

7

Reconnaissance - Identifying Vulnerable S3 Buckets

Simple Storage Service (S3) buckets are one of the most popular attack surfaces for AWS infrastructures, and they're the most prone to hacking attacks.

This chapter explains the concept of AWS S3 buckets, what they're used for, and how to set them up and access them. However, the main focus for this chapter is on the various S3 bucket permissions, the different ways of identifying poorly configured or permissive buckets, as well as connecting to these buckets. Finally, we will focus on automated approaches to identifying vulnerable S3 buckets in multiple regions based on domain and subdomain names, and probing their permissions to find potentially vulnerable buckets.

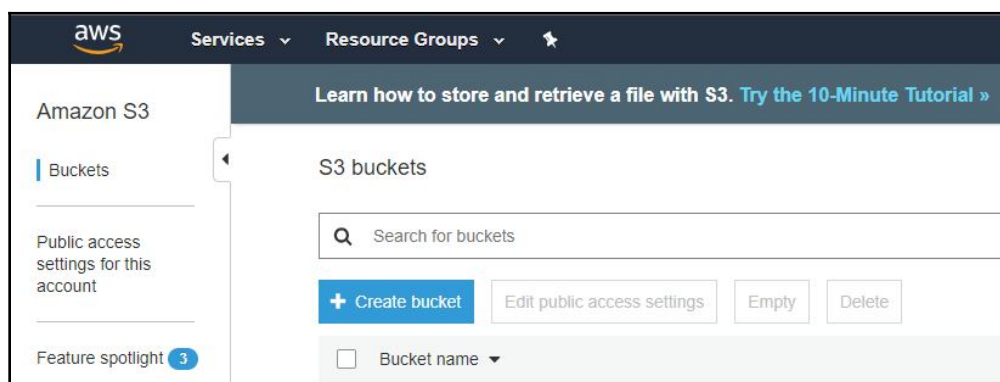
In this chapter, we will cover the following topics:

- Setting up our first S3 bucket
- Exploring AWS S3 permissions and the access API
- Reading and writing from a vulnerable S3 bucket

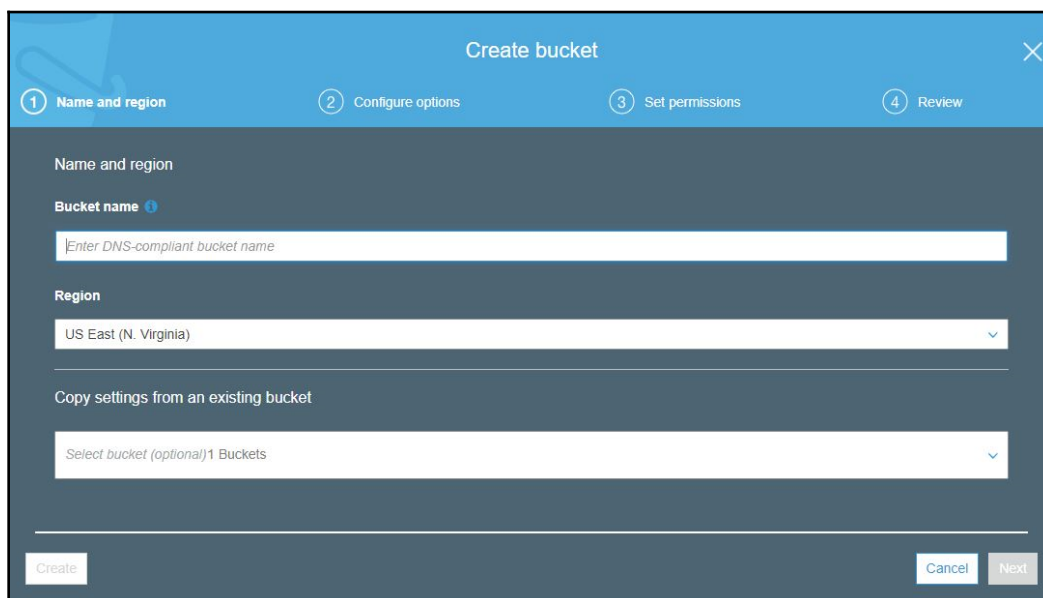
Setting up your first S3 bucket

We will start by heading over to the S3 home page at <https://s3.console.aws.amazon.com/s3/>:

