TR-4921: Ransomware defense

StorageGRID solutions and resources

NetApp
November 21, 2025

Table of Contents

TR-4921: Ransomware defense

Protect StorageGRID S3 objects from ransomware

Learn about ransomware attacks and how to protect data with StorageGRID security best practices.

Ransomware attacks are on the rise. This document provides some recommendations on how to protect your object data on StorageGRID.

Ransomware today is the ever-present danger in the data center. Ransomware is designed to encrypt data and make it unusable by the users and applications that rely on it. Protection starts with the usual defenses of hardened networking and solid user security practices, and we need to follow through with data access security practices.

Ransomware is one of today's largest security threats. The NetApp StorageGRID team is working with our customers to keep ahead of these threats. With the use of object lock and versioning, you can protect against unwanted alterations and recover from malicious attacks. Data security is a multi-layer venture, with your object storage being just one part in your data center.

StorageGRID best practices

For StorageGRID, security best practices should include using HTTPS with signed certificates for both management and object access. Create dedicated user accounts for applications and individuals, and do not use the tenant root accounts for application access or user data access. In other words, follow the least privilege principle. Use security groups with defined Identity and Access Management (IAM) policies to govern user rights, and access accounts specific to the applications and users. With these measures in place, you still must ensure that your data is protected. In the case of Simple Storage Service (S3), when objects are modified to encrypt them, it is accomplished by an overwrite of the original object.

Methods of defense

The primary ransomware protection mechanism in the S3 API is to implement object lock. Not all applications are compatible with object lock, so there are two other options to protect your objects that are described in this report: replication to another bucket with versioning enabled and versioning with IAM policies.

Where to find additional information

To learn more about the information that is described in this document, review the following documents and/or websites:

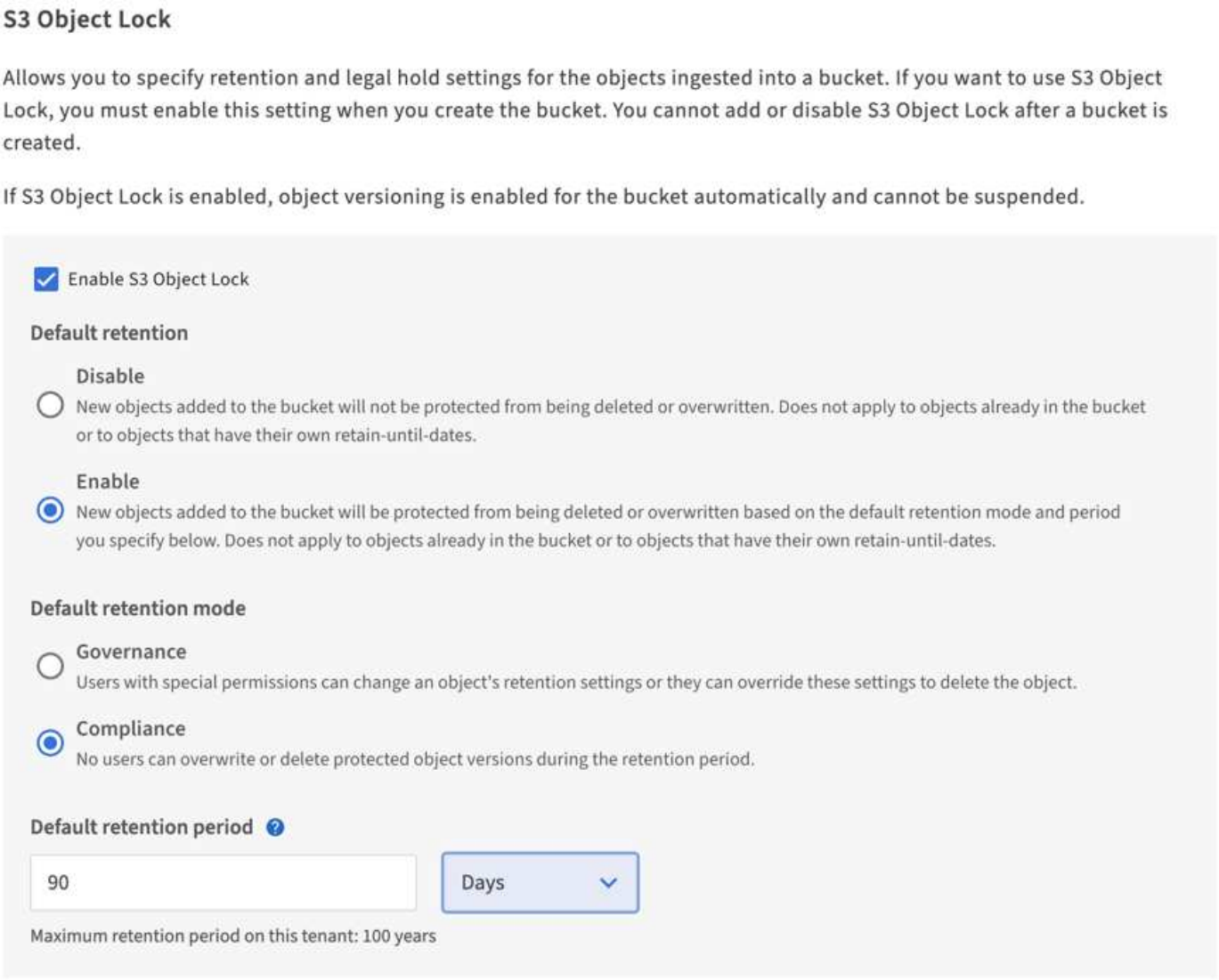
- NetApp StorageGRID Documentation Center
  https://docs.netapp.com/us-en/storagegrid/
- NetApp StorageGRID Enablement
  https://docs.netapp.com/us-en/storagegrid-enable/
- NetApp Product Documentation
  https://www.netapp.com/support-and-training/documentation/

