



Configuring Amazon S3 security settings and access controls



Getting started at an AWS hosted workshop

▼ S3 Security Best Practices

▼ Prepare Your Lab

- Attach IAM Role to EC2 Instance
- Connect to the EC2 Instance
- Bucket Name

▼ Lab 1 - S3 Security Exercises

- Require HTTPS
- Require SSE-KMS Encryption
- Restrict Access to an S3 VPC Endpoint
- Use AWS Config Rules to Detect a Public Bucket
- Use Amazon Access Analyzer for S3

▼ Lab 2 - S3 Access Grants

- S3 Access Grants Lab - Initial Setup
- Configure S3 Access Grants for IAM user

▼ Lab 3 - Enabling Malware Protection for S3 by using GuardDuty

- Enabling Malware Protection for S3 for your bucket
- Testing GuardDuty Malware with an object.

▼ Lab 4 - S3 Access Control Lists

Block Public ACLs

- Configure S3 Block Public Access
- Disable S3 ACLs
- Finding S3 access control lists with S3 Inventory
- Use Amazon Athena to query CloudTrail logs and identify S3 requests that depend on ACLs

Lab Summary

[Configuring Amazon S3 security settings and access controls](#) > [S3 Security Best Practices](#) > [Lab 4 - S3 Access Control](#)

Block Public ACLs

In this exercise, we will demonstrate the use of block public ACLs. As mentioned, as of April 2023 Amazon S3 now automatically enables S3 Block Public Access and disables S3 access control lists (ACLs) for all new S3 buckets in all AWS Regions. Due to this we will disable both of the aforementioned features before we start.

From the AWS console in the top search bar, search and select S3.

- Click the bucket name starting with `sid-security-xxxxxxx`.
- Click on the **Permissions** tab.
- Under **Block public access** (bucket settings) click **Edit**.
- Uncheck the **Block all public access** box.
- Click **Save changes** and confirm the changes by typing "confirm"

This should have already been done in the previous section.

- Next scroll down to **Object Ownership** and click **Edit**.
- Click the box labeled **ACLs enabled** and the "I acknowledge that ACLs will be restored." box below it.
- Keep the rest of the settings and click **Save changes**.

Stay on the **Permissions** tab.

- Under **Bucket Policy** click **Edit**. Copy the bucket policy below and paste into the **Bucket Policy Editor**.

```
{
  "Id": "S3-Security-allow-if-private-acl",
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": "*",
    "Action": [
      "s3:PutObject",
      "s3:PutObjectAcl"
    ],
    "Resource": "arn:aws:s3:::BUCKET_NAME/*",
    "Condition": {
      "StringEquals": {
        "s3:x-amz-acl": "private"
      }
    }
  }]
}
```



Replace `BUCKET_NAME` with the bucket name you copied to your text editor and click **Save changes**.

Make sure you keep the `/*` at the end of the bucket name.

Open an SSH session to the `SID-security-instance` using **EC2 Instance Connect** if it is not already open. Run the following command.

```
aws s3api put-object --key text01 --body textfile --bucket ${bucket}
```



The request should succeed since the default for an object ACL is private.

```
ec2-user@storage-workshop ~]$ aws s3api put-object --key text01 --body textfile --bucket ${bucket}
{
  "SSEKMSKeyId": "arn:aws:kms:us-west-2:121049687582:key/7f76ab8c-d042-4392-afd8-020e04f9439a",
  "ETag": "\"f39bb2ed7b4776e409bea6ac19e188db\"",
  "ServerSideEncryption": "aws:kms"
}
ec2-user@storage-workshop ~]$
```

Now run the following command in your SSH session.

```
aws s3api put-object --key text01 --body textfile --acl public-read --bucket ${bucket}
```

This command also succeeded, but it is not the behavior one would expect. Why do we still have access?

```
ec2-user@storage-workshop ~]$ aws s3api put-object --key text01 --body textfile --acl public-read --bucket ${bucket}
{
  "SSEKMSKeyId": "arn:aws:kms:us-west-2:121049687582:key/7f76ab8c-d042-4392-afd8-020e04f9439a",
  "ETag": "\"77e0da40129c5bae0b7854cd071078ca\"",
  "ServerSideEncryption": "aws:kms"
}
ec2-user@storage-workshop ~]$
```

The current bucket policy allows ACLs that are private but doesn't DENY anything. It is important to write policies that prevent actions, not allow it when trying to restrict actions against a bucket. The current bucket policy also allows Public access to the bucket unintentionally due to the principal being a wildcard.

Replace the existing policy with the following policy. Copy and paste your bucket name into the policy. Notice this is a deny policy.

```
{
  "Id": "S3-Security-Deny-if-public",
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "Principal": "*",
    "Action": [
      "s3:PutObject",
      "s3:PutObjectAcl"
    ],
    "Resource": "arn:aws:s3::BUCKET_NAME/*",
    "Condition": {
      "StringEquals": {
        "s3:x-amz-acl": [
          "public-read",
          "public-read-write",
          "authenticated-read"
        ]
      }
    }
  }]
}
```

 Make sure you keep the /* at the end of the bucket name.

Click **Save** changes

Run the following command in the SSH session.

```
aws s3api put-object --key text01 --body textfile --bucket ${bucket}
```

The request should succeed since the default for an object ACL is private.


The bucket policy explicitly denies applying public read and public write ACLs, but since private ACLs are not denied then applying private ACLs is permitted.

Now run the following command in the SSH session to test if we are able to create a new object in the bucket with a public read ACL.

```
aws s3api put-object --key text01 --body textfile --acl public-read --bucket ${bucket}
```



The request fails as the bucket policy now restricts public-read ACL.

For more information on condition key `s3:x-amz-acl`, see the below link for a list of canned ACLs and predefined grants. <https://docs.aws.amazon.com/AmazonS3/latest/userguide/acl-overview.html#canned-acl> 

[Previous](#)[Next](#)