1. On the S3 home page, click on **Create bucket**:

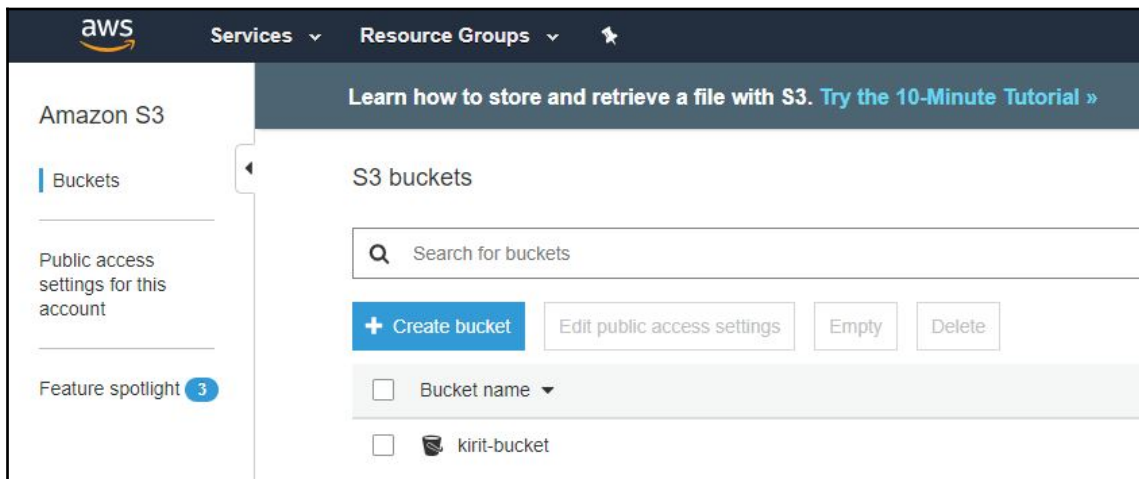


2. In the next page, assign your bucket a name:



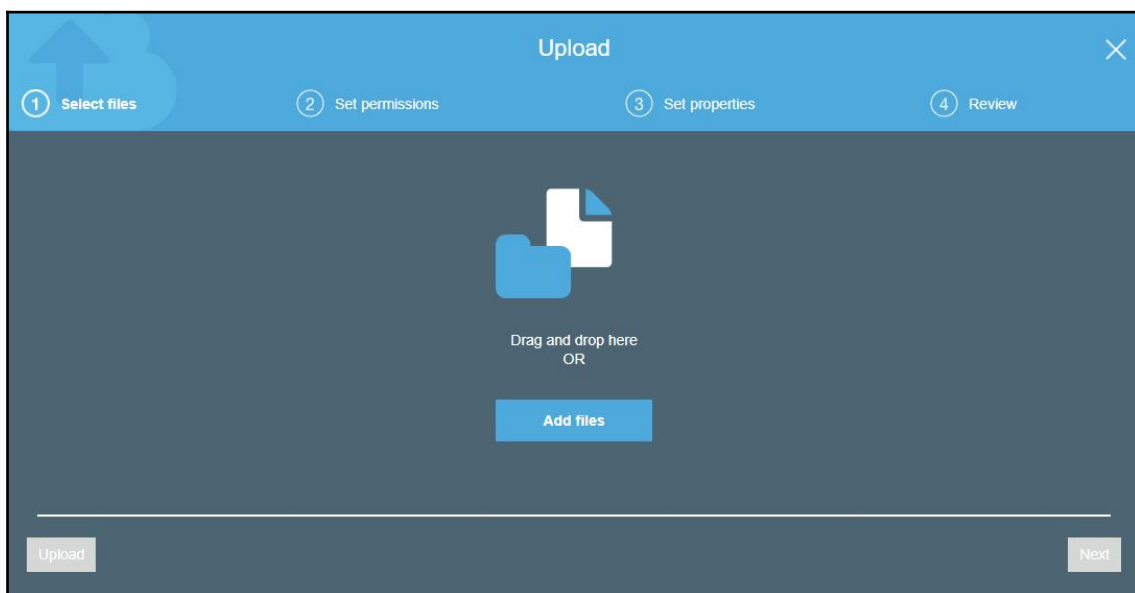
When assigning the name of the bucket, you must follow these guidelines:

- Use a unique and **Domain Name System** (DNS)-compliant bucket name for your S3 bucket.
 - Bucket names must be a minimum of 3 characters and a maximum of 63 characters.
 - Uppercase characters or underscores are not allowed.
 - Bucket names can either start with a lowercase letter or a number.
 - Bucket names can contain lowercase letters, numbers, and hyphens. The bucket name can also be separated based on labels using the (.) character.
 - Do not format bucket names in the form of an IP address (for example, 172.16.1.3).
3. You can choose the geographic region if you wish to; we are naming our bucket `kirit-bucket`.
 4. Click on **Create bucket** and your bucket will be created:



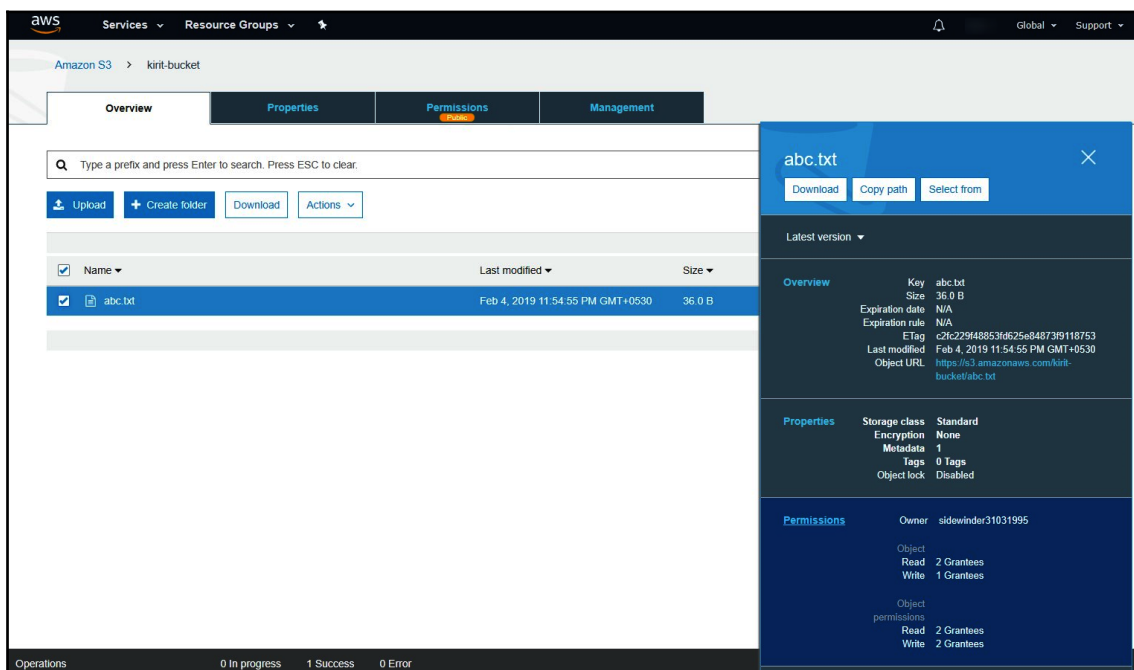
Once the bucket is up and running, you should be able to upload objects to the bucket. In case you are wondering what an object is, it can be any file, such as image files, music files, video files, or documents.

5. To upload an object, click on the bucket and select **Upload**:



A file browser will open and you can upload any file that you want.

6. To download an object, simply tick the checkbox of the object, and then choose **Download**:



S3 permissions and the access API

S3 buckets have two permission systems. The first is **access control policies (ACPs)**, which are primarily used by the web UI. This is a simplified permission system that provides a layer of abstraction for the other permission system. Alternatively, we have **IAM access policies**, which are JSON objects that give you a granular view of permissions.

Permissions apply either to a bucket or an object. Bucket permissions are like the master key; in order to provide someone access to an object, you need to provide them access to a bucket first, and then the individual objects themselves.

S3 bucket objects can be accessed from the WebGUI, as we saw earlier. Otherwise, they can be accessed from the AWS command-line interface (CLI) using the `aws s3` cmdlet. You can use it to upload, download, or delete bucket objects.

In order to upload and download objects using the AWS CLI, we can take the following approach:

1. Start by installing `awscli`:

```
sudo apt install awscli
```

2. Configure `awscli` with the new user credential. For this, we will need the access key ID and the secret access key. To get these, follow this procedure:
 1. Log in to your AWS Management Console
 2. Click on your username at the top-right of the page
 3. Click on the **Security Credentials** link from the drop-down menu
 4. Find the **Access Credentials** section, and copy the latest access key ID
 5. Click on the **Show link** in the same row, and copy the secret access key
3. Once you have acquired these, issue the following command:

```
aws configure
```

Enter your access key ID and secret access key. Remember to not make this public to ensure your accounts are safe. You may leave your default region and output format set to none.

4. Once your account has been set up, it is very easy to access the contents of the S3 bucket:

```
aws s3 ls s3://kirit-bucket
```

`kirit-bucket` in the preceding code will be replaced by your bucket name.

