



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

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

Lab Summary

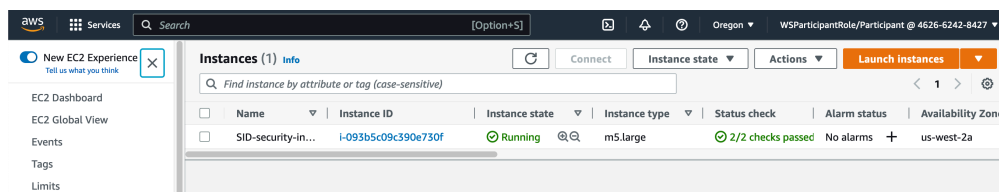
[Configuring Amazon S3 security settings and access controls](#) > [S3 Security Best Practices](#) > [Lab 3 - Enabling Malware](#)

Testing GuardDuty Malware with an object.

Now that we have GuardDuty configured with our S3 bucket, we will create an object and test whether GuardDuty tags the object correctly.

From the AWS console in the top search bar, search and select EC2.

Connect to the EC2 Instance using EC2 Instance Connect



Right click the SID-security-instance and click connect.

Select the EC2 Instance Connect (browser-based SSH connection).

Set the User name to ec2-user if not already. It should look like the following:

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID

i-02f8a19f550d6ecfa (SID-security-instance)

Connection Type

☒ Connect using EC2 Instance Connect

Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

☐ Connect using EC2 Instance Connect Endpoint

Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IP address

35.88.123.195

User name

Enter the user name defined in the AMI used to launch the instance. If you didn't define a custom user name, use the default user name, ec2-user.

ec2-user

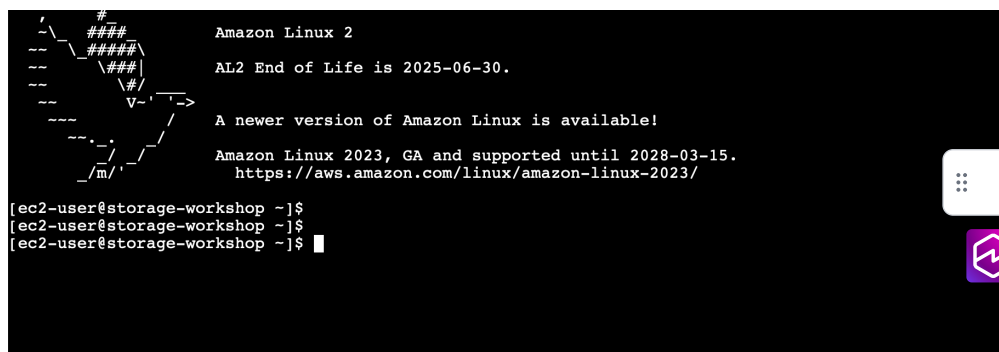
Note:

In most cases, the default user name, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel

Connect

If your screen looks correct, click Connect. Your screen should look similar to the following:



This is where we will create two test objects:

- Let's create a test object by using the following code:

```
echo 'This is a test object with text.' >> testobject.txt
```



- Now let's create another test object but this time we will use a string:

```
echo 'X50!P#@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*' >> testobject2.txt
```



- Let's ensure those two files were created correctly, as we will use them for our test.

```
cat testobject.txt testobject2.txt
```



- Your screen should have two that correspond to each file, like this:

```
[ec2-user@storage-workshop ~]$
[ec2-user@storage-workshop ~]$
[ec2-user@storage-workshop ~]$
[ec2-user@storage-workshop ~]$ cat testobject.txt testobject2.txt
This is a test object with text.
X50!P#@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
[ec2-user@storage-workshop ~]$
```

Now we can create a test by uploading each of those files to S3. *Note* Make sure to change the name of the s3 bucket to the one in your account.

```
aws s3 cp testobject.txt s3://{bucket}
aws s3 cp testobject2.txt s3://{bucket}
```



Let's check whether the object gets tagged appropriately:

- First we can check the testobject.txt:

```
aws s3api get-object-tagging --key testobject.txt --bucket ${bucket}
```



- Then the testobject2.txt:

```
aws s3api get-object-tagging --key testobject2.txt --bucket ${bucket}
```



Your screen should look something like this:

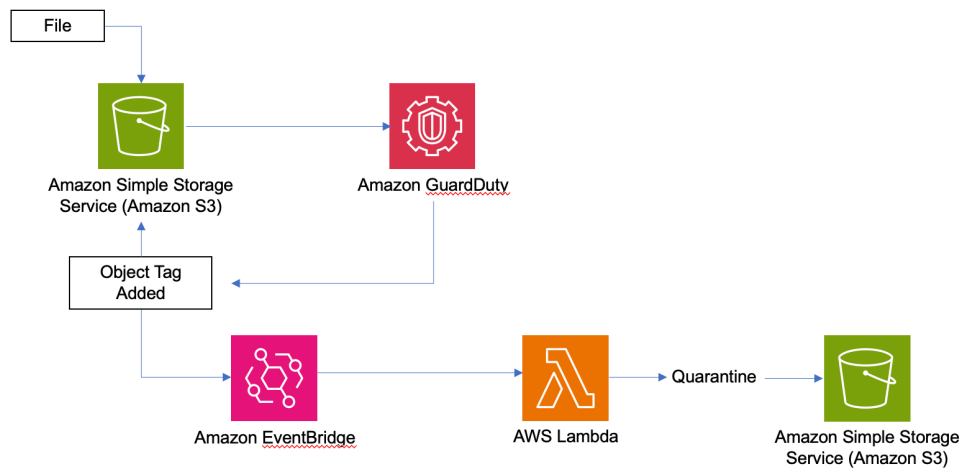
```
[ec2-user@storage-workshop ~]$
[ec2-user@storage-workshop ~]$
[ec2-user@storage-workshop ~]$ aws s3api get-object-tagging --key testobject.txt --bucket sid-security-4f6be270-a10a-11ef-a6d7-0200dca164e3
{
  "TagSet": [
    {
      "Value": "NO_THREATS_FOUND",
      "Key": "GuardDutyMalwareScanStatus"
    }
  ]
}
[ec2-user@storage-workshop ~]$
[ec2-user@storage-workshop ~]$ aws s3api get-object-tagging --key testobject2.txt --bucket sid-security-4f6be270-a10a-11ef-a6d7-0200dca164e3
{
  "TagSet": [
    {
      "Value": "THREATS_FOUND",
      "Key": "GuardDutyMalwareScanStatus"
    }
  ]
}
[ec2-user@storage-workshop ~]$
```



Take note of how the test file we created with the fake malware signature was tagged appropriately but the other object was not.

- Using the tags on the object is a great way to take action on the objects that are tagged as a threat and quarantine them or delete them, depending on the environment and use case. For example you

could do something like:



✓ **Lab complete**

Congratulations you have completed the GuardDuty Malware Protection for S3 lab. In this lab, you learned how to configure GuardDuty Malware Protection to detect the presence of threats within S3 buckets.

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