Ransomware defense using object lock

Explore how object lock in StorageGRID provides a WORM model to prevent data deletion or overwrite, and how it meets regulatory requirements.

Object lock provides a WORM model to prevent objects from being deleted or overwritten. StorageGRID implementation of object lock is Cohasset assessed to help meet regulatory requirements, supporting legal hold, compliance mode, and governance mode for object retention, and default bucket retention policies. You must enable object lock as part of the bucket creation and versioning. A specific version of an object is locked, and if no version ID is defined, the retention is placed on the current version of the object. If the current version has the retention configured and an attempt is made to delete, modify, or overwrite the object, a new version is created with either a delete marker, or the new revision of the object as the current version, and the locked version is retained as a non-current version. For applications that are not yet compatible, you might still be able to make use of object lock and a default retention configuration placed on the bucket. After the configuration is defined, this applies an object retention to each new object put into the bucket. This works as long as the application is configured to not delete or overwrite the objects before the retention time has passed.

When creating a bucket in the Tenant management UI, you can enable object lock and configure a default retention mode and retention period. When configured this will set a minimum object lock retention on every object that is ingested to that bucket.



Here are a few examples using the object lock API:

Object lock legal hold is a simple on/off status applied to an object.

```
aws s3api put-object-legal-hold --bucket mybucket --key myfile.txt --legal
-hold Status=ON --endpoint-url https://s3.company.com
```

Setting the legal hold status does not return any value if successful, so it can be verified with a GET operation.

```
aws s3api get-object-legal-hold --bucket mybucket --key myfile.txt
--endpoint-url https://s3.company.com
{
    "LegalHold": {
        "Status": "ON"
    }
}
```

To turn legal hold off, apply the OFF status.

```
aws s3api put-object-legal-hold --bucket mybucket --key myfile.txt --legal
-hold Status=OFF --endpoint-url https://s3.company.com
aws s3api get-object-legal-hold --bucket mybucket --key myfile.txt
--endpoint-url https://s3.company.com
{
    "LegalHold": {
        "Status": "OFF"
    }
}
```

Setting the object retention is done with a retain until timestamp.

```
aws s3api put-object-retention --bucket mybucket --key myfile.txt
--retention '{"Mode":"COMPLIANCE", "RetainUntilDate": "2022-06-
10T16:00:00"}'  --endpoint-url https://s3.company.com
```

Again, there is no returned value on success, so you can verify the retention status similarly with a get call.

```
aws s3api get-object-retention --bucket mybucket --key myfile.txt
--endpoint-url https://s3.company.com
{
    "Retention": {
        "Mode": "COMPLIANCE",
        "RetainUntilDate": "2022-06-10T16:00:00+00:00"
    }
```

Putting a default retention on an object lock enabled bucket uses a retention period in days and years.

```
aws s3api put-object-lock-configuration --bucket mybucket --object-lock
-configuration '{ "ObjectLockEnabled": "Enabled", "Rule": {
"DefaultRetention": { "Mode": "COMPLIANCE", "Days": 1 }}}' --endpoint-url
https://s3.company.com
```

As with most of these operations, no response is returned on success so, we can perform a GET for the configuration to verify.

```
aws s3api get-object-lock-configuration --bucket mybucket --endpoint-url
https://s3.company.com
{
    "ObjectLockConfiguration": {
        "ObjectLockEnabled": "Enabled",
        "Rule": {
            "DefaultRetention": {
                "Mode": "COMPLIANCE",
                "Days": 1
            }
        }
    }
}
```

Next, you can put an object in the bucket with the retention configuration applied.

```
aws s3 cp myfile.txt s3://mybucket --endpoint-url https://s3.company.com
```

The PUT operation does return a response.

```
upload: ./myfile.txt to s3://mybucket/myfile.txt
```

On the retention object, the retention duration set on the bucket in the preceding example is converted to a retention timestamp on the object.

```
aws s3api get-object-retention --bucket mybucket --key myfile.txt
--endpoint-url https://s3.company.com
{
    "Retention": {
        "Mode": "COMPLIANCE",
        "RetainUntilDate": "2022-03-02T15:22:47.202000+00:00"
    }
}
```

Ransomware defense using replicated bucket with versioning

Learn how to replicate objects to a secondary bucket using StorageGRID CloudMirror.