5. If you want to traverse directories inside a bucket, simply put `/` followed by the directory named listed from the preceding output, for example, if we have a folder named `new`:

```
aws s3 ls s3://kirit-bucket/new
```

6. To upload a file to the S3 bucket, issue the `cp` cmdlet, followed by the filename and the destination bucket with full file path:

```
aws s3 cp abc.txt s3://kirit-bucket/new/abc.txt
```

7. To delete a file on the S3 bucket, issue the `rm` cmdlet followed by the full file path:

```
aws s3 rm s3://kirit-bucket/new/abc.txt
```

ACPs/ACLs

The idea of **access control lists (ACLs)** is very similar to the firewall rules that can be used to allow access to an S3 bucket. Each S3 bucket has an ACL attached to it. These ACLs can be configured to provide an AWS account or group access to an S3 bucket.

There are four main types of ACLs:

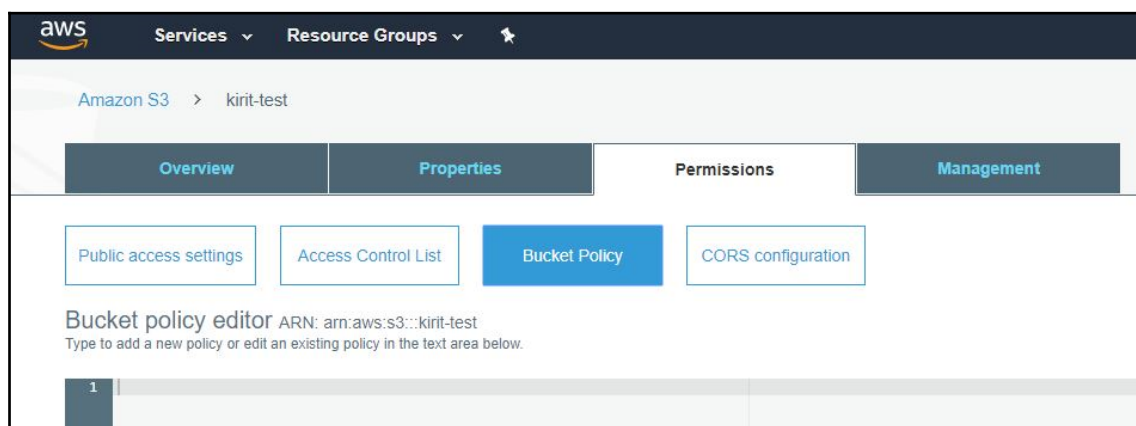
- **read**: An authenticated user with read permissions will be able to view filenames, size, and the last modified information of an object within a bucket. They may also download any object that they have access to.
- **write**: An authenticated user has the permission to read as well as delete objects. A user may also be able to delete objects they don't have permissions to; additionally, they can upload new objects.
- **read-acp**: An authenticated user can view the ACLs of any bucket or object they have access to.
- **write-acp**: An authenticated user can modify the ACL of any bucket or object they have access to.

An object can only have a maximum of 20 policies in a combination of the preceding four types for a specific grantee. A grantee is referred to any individual AWS account (that is, email address) or a predefined group. IAM accounts cannot be considered as a grantee.

Bucket policies

Each S3 bucket has bucket policies attached to it that can be applied to both the bucket and the objects inside it. In case of multiple buckets, the policies can be easily replicated. Policies can be applied to individual folders by specifying a resource such as `"data/*"`. This will apply the policy to each object in a folder.

You can add a policy to your S3 bucket using the web UI. The action is under the **Permissions** tab of the bucket **Properties** page:



Next, we will see how bucket access can be configured for IAM users.

IAM user policies

In order to provide S3 access to individual IAM accounts, we can use IAM user policies. They are a very easy way to provide restricted access to any IAM account.

IAM user policies come in handy when an ACL permission must be applied to one specific IAM account. If you are wondering whether to use IAM or a bucket policy, a simple rule of thumb is to determine whether the permissions are for specific users across a number of buckets, or if you have multiple users, each needing their own set of permissions. In such a scenario, IAM policies are much better suited than bucket policies, as bucket policies are limited to only 20 KB.

Access policies

Access policies are fine-grained permissions that describe permissions granted to any user on an object or bucket. They are described in JSON format and can be divided into three main sections: "Statement", "Action", and "Resource".

Here is an example of a bucket policy in JSON:

```
{
  "Version": "2008-02-27",
  "Statement": [
    {
      "Sid": "Statement",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::Account-ID:user/kirit"
      },
      "Action": [
        "s3:GetBucketLocation",
        "s3:ListBucket",
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3::kirit-bucket"
      ]
    }
  ]
}
```

The JSON object has three main parts. First, within the "Statement" section, we can see there are two points to note – "Effect": "Allow", and the "Principal" section containing "AWS": "arn:aws:iam::Account-ID:user/kirit". This essentially means that the "kirit" user account is being granted permissions to an object.

