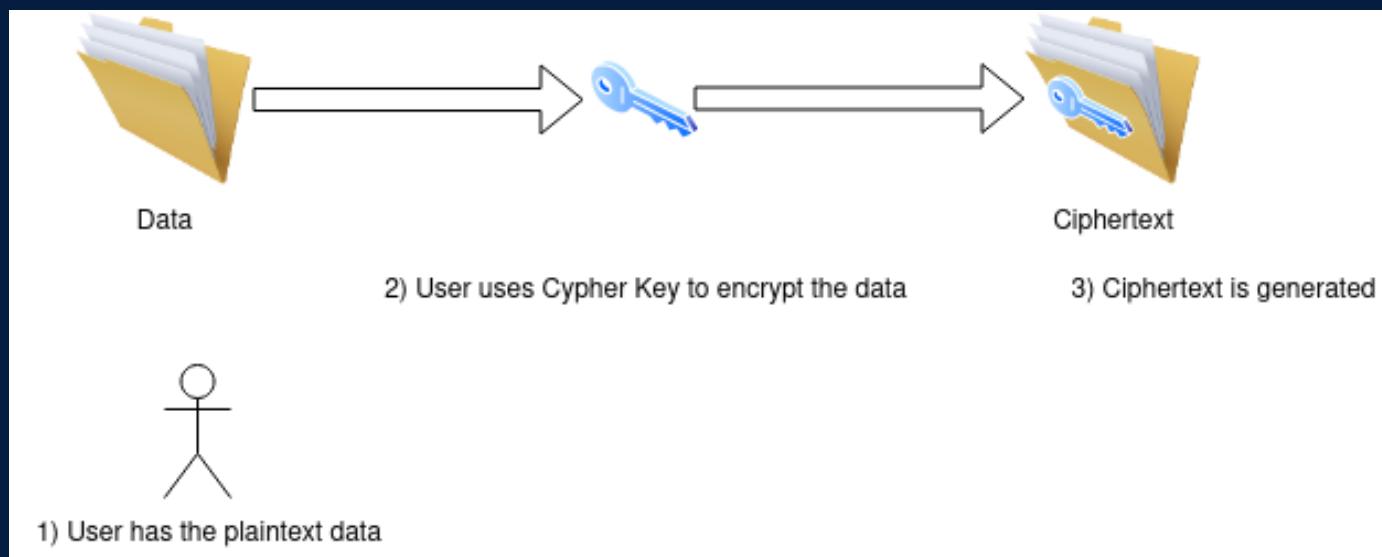
- Storage Costs - Whether it's S3, local storage or even other storage like webdav, having a large amount of data costs a lot
- Data logs - Downloading a large amount of data takes time and creates a lot of logs that will be triggering a warning.

Our TAs have to find a way to encrypt the data, have access to the Plaintext at will and not have to spend much time and money.

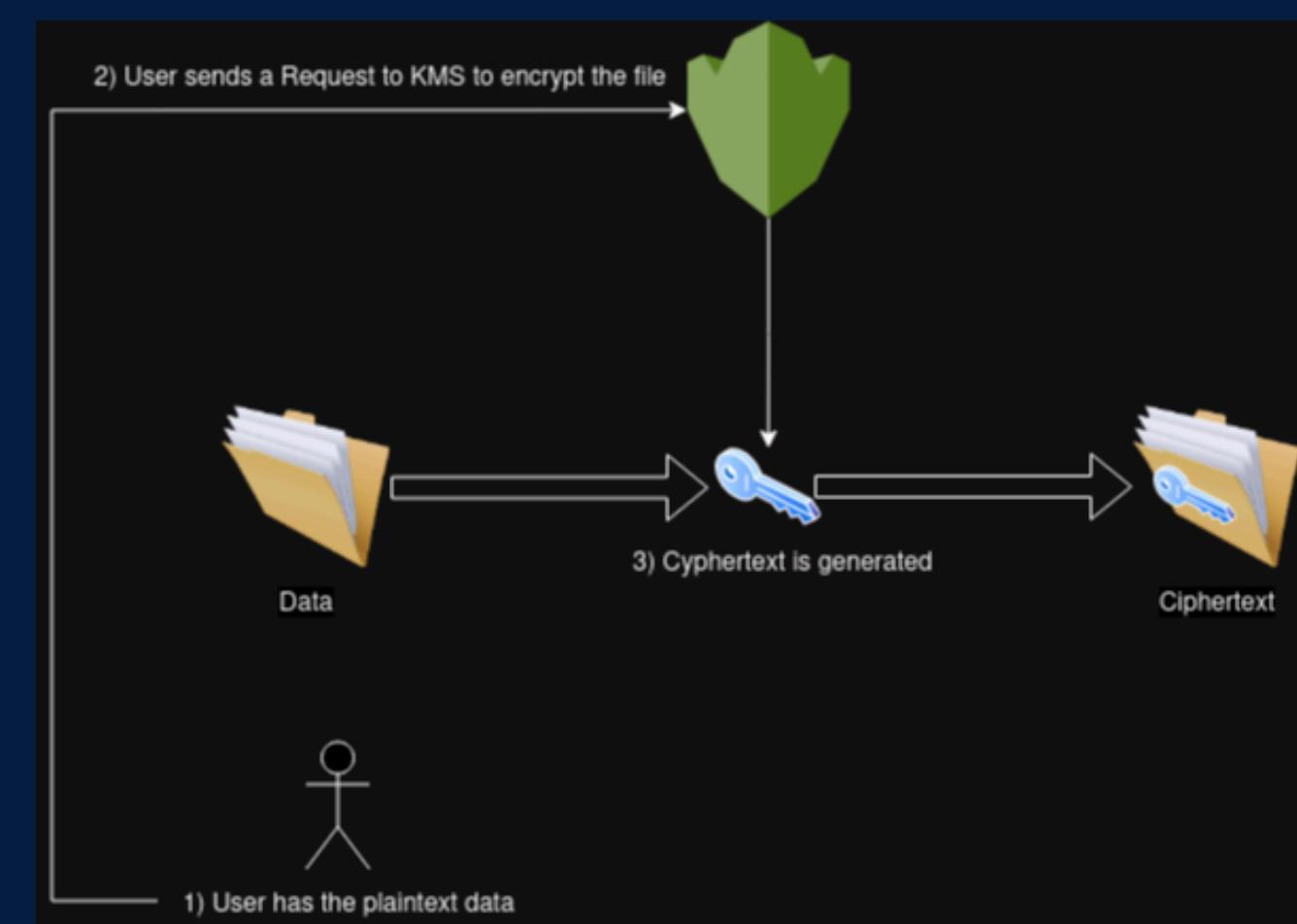
# KMS ENCRYPTION

AWS KMS is an AWS service that allows management and usage of Encryption Keys, as well as encrypting and decrypting data without the need of the key content.