Not all applications and workloads are going to be compatible with object lock. Another option is to replicate the objects to a secondary bucket either in the same grid (preferably a different tenant with restricted access), or any other S3 endpoint with the StorageGRID platform service, CloudMirror.

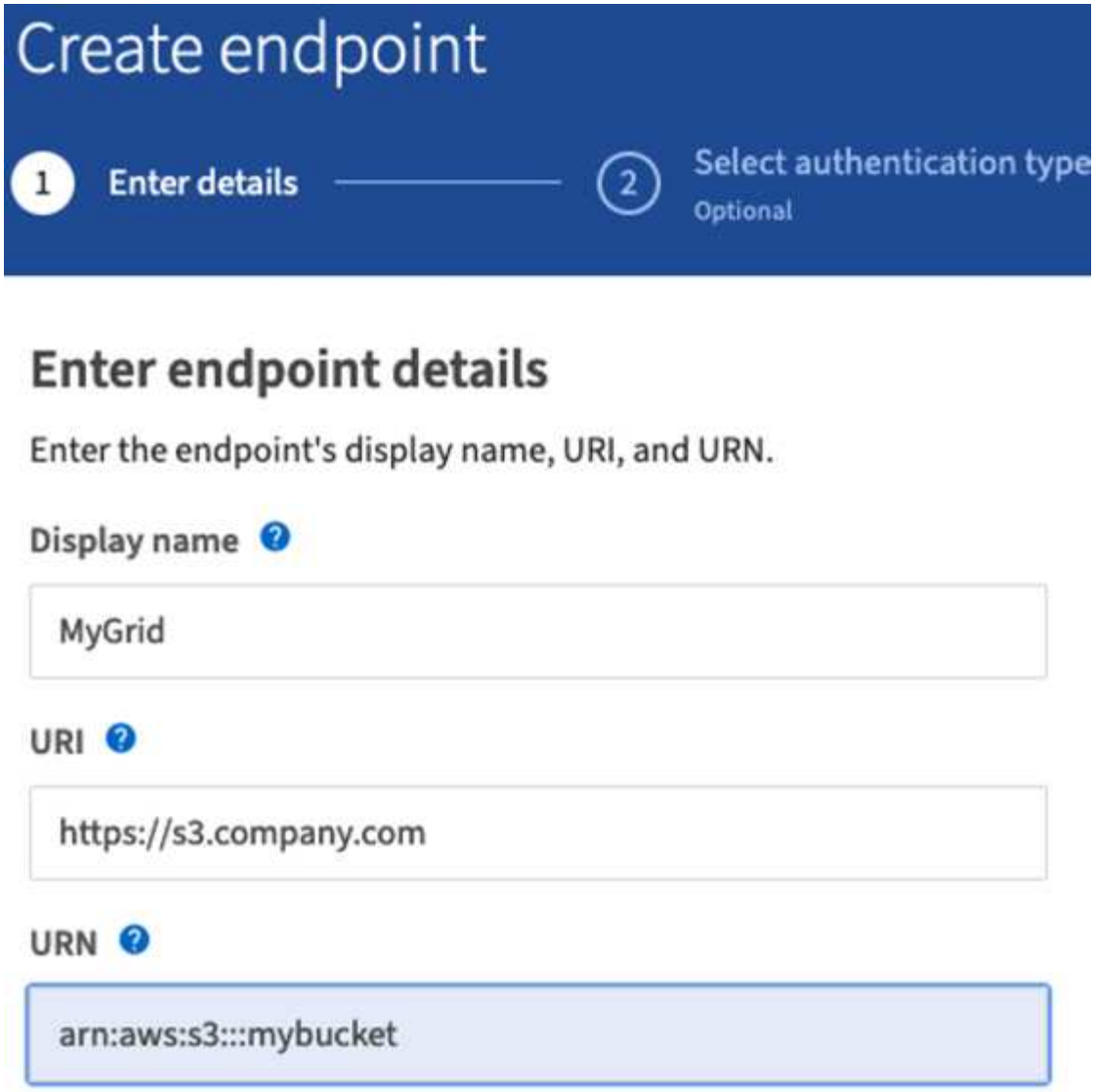
StorageGRID