Second, is the "Action" section, which describes what permissions are being allowed to the user. We can see the user is allowed to list objects inside the "s3:ListBucket" bucket, and download objects from the "s3:GetObject" bucket.

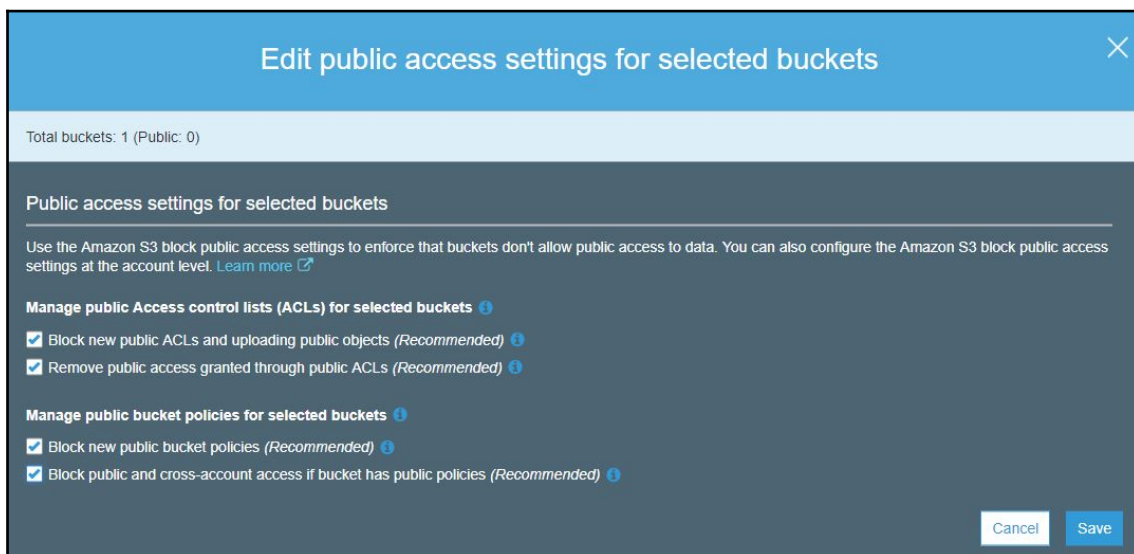
Finally, the Resource part describes on which resource the permissions are being granted. To put it all together, the policy summarizes to allow the kirit user account to GetBucketLocation, ListBucket, and GetObject under the bucket named kirit-bucket.

Creating a vulnerable S3 bucket

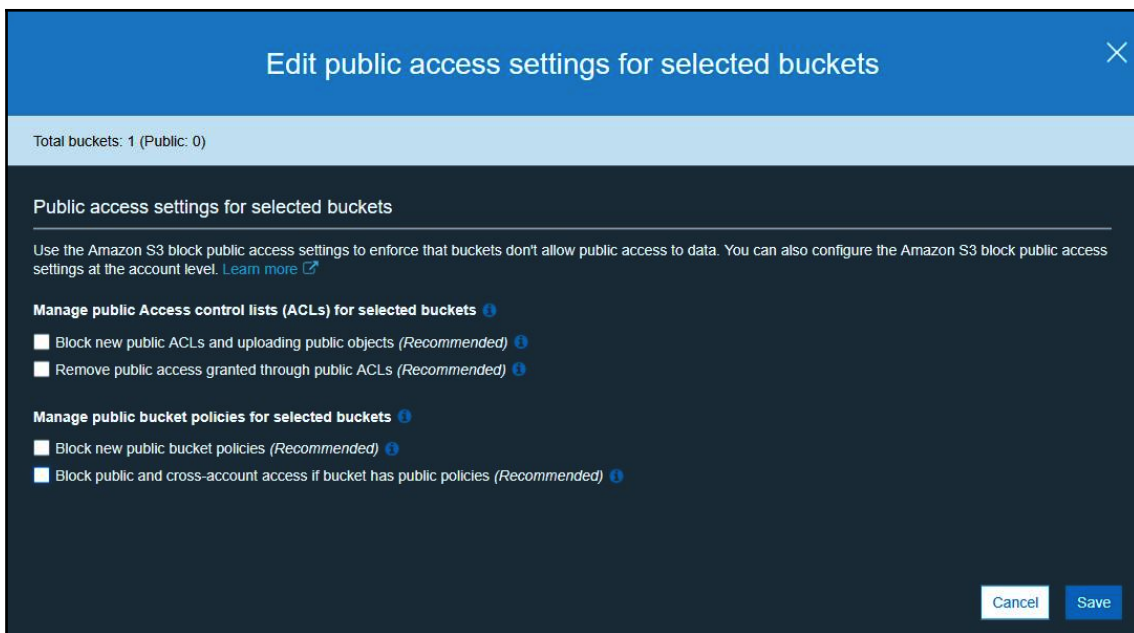
For our next exercise, we will try to read and write from a vulnerable S3 bucket that has been made public to the entire world. In order to do this, we will set up an S3 bucket and intentionally make it vulnerable by making it publicly readable and writeable.