## Traditional Encryption



VS



# KMS KEY POLICY

AWS KMS manages access to the CMK using the Key Policy. The catch is no other identity, not even **root** has access to it.

```
{  
  "Version": "2012-10-17",  
  "Id": "key-consolepolicy-3",  
  "Statement": [  
    {  
      "Sid": "Allow attachment of persistent resources",  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": [  
          "arn:aws:iam::012345678912:user/testuser"  
        ]  
      },  
      "Action": "kms:*",  
      "Resource": "*"  
    }  
  ]  
}
```

# S3 BUCKET ENCRYPTION

S3 buckets can be configured to encrypt the objects uploaded on them. This requires access to the key on KMS API Call kms:Encrypt

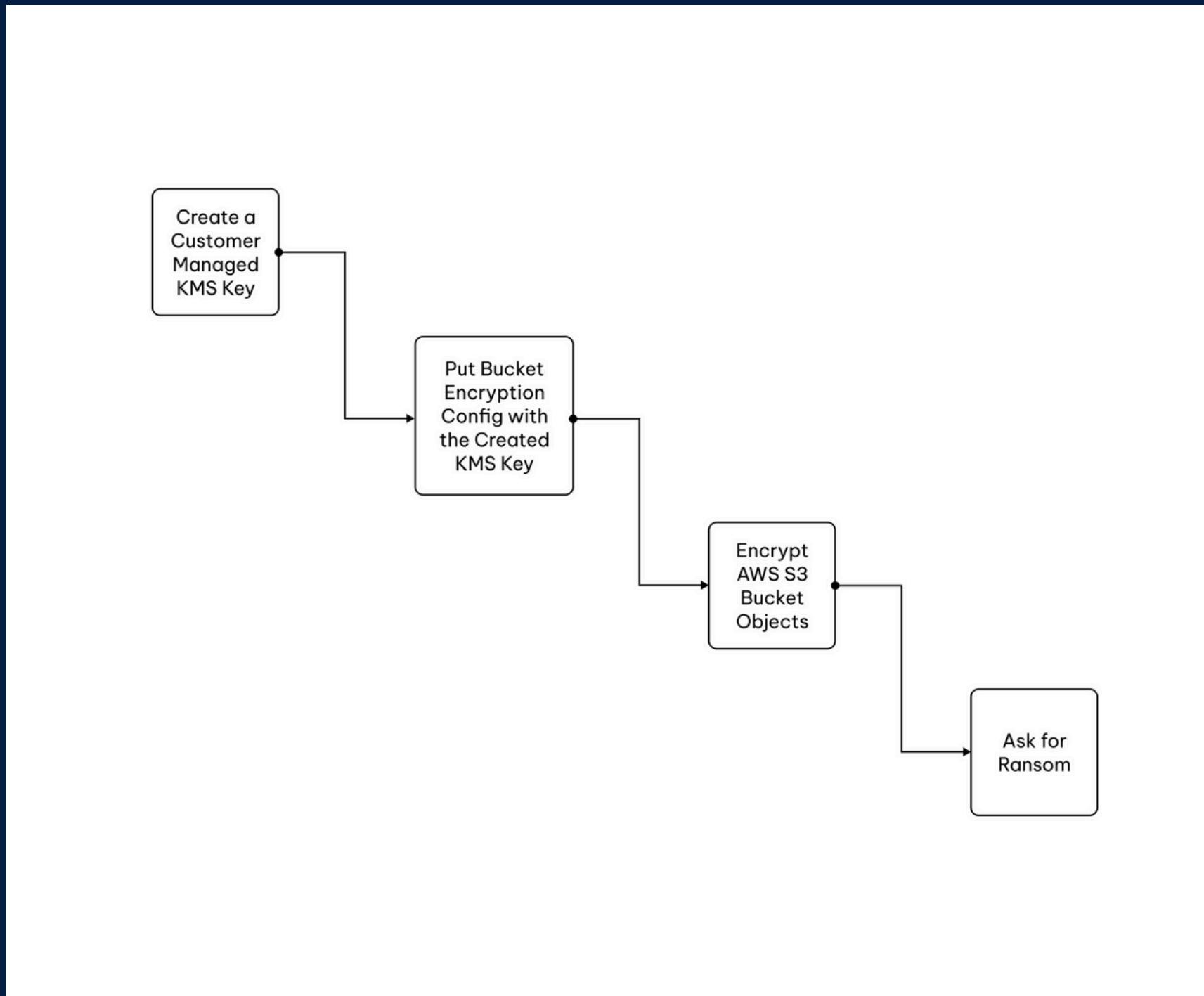


An S3 Bucket configured to encrypt the objects will have a Bucket Encryption Configuration attached to it:

```
{  
    "ServerSideEncryptionConfiguration": {  
        "Rules": [  
            {  
                "ApplyServerSideEncryptionByDefault": {  
                    "SSEAlgorithm": "aws:kms",  
                    "KMSMasterKeyID": "arn:aws:kms:us-east-1:  
cd027e4f29" :key/dfd9abba-c1a3-4f27-adc0-49  
                },  
                "BucketKeyEnabled": true  
            }  
        ]  
    }  
}
```

Encryption Config will define the KMS key that will be used to encrypt the objects

# RANSOMWARE DE-BOTTLENECK



## Notes:

- Key is Created using a compromised user on the target's account
- Key resides on the target's account
- Only the compromised identity has access to the key

# RANSOMWARE ISSUES

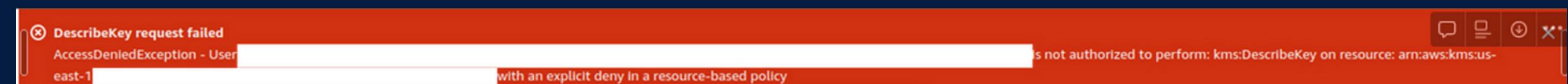
- **Key is Created using a compromised user on the target's account:** Even though KMS Key policy denies access to other identities, an Administrator or the root account still has access to all the identities
- **Key resides on the target's account:** This means the root will still have some sort of access to the key, even if only some identities will have access to it.
- **Only the compromised identity has access to the key:** Going over to point nr.1, only that identity has access to the key, but the Administrator/root has access to the identity.

