# TABLE OF CONTENTS

3 ABSTRACT

3 SHARED RESPONSIBILITY MODEL

4 HARDENING RECOMMENDATIONS
  4 Adhere to the Principle of Least Privilege:
  5 Encryption and Key Management:
  8 Monitoring and Maintenance:

10 CONCLUSION

10 VERSION HISTORY
ABSTRACT

It is no secret that Rubrik has built an underlying immutable architecture to help secure its data in this era of ransomware that really did not exist a decade ago. The data residing locally on Rubrik’s append-only Atlas filesystem is immutable, however once that data is archived to Amazon S3 via CloudOut it is at the mercy of the customer’s Identity and Access Management configuration. In today’s age of ransomware and advanced persistent threats it is more critical than ever that this archive data not be exposed, as it is often the last line of recovery against such attacks. This document provides the reader some best practices to further secure Rubrik archive data stored in Amazon S3.

SHARED RESPONSIBILITY MODEL

Before we dive into security recommendations, it’s important to ensure that the concept of shared responsibility is fully understood. AWS has created the concept of a shared responsibility model to reinforce the fact that a customer has to take responsibility for the security of their own data. AWS can only do so much to protect their customers, but it is also up to the individual customer to cover the rest of the responsibilities. AWS breaks down the demarcation of responsibilities in the below diagram:

AWS cannot fully control what a customer does around the security of their own data in the cloud. What this means, practically speaking, is that it is ultimately the customer’s responsibility to protect their own applications and data that is running within AWS. Also a good protection scheme requires best practices and constant monitoring and maintenance of those practices. Further details about the AWS shared responsibility model can be found on AWS’s Shared Responsibility Page.

The concept of shared responsibility also applies to Rubrik and the CloudOut feature. Once the data is archived out to AWS S3, it is out of Rubrik’s hands at that point. Rubrik does all that it can to encrypt and obscure the data once it leaves the system. However if an attacker had control of enough information and somehow was able to access the S3 bucket externally, they could then use a copy of CDM to recover the contents of that data back to its original form or delete it entirely. It is still imperative that the customer protect the archival information in the bucket to the best of their ability through best practices and continual monitoring and maintenance.

1 https://www.rubrik.com/blog/ransomware-recovery-immutable-backup-architecture/  
2 https://aws.amazon.com/compliance/shared-responsibility-model/
HARDENING RECOMMENDATIONS

DISCLAIMER: Some of the recommendations mentioned below (ie. KMS, CloudTrail, etc..) incur additional costs for usage. We recommend weighing the cost benefits before moving forward with any feature enablement.

ADHERE TO THE PRINCIPLE OF LEAST PRIVILEGE:

The Principle of Least Privilege is the idea and practice of providing the absolute minimum privileges needed for a user or process in order to perform the required actions. This is a best practice that should be extended out to AWS S3 as well in order to give you the best possible protection. This involves the following:

1. Pre-Create S3 bucket. Do not have Rubrik auto-create the bucket during the archival location creation step. When Rubrik auto-creates the bucket, the IAM user that Rubrik uses has the “Create Bucket” privilege which is more than what is necessary. Rubrik will also not enable the Block Public Access feature which can offer an extra layer of protection for the S3 bucket.

2. Create a service account in IAM that is specifically dedicated for Rubrik CloudOut operations. Additionally, it is a best practice to limit this account access to only the buckets that are necessary for a single Rubrik archival location. If a new archival location is needed, it is recommended to create a new dedicated IAM user for this location instead of re-using IAM accounts for previously created archival locations. This step is designed to limit the exposure to a single location in the event that credentials for the account are compromised.