We will start by heading over to the S3 home page (<https://s3.console.aws.amazon.com/s3/>) and creating a vulnerable bucket that is publicly accessible:

1. Create a new S3 bucket.
2. Once the bucket has been created, select the bucket and click on **Edit public access settings for selected buckets**:



3. Unselect all the checkboxes and click on **Save**. This is done in order to remove any access restrictions that have been enforced on a bucket:

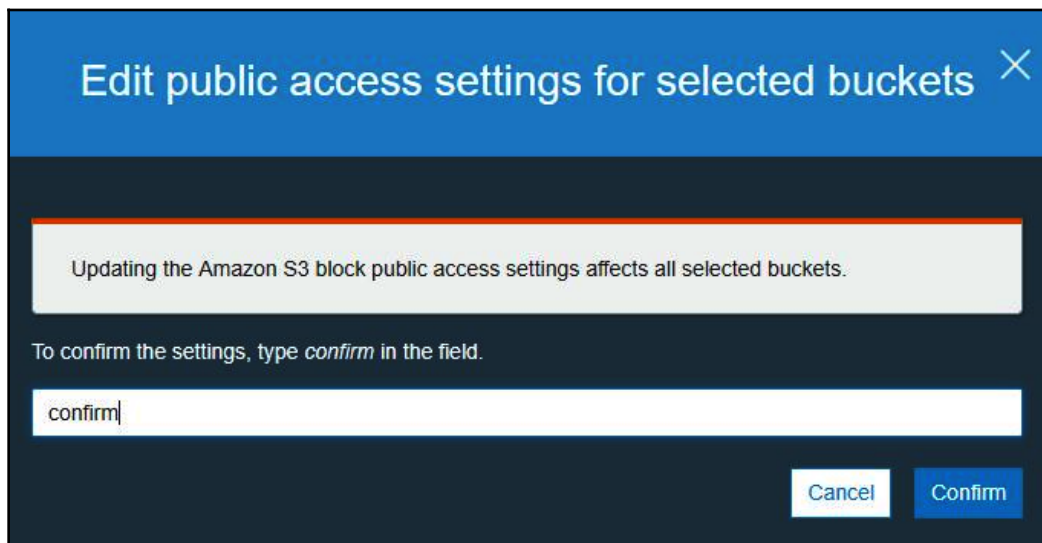


The screenshot shows the 'Edit public access settings for selected buckets' dialog. At the top, it says 'Total buckets: 1 (Public: 0)'. Below this, there's a section titled 'Public access settings for selected buckets' with a brief explanation: 'Use the Amazon S3 block public access settings to enforce that buckets don't allow public access to data. You can also configure the Amazon S3 block public access settings at the account level. [Learn more](#)'. There are two main sections for configuration, each with a header and a list of checkboxes:

- Manage public Access control lists (ACLs) for selected buckets**
 - ☐ Block new public ACLs and uploading public objects (Recommended)
 - ☐ Remove public access granted through public ACLs (Recommended)
- Manage public bucket policies for selected buckets**
 - ☐ Block new public bucket policies (Recommended)
 - ☐ Block public and cross-account access if bucket has public policies (Recommended)