# FINDING IDENTITY WITH ACCESS TO THE KEY

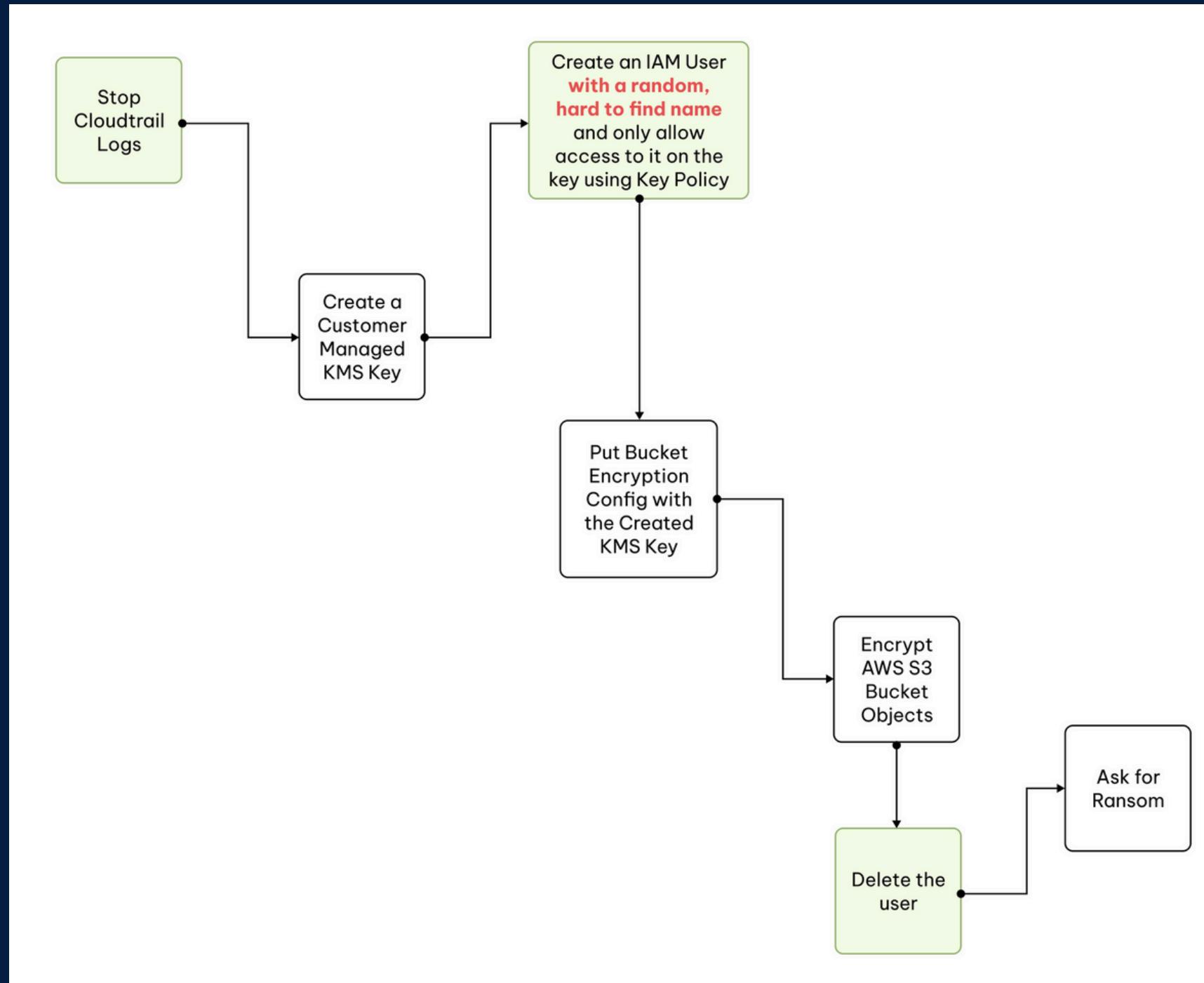


- Bruteforce KMS permissions
  - Create an IAM policy that allows kms:DescribeKey on the specific key
  - Get a list of all the identities (users and roles)
  - Attach the policy to each user and role
  - Modify each role's Trust Policy to allow the identity that is running the script to Assume them
  - If the user has 1 set of credentials (AccessKey and SecretKey), since by default each user can have 2, create a 2nd. If the user has 2 sets of credentials, delete the 1st and create a new one. As for the roles, they can use sts:AssumeRole to get a set of temporary credentials
  - Using the credentials of the identities, run kms:DescribeKey
  - Lastly detach the policy on the users and roles, detach the policy and delete the key, detach the policy, delete the users' credentials, and delete the policy
- Looking at CloudTrail logs
  - Look for kms>CreateKey and kms:PutKeyPolicy

To find the KMS Key ID, just go to KMS Page on AWS Console and look for the access error on the top



# RE-DE-BOTTLENECK-ING THE RANSOMWARE



## Notes:

- This will stop the logging, so reading logs to find the identity that has access to the key will not be possible
- A user on AWS can have characters **A-Z, a-z, 0-9, and + = , . @ \_ - (hyphen)**, for example **qQ5ybE3L05eS.=8d6MuWN@wTSyl.OCs.@.=+UTbS90ECRJoNYu**