3. Create an IAM security policy for the bucket created in step 1 that limits access to just the user created in step 2. This involves just assigning the 10 required privileges that CloudOut needs to operate and also ensuring that the “Create Bucket” privilege is not present as mentioned in step 1. These privileges include the following in a sample policy (Note: svc-rubrik-cloudout in the policy below is the name of the bucket that was created to be the target of this policy as an example.):

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:ListBucketMultipartUploads",
            "s3:AbortMultipartUpload",
            "s3:RestoreObject",
            "s3:HeadBucket",
            "s3:ListBucket",
            "s3:DeleteObject",
            "s3:GetBucketLocation",
            "s3:ListMultipartUploadParts"
         ],
         "Resource": [
            "arn:aws:s3:::svc-rubrik-cloudout",
            "arn:aws:s3:::svc-rubrik-cloudout/*"
         ]
      }
   ]
}
```
4. One can also take the bucket policy an extra step further by limiting which IP addresses or VPC endpoints can access the bucket that is used for archival. If Cloud Cluster is being used, this policy can limit access to just the VPC where Cloud Cluster is running. If a physical cluster is being used, IP addresses can be used here to provide an extra security layer. If enhancing the bucket policy to provide these extra security measures is desired, more information can be found on AWS’s website³.

5. For further protection, one can create a S3 bucket policy that prevents any other user from accessing the designated bucket. Even after we created a least privilege policy in the previous step and assigned it to our dedicated service account, this same policy can be inadvertently assigned to another user. The below is a sample policy that denies access to other accounts outside of the dedicated service account we created:

```json
{
  "Id": "Policy1586381269739",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1586381248743",
      "Action": ["*"],
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::svc-rubrik-cloudout", "arn:aws:s3:::svc-rubrik-cloudout/*"],
      "NotPrincipal": {
        "AWS": ["arn:aws:iam::123456789000:user/svc-rubrik-cloudout"]
      }
    }
  ]
}
```

**ENCRYPTION AND KEY MANAGEMENT:**

Rubrik CloudOut does client side encryption of the archival data in one of two ways. One method is through AWS KMS integration and the other is through using a user-created RSA key manually. There are a few key reasons why it makes sense to use AWS KMS integration over a user-created key:

1. For organizations under strict compliance regulations, utilizing AWS KMS can ensure you adhere to the compliance standards that are put in place. For some organizations, utilizing anything else might not even be an option.

2. AWS KMS doesn’t reveal the encryption keys (also known as “backing keys”) that are actually used to encrypt the data to the user. The only known entity to the user is the ID of a CMK (Customer Master Key). This ensures added protection from attackers potentially stealing the key or keys used to encrypt the data. Key Aliases can also be used to further abstract the underlying CMK also to provide a further security measure⁴.

³ https://aws.amazon.com/premiumsupport/knowledge-center/block-s3-traffic-vpc-ip/
⁴ https://docs.aws.amazon.com/kms/latest/developerguide/programming-aliases.html
3. AWS KMS has the added benefit of automatic key rotation as well. For AWS managed keys, this is on by default and cannot be modified. Please note though that for customer managed keys, this is disabled by default and has to be enabled to be activated.

4. AWS KMS is integrated with IAM in a granular way such that you can specify a separation of duties between administration for a particular CMK and the actual usage of the CMK.

**BEST PRACTICE NOTE:** When creating a CMK in AWS KMS, it's important to remember the least privilege principle. Ensure that the account utilizing the archival bucket is only specified in the Key Usage permissions and not in the Key Administrative permissions. The below is a sample KMS policy that clearly defines different IAM accounts for administrative and usage:

```json
{
    "Id": "key-consolepolicy-3",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789000:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow access for Key Administrators",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789000:user/admin.user"
            },
            "Action": [
                "kms:Create*",
                "kms:Describe*",
                "kms:Enable*",
                "kms:List*",
                "kms:Put*",
                "kms:Update*",
                "kms:Revoke*",
                "kms:Disable*",
                "kms:Get*",
                "kms:Delete*",
                "kms:TagResource",
                "kms:UntagResource",
                "kms:ScheduleKeyDeletion",
                "kms:CancelKeyDeletion"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
        }
    ]
}
```
"Principal": {
  "AWS": "arn:aws:iam::123456789000:user/svc-rubrik-cloudout"
},
"Action": [
  "kms:Encrypt",
  "kms:Decrypt",
  "kms:ReEncrypt*",
  "kms:GenerateDataKey*",
  "kms:DescribeKey"
],
"Resource": "*"
},
{
  "Sid": "Allow attachment of persistent resources",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::123456789000:user/svc-rubrik-cloudout"
  },
  "Action": [
    "kms:CreateGrant",
    "kms:ListGrants",
    "kms:RevokeGrant"
  ],
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms:GrantIsForAWSResource": "true"
    }
  }
}
]
Note: Additional information around KMS best practices can be found in AWS’s KMS Best Practices Guide.

MONITORING AND MAINTENANCE:

Since security is an ongoing exercise, it’s always a recommendation to monitor and audit your S3 bucket activity on a regular basis. Even though you can adhere to a least privilege principle, a bad actor such as a disgruntled admin can still come along and potentially modify the permissions. One way to do this effectively is by enabling CloudTrail monitoring on the S3 bucket that’s used as an archive location. Server Access Logging can also be enabled on the bucket for a more log/text based delivery. Both log delivery mechanisms can be exported out to external frameworks such as Splunk or Elk for ease of monitoring instead of reading them individually out of the AWS console.

Here is an excerpt from CloudTrail logs where it shows an unauthorized account snooping around in one of the archive buckets. In this particular scenario, only the `svc-rubrik-cloudout` should have access to the bucket:

```json
{
  "eventVersion": "1.07",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "XXXXXXXXXXXXXXXXXX",
    "arn": "arn:aws:iam::123456789000:user/badactor1",
    "accountId": "123456789000",
    "accessKeyId": "XXXXXXXXXXXXXXXXXX",
    "userName": "badactor1"
  },
  "eventTime": "2020-04-13T19:12:12Z",
  "eventSource": "s3.amazonaws.com",
  "eventName": "GetObject",
  "userAgent": "[S3 Browser 8.6.7 https://s3browser.com]",
  "requestParameters": {
    "bucketName": "svc-rubrik-securebucket",
    "Host": "svc-rubrik-securebucket.s3.us-west-1.amazonaws.com",
    "key": "rubrik_encryption_key_check.txt"
  }
}
```

It is important to continually monitor the activity within the buckets especially if it’s locked down to a limited security specification. Maintenance should then be performed accordingly to remediate the access breach.
Below is an example of what could happen if actions are taken soon enough, or if the activity isn’t caught altogether:

```json
{
  "eventVersion": "1.07",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "XXXXXXXXXXXXXXXXXXXX",
    "arn": "arn:aws:iam::123456789000:user/badactor1",
    "accountId": "123456789000",
    "accessKeyId": "XXXXXXXXXXXXXXXXXXXX",
    "userName": "badactor1"
  },
  "eventTime": "2020-04-14T16:45:19Z",
  "eventSource": "s3.amazonaws.com",
  "eventName": "DeleteObject",
  "userAgent": "[S3 Browser 8.6.7 https://s3browser.com]",
  "requestParameters": {
    "bucketName": "svc-rubrik-securebucket",
    "Host": "svc-rubrik-securebucket.s3.us-west-1.amazonaws.com",
    "key": "snappables/"
  }
}
```

CloudTrail further logs the unauthorized account deleting the “snappables” key within the archive bucket almost a day later. In this particular example, the unauthorized access occurred due to a bucket policy not being applied properly therefore allowing access from multiple service principal(s) outside of the service account. Even in the worst case scenario however, CloudTrail and other logging can provide a forensic audit of the damaging activity that occurred that can be traced back.

Also if KMS is being used to manage encryption keys, it is recommended to also use CloudTrail to monitor the activity around CMK management. Even if one has separated the duties between CMK management and usage by assigning separate service principals to each, it is recommended to keep an eye on the activity. The screenshot below shows an example of an admin level principal scheduling a CMK for deletion prematurely that is currently in use for CloudOut:
CONCLUSION

While Rubrik does all that it can to secure the data while it’s local to the cluster, once it leaves via CloudOut archival operations it now resides in a realm that is outside of the control of Rubrik. It then becomes critical for the customer to ensure that this archival data is protected by ensuring best practices are followed. It’s also imperative to not let your guard down as well once security practices are put into place by applying monitoring and maintenance procedures. By following these recommendations mentioned above, one should be able to extend optimal data security out into Amazon S3.

VERSION HISTORY

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>April 2020</td>
<td>Initial Release</td>
</tr>
</tbody>
</table>