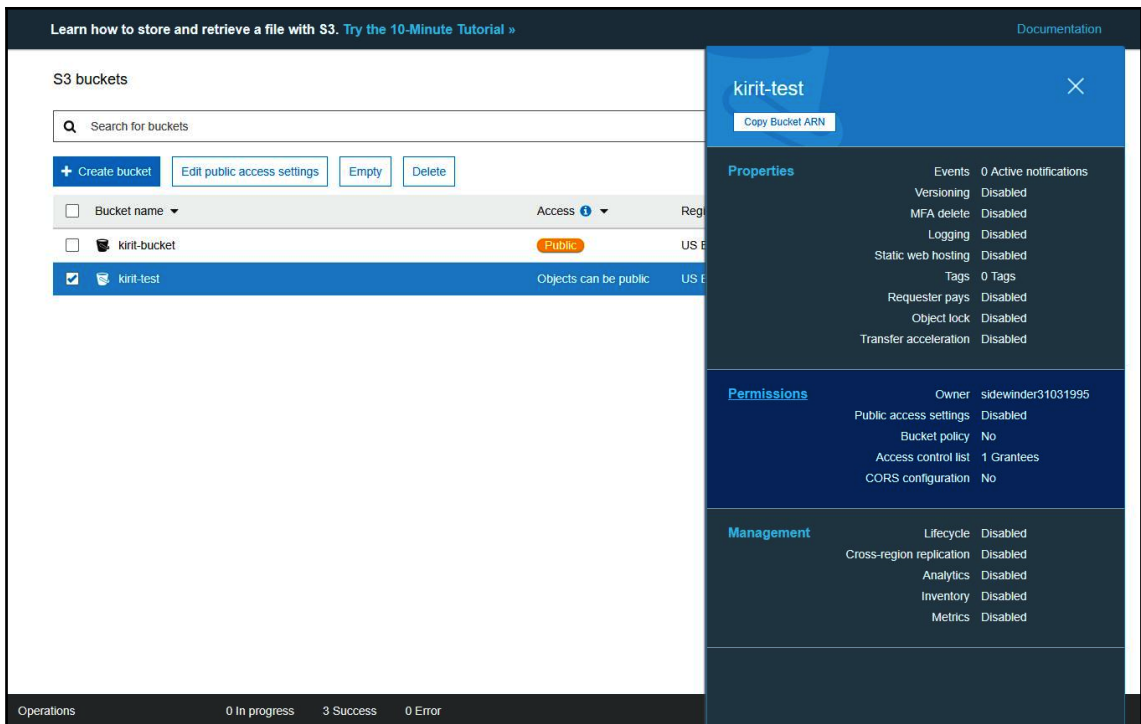
At the bottom right, there are 'Cancel' and 'Save' buttons.

4. AWS will ask you to confirm the changes; type `confirm` into the field and click on **Confirm**:



The screenshot shows the confirmation step of the 'Edit public access settings for selected buckets' dialog. It features a blue header with the title and a close button. Below the header, a light gray box contains the message: 'Updating the Amazon S3 block public access settings affects all selected buckets.' Below this, a text prompt says: 'To confirm the settings, type `confirm` in the field.' There is a text input field containing the text 'confirm'. At the bottom right, there are 'Cancel' and 'Confirm' buttons.

5. Click on the bucket, and then on the side panel, click on the **Permissions** tab:



The screenshot shows the AWS S3 console interface. On the left, a list of S3 buckets is displayed, with 'kirit-test' selected. The right-hand side panel is open, showing the 'kirit-test' bucket details. The 'Permissions' tab is active, displaying the following settings:

Property	Value
Owner	sidewinder31031995
Public access settings	Disabled
Bucket policy	No
Access control list	1 Grantees
CORS configuration	No

Below the 'Permissions' tab, the 'Management' tab is visible, showing the following settings:

Property	Value
Lifecycle	Disabled
Cross-region replication	Disabled
Analytics	Disabled
Inventory	Disabled
Metrics	Disabled

6. Go to **Access Control List**, and under **Public Access**, click on **Everyone**. A side panel will open; enable all the checkboxes. This tells AWS to allow public access to the bucket; this is what makes the bucket vulnerable:

Public access settings | **Access Control List** | Bucket Policy | CORS configuration

Everyone X

This bucket has public access
You have provided public access to this bucket. We highly recommend that you never grant any kind of public access to your S3 bucket.

Access for your AWS account root user

Account	List objects	Write objects	Read bucket permissions
<input type="radio"/> Your AWS account (owner) Canonical ID: 2149e9e27f837e1fa72104c850942c527859b0138208e9301f59b0118a9f	Yes	Yes	Yes

Access for other AWS accounts

[+ Add account](#) [Delete](#)

Account	List objects	Write objects	Read bucket permissions
---------	--------------	---------------	-------------------------

Public access

Group	List objects	Write objects	Read bucket permissions
<input checked="" type="radio"/> Everyone	Yes	Yes	Yes

S3 log delivery group

Group	List objects	Write objects	Read bucket permissions
-------	--------------	---------------	-------------------------

Access to the objects

- ☒ List objects
- ☒ Write objects

Access to this bucket's ACL

- ☒ Read bucket permissions
- ☒ Write bucket permissions

[Cancel](#) [Save](#)

7. Click on **Save** and the bucket will be made public.

Now that we have our vulnerable bucket, we can upload some objects to it and make them public; for example, we upload a small text file to the bucket as follows:

1. Create a small text document.
2. Enter your bucket and click on **Upload**:

aws Services Resource Groups

Amazon S3 > kirit-bucket

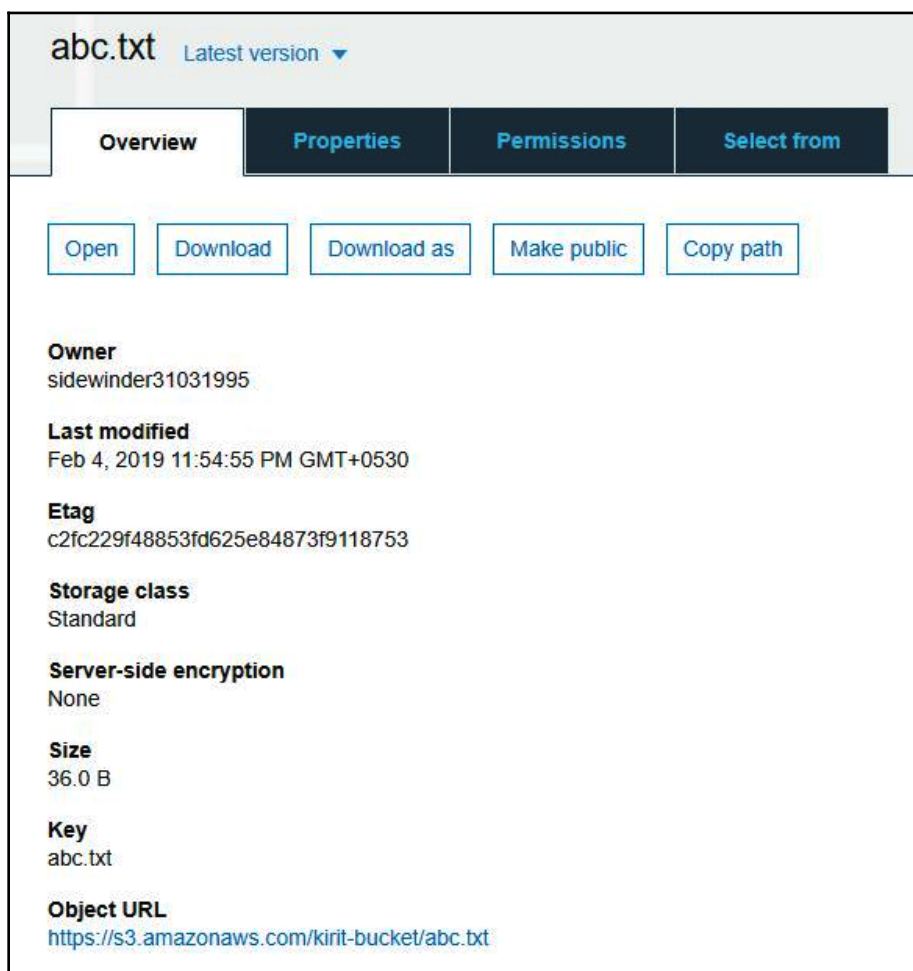
Overview **Properties**

🔍 Type a prefix and press Enter to search. Press ESC to clear.

[Upload](#) [+ Create folder](#) [Download](#) [Actions](#)

3. Select the file and upload it.

Once the file has been uploaded, click on the object, and you will receive an S3 URL to access the object from the outside. You can simply point your browser to the URL in order to access the bucket:



The **Object URL** link is located at the bottom of the page, as demonstrated in the preceding screenshot.

Our vulnerable S3 bucket has now been set up and made accessible to the public; anyone can read or write to this bucket.

In the next chapter, we will learn how to identify such vulnerable buckets and exfiltrate data using AWSBucketDump.

Summary

In this chapter, we have learned about what S3 buckets are, how to set up S3 buckets, and how access rights are granted on an S3 bucket. We learned about S3 permissions in detail, as well as how and where each kind of permission is applicable. We walked through how to set up the AWS CLI and access the S3 bucket via the CLI. We also learned about the kind of settings that can make an S3 bucket vulnerable. And finally, we set up our own vulnerable S3 bucket, which we will be using in the next chapter.

In the next chapter, we will learn how to exploit S3 buckets. We will look into the tools that are used to exploit a vulnerable S3 bucket. And, we will learn various post-exploitation techniques that we can apply after exploiting a vulnerable S3 bucket.

Further reading

- **Amazon S3 REST API Introduction:** <https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html>
- **Amazon S3 Examples:** <https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-examples.html>
- **Specifying Permissions in a Policy:** <https://docs.aws.amazon.com/AmazonS3/latest/dev/using-with-s3-actions.html>