# RANSOMWARE ISSUES TIME

Even if CloudTrail Trail is stopped or deleted,  
Event History will still be keeping the logs

```
{  
    "eventVersion": "1.08",  
    "userIdentity": {  
        "type": "IAMUser",  
        "principalId": "AIDAUNVW3VEZYWRSN5SON",  
        "arn": "arn:aws:iam:█████████████████████:user/qQ5ybE3L05eS.=8d6MuWN@wTSyl.0Cs.@.=+UTbS90ECRJoNYu",  
        "accountId": "304252430643",  
        "accessKeyId": "AKIAUNVW3VEZT4HH7X4B",  
        "userName": "qQ5ybE3L05eS.=8d6MuWN@wTSyl.0Cs.@.=+UTbS90ECRJoNYu"  
    },  
    "eventTime": "2023-05-22T15:42:03Z",  
    "eventSource": "kms.amazonaws.com",  
    "eventName": "CreateKey",  
    "awsRegion": "us-east-1",  
    "sourceIPAddress": "46.252.45.8",  
    "userAgent": "aws-cli/1.24.8 Python/3.10.7 Linux/5.19.0-41-generic botocore/1.29.22",  
    "requestParameters": {  
        "keyUsage": "ENCRYPT_DECRYPT",  
        "description": "ransomware-key",  
        "policy": "{\"Version\": \"2012-10-17\", \"Id\": \"key-consolepolicy-3\", \"Statement\": [{\"Sid\": \"Allowattachmentofpersistenteresources\", \"Effect\": \"Allow\", \"Principal\": \"AWS\", \"Action\": \"kms:*\", \"Resource\": \"*\"]}]",  
        "multiRegion": true,  
        "customerMasterKeySpec": "SYMMETRIC_DEFAULT",  
        "keySpec": "SYMMETRIC_DEFAULT",  
        "origin": "AWS_KMS",  
        "bypassPolicyLockoutSafetyCheck": true  
    },  
    "responseElements": {  
        "keyMetadata": {  
            "aWSAccountId": "████████████████",  
            "keyId": "mrk-353260cf6ce4fe1a3af1364ca18edb4",  
            "arn": "arn:aws:kms:us-east-1:████████████:key/mrk-353260cf6ce4fe1a3af1364ca18edb4",  
            "creationDate": "May 22, 2023, 3:42:02 PM",  
            "enabled": true  
        }  
    }  
}
```

Deleting a user will delete the access to the key even if an identity with the same name is recreated

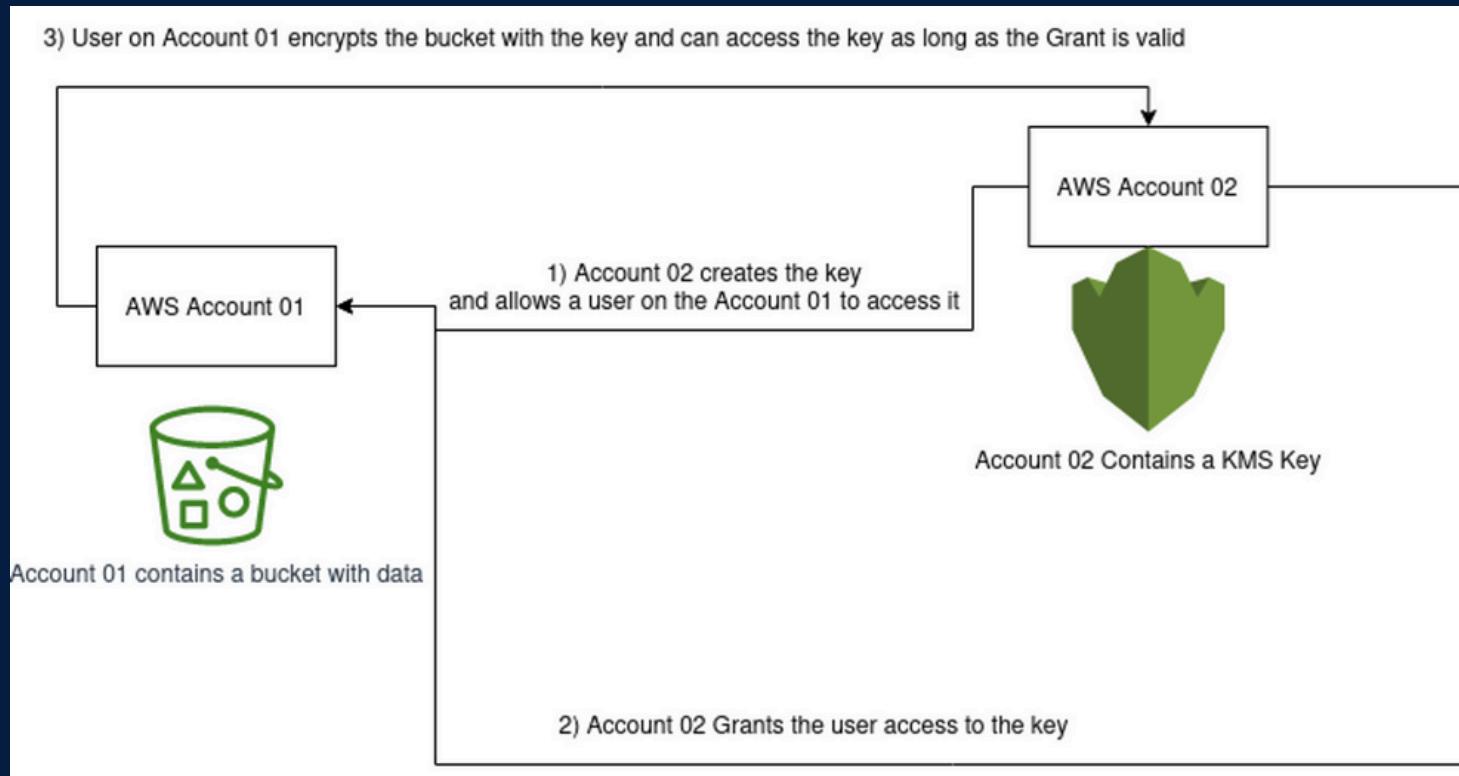
```
:~/Desktop$ aws iam delete-access-key --user-name 'qQ5ybE3L05eS.=8d6MuWN@wTSyl.0Cs.@.=+UTbS90ECRJoNYu' --access-key-id 'AKIAUNVW3VEZT4HH7X4B' --profile administrator_user  
:~/Desktop$ aws iam delete-user --user-name 'qQ5ybE3L05eS.=8d6MuWN@wTSyl.0Cs.@.=+UTbS90ECRJoNYu' --profile administrator_user
```

Now, even if the target creates the identity with the correct name, they cannot access the key, making the bucket objects encrypted and inaccessible.

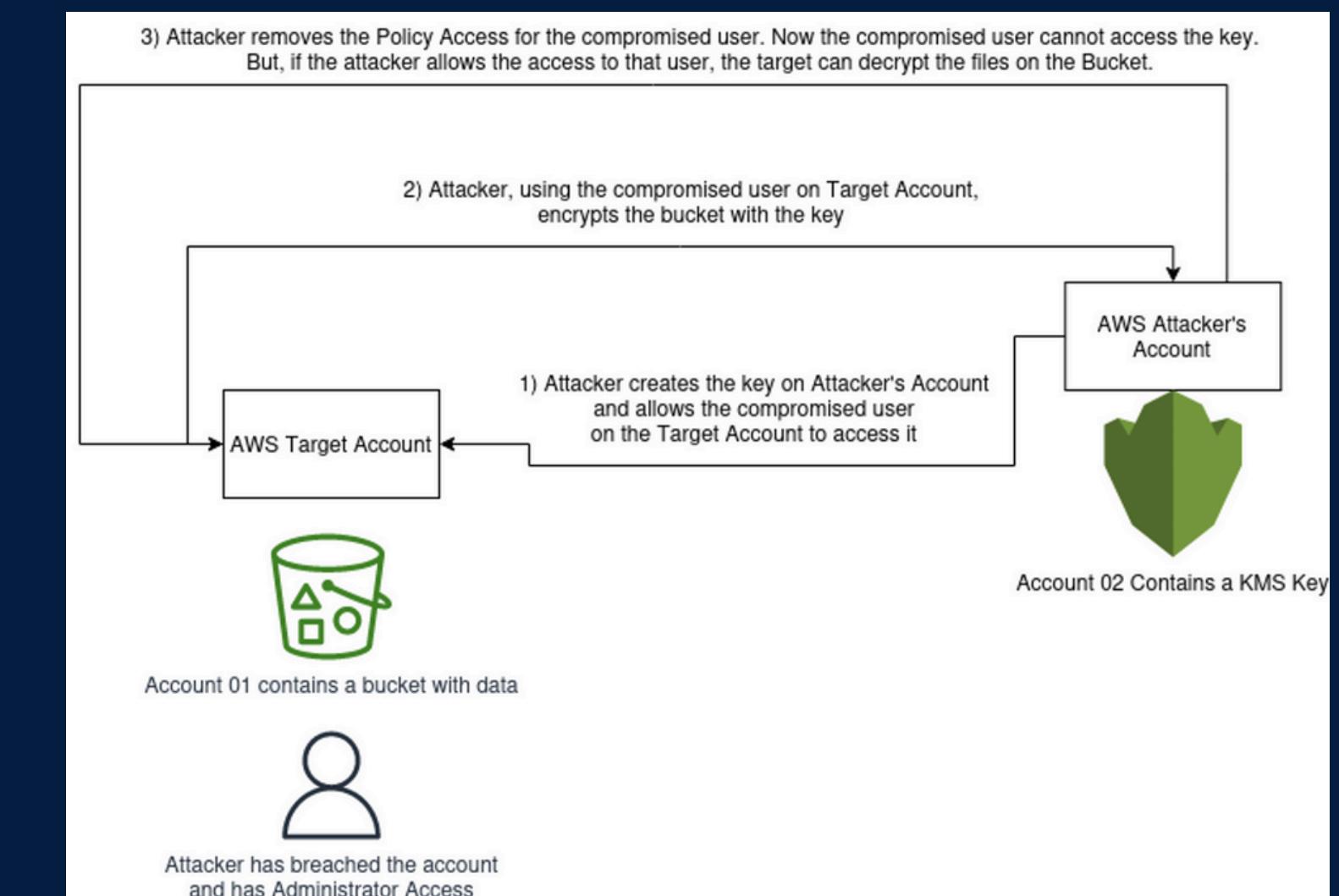
```
:~/Desktop$ aws kms describe-key --key-id "arn:aws:kms:us-east-1:304252430643:key/mrk-353260cf6ce4fe1a3af1364ca18edb4" --profile ransomUser  
An error occurred (AccessDeniedException) when calling the DescribeKey operation: User: arn:aws:iam::304252430643:user/qQ5ybE3L05eS.=8d6MuWN@wTSyl.0Cs.@.=+UTbS90ECRJoNYu is not authorized to perform : kms:DescribeKey on resource: arn:aws:kms:us-east-1:304252430643:key/mrk-353260cf6ce4fe1a3af1364ca18edb4 because no resource-based policy allows the kms:DescribeKey action
```

This leads to data destruction, but not ransomware.

# KMS CROSS ACCOUNT KEY ACCESS



Which results  
in the attack



SHARING  
IS  
CARING

# AN OBJECT A DAY KEEPS DATA ACCESS AWAY

Server-side encryption settings [Info](#)  
Server-side encryption protects data at rest.

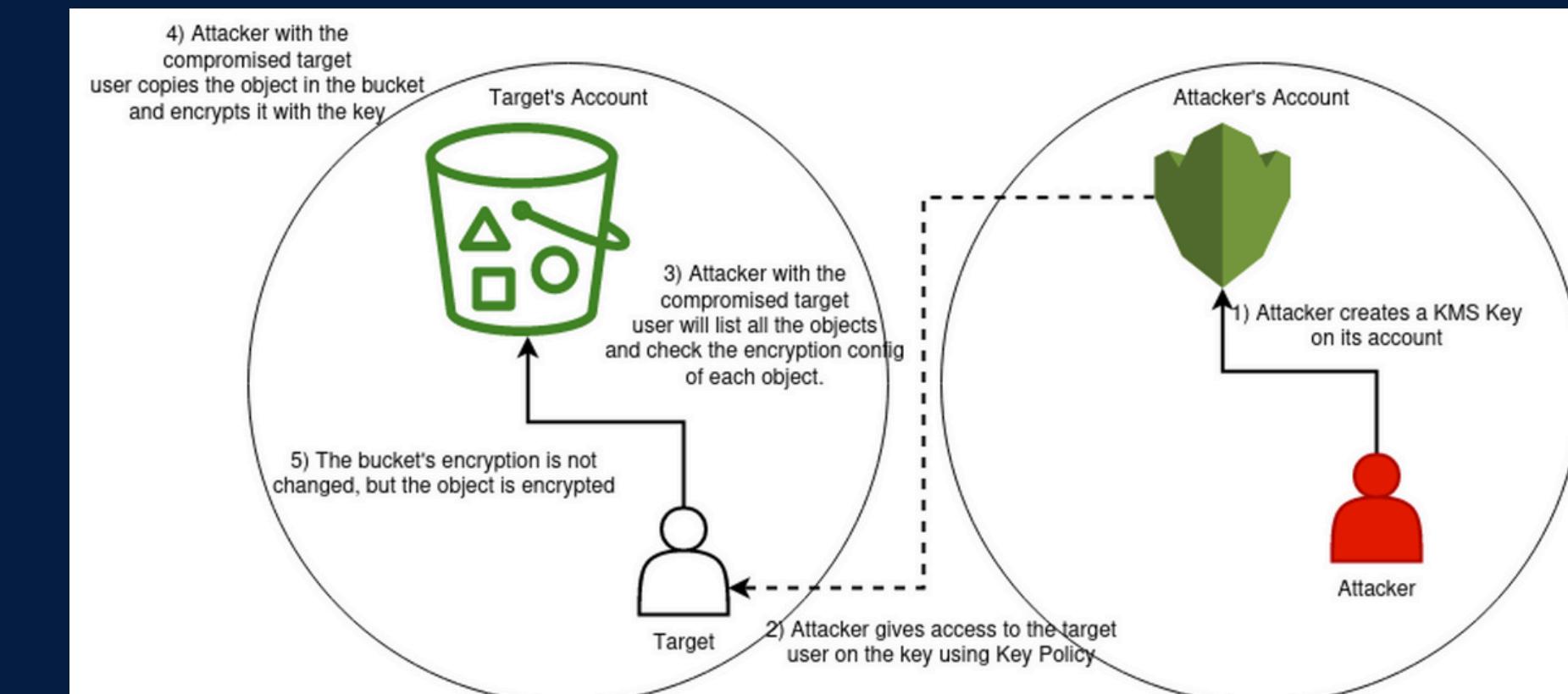
Encryption key type [Info](#)  
AWS Key Management Service key (SSE-KMS)

Encryption key ARN  
 arn:aws:kms:us-east-1: :key/a1a4533f-765a-47dd-82a1-f9be4a2ba7a7 [Copy](#)

Bucket Key  
When KMS encryption is used to encrypt new objects in this bucket, the bucket key reduces encryption costs by lowering calls to AWS KMS. [Learn more](#)

Disabled

Which results  
in the attack



# SCP BOTTLENECK



```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "RestrictKMSOutsideOrg",  
      "Effect": "Deny",  
      "Action": [  
        "kms:*"  
      ],  
      "Resource": [ "*" ],  
      "Condition": {  
        "StringNotEquals": {  
          "aws:ResourceOrgID": "o-xxxxxxxxx"  
        }  
      }  
    }  
  ]  
}
```

Running the KMS command will result in  
access denied

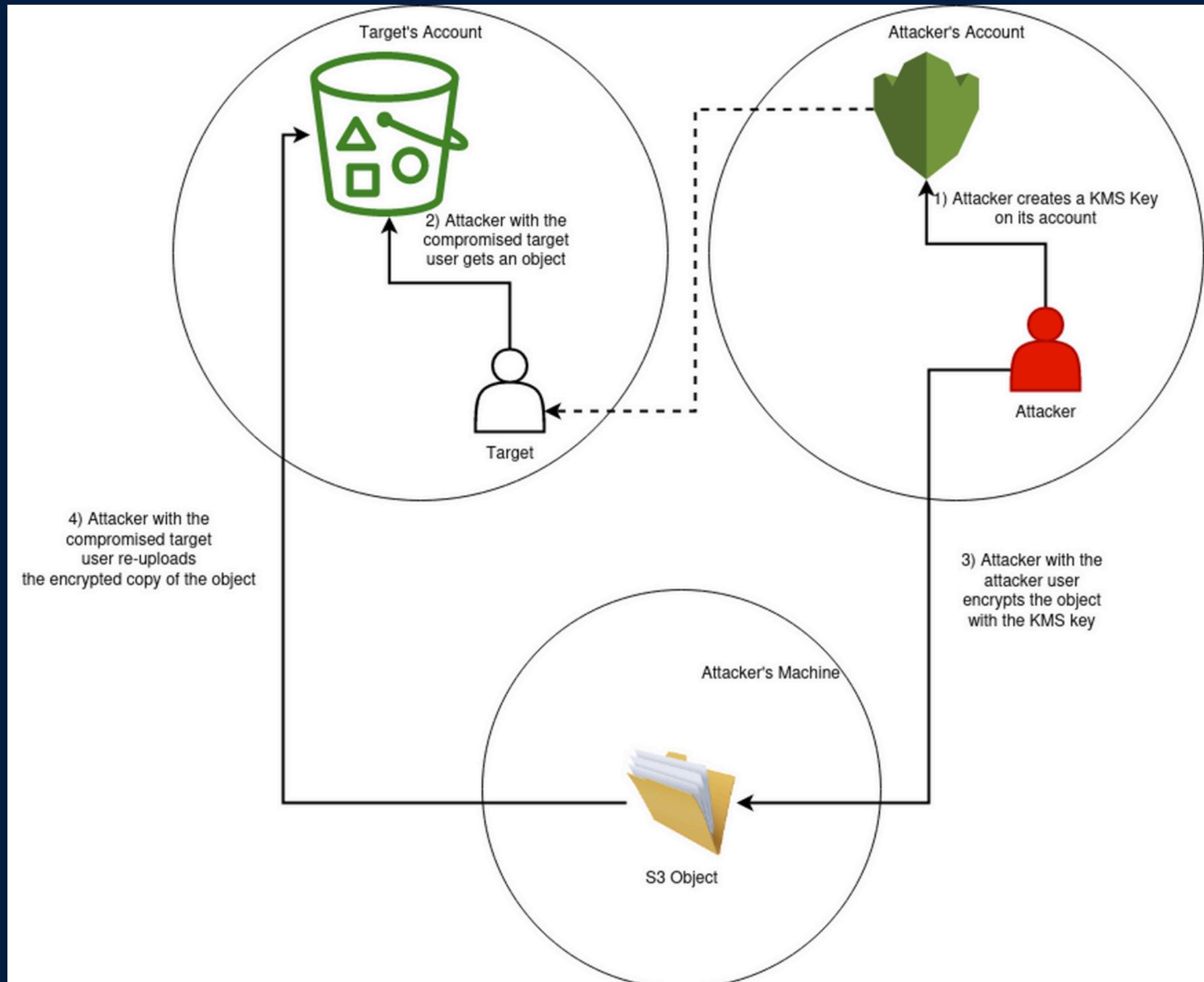
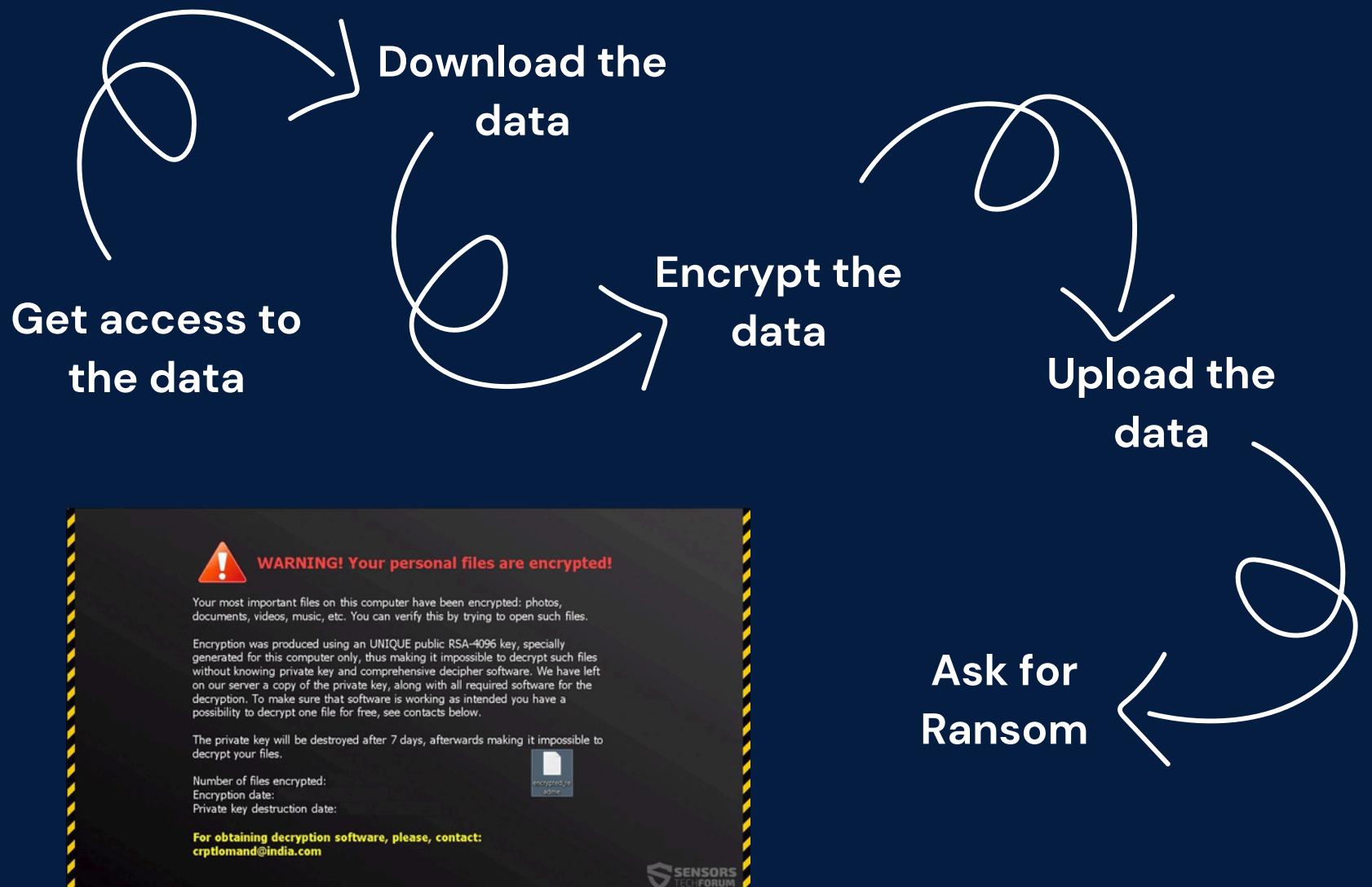
```
:~$ aws kms describe-key --key-id arn:aws:kms:us-east-1:  
7 --profile slaveUser --region us-east-1  
{  
  "KeyMetadata": {  
    "AWSAccountId": "██████████",  
    "KeyId": "a1a4533f-765a-47dd-82a1-f9be4a2ba7a7",  
    "Arn": "arn:aws:kms:us-east-1:  
key/a1a4533f-765a-47dd-82a1-f9be4a2ba7a7",  
    "CreationDate": 1685568493.985,  
    "Enabled": true,  
    "Description": "",  
    "KeyUsage": "ENCRYPT_DECRYPT",  
    "KeyState": "Enabled",  
    "Origin": "AWS_KMS",  
    "KeyManager": "CUSTOMER",  
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",  
    "KeySpec": "SYMMETRIC_DEFAULT",  
    "EncryptionAlgorithms": [  
      "SYMMETRIC_DEFAULT"  
    ],  
    "MultiRegion": false  
  }  
}  
:~$ aws kms describe-key --key-id arn:aws:kms:us-east-1:  
7 --profile slaveUser --region us-east-1  
An error occurred (AccessDeniedException) when calling the DescribeKey operation:  
██████████
```

Before SCP

After SCP

# KEEPING IT SIMPLE WORKS TOO

Going back to the first attack, we can use the same thing while utilizing KMS:



# I'M IN LOVE WITH THIS VERSION OF YOU

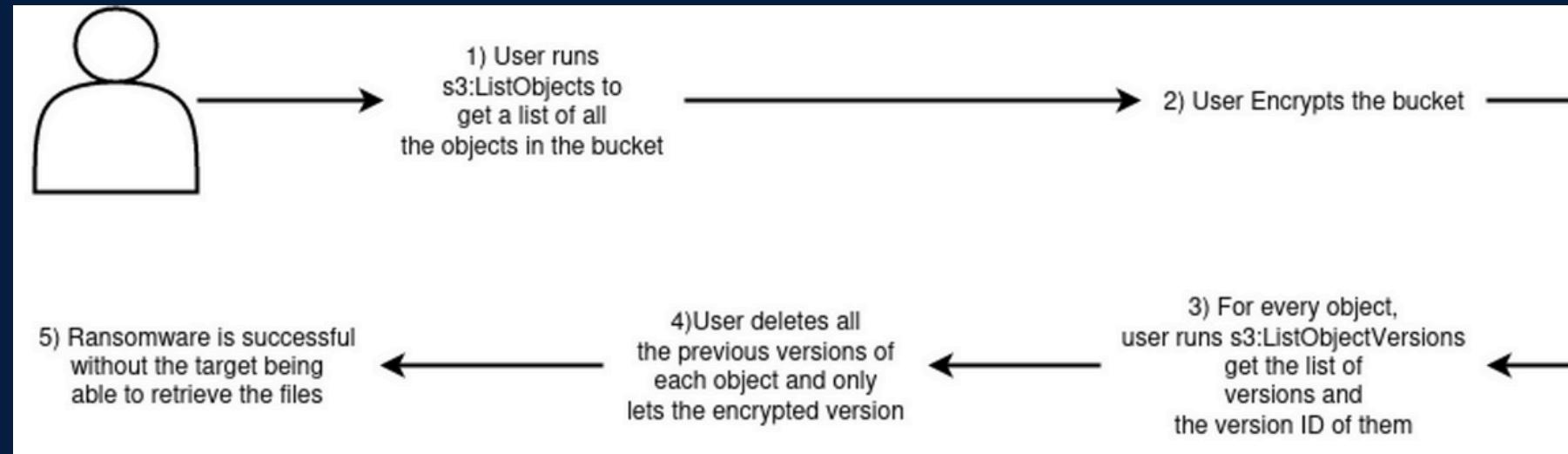


If Bucket Versioning is enabled, if a file is deleted/overwritten, a copy of the old file will still remain.

```
[ec2-user@ip-172-31-3-25 ~]$ aws s3api delete-object --bucket mfadeletedemobucket --version-id BwremsGdWymu711INIV2y604jWUE16QT --key IMG_9806.JPG
An error occurred (AccessDenied) when calling the DeleteObject operation: Mfa Authentication must be used for this request
```

If MFA Delete is enabled, it will not allow deletion of a version unless MFA code is put. This puts an extra protection to the data.

# BYPASSING BUCKET VERSIONING



One way is delete by brute force

```
[~]:~$ aws s3api get-bucket-versioning --bucket supersuperpermisosensitivedata --profile [REDACTED]
{
  "Status": "Enabled",
  "MFADelete": "Disabled"
}
[~]:~$ aws s3api put-bucket-versioning --bucket supersuperpermisosensitivedata --versioning-configuration 'Status=Suspended' --profile [REDACTED]
[~]:~$ aws s3api get-bucket-versioning --bucket supersuperpermisosensitivedata --profile [REDACTED]
{
  "Status": "Suspended",
  "MFADelete": "Disabled"
}
```

The other one is to just disable it from the bucket

# PERMISO COVERAGE



Alert	Description
PO_AWS_CLOUDTRAIL_LOGGING_STOPPED_1	An attacker may attempt to disable a CloudTrail trail in order to avoid detection. Successful execution of clouptrail:StopLogging will disrupt recording and delivery of CloudTrail logs.
PO_AWS_CLOUDTRAIL_TRAIL_DELETED_1	An attacker may attempt to delete a CloudTrail trail in order to avoid detection. Successful execution of clouptrail>DeleteTrail will disrupt recording and delivery of CloudTrail logs.
PO_AWS_KMS_KEY_CREATION_LOCKOUT_BYPASS_1	An attacker has created a KMS Key, provided a policy, and has set the bypassPolicyLockoutSafetyCheck control to true. This may indicate an attempt to lockout the key making it unusable, resulting in encrypted resources that can not be unencrypted.
PO_AWS_KMS_KEY_UPDATE_LOCKOUT_BYPASS_1	An attacker has updated a KMS Key, provided a policy, and has set the bypassPolicyLockoutSafetyCheck control to true. This may indicate an attempt to lockout the key making it unusable, resulting in encrypted resources that can not be unencrypted.
PO_AWS_S3_VERSIONING_DISABLED_1	An S3 bucket with versioning previously Enabled has been suspended. This setting helps to protect against accidental or malicious deletion of objects by providing backups as versions. An attacker may disable this feature and delete backups to cause destruction or to ensure successful attacks such as ransomware.
PO_AWS_S3_MFADELETE_SETTING_DISABLED_1	An S3 bucket with versioning setting 'MFADelete' previously set to Enabled, has been Disabled. This setting helps to protect against accidental or malicious deletion of objects or versions by requiring MFA and will also require MFA if trying to disable versioning altogether. It is often used as part of a defense in depth strategy against ransomware.
PO_AWS_S3_EXTERNAL_ENCRYPTION_KEY_SET_1	An S3 bucket encryption key has been set where the key comes from an unknown external account. This may indicate an attempt at data lockout tactics, including ransomware.



# CONCLUSIONS

## Key Access

Be vigilant and continuously check who has access to sensitive data or sensitive services like KMS.

## Bucket Versioning

Always enable bucket versioning + MFA delete with least privilege in mind for the identities that have access to MFA.

## SCP Usage

If your account is part of an organization, the account and organization level access should be limited.



# MENTIONING



After finishing this blog and looking for other ways to ransomware, we noticed RhinoSec did a blog a while ago on the Cross Account Key Ransomware. Since they found this first, it is fair to have them mentioned for it.

## S3 Ransomware Part 1: Attack Vector

In part one of this two-part blog series, we detail the attack vector of Amazon S3 Ransomware. We also include a PoC script to demonstrate the attack.

 Rhino Security Labs / Jun 11, 2019

## S3 Ransomware Part 2: Prevention and Defense

In part 2 of this blog series, we discuss various methods to prevent and defend against S3 Ransomware, including a script to check bucket configurations

 Rhino Security Labs / Jun 10, 2019

Thank you!

Any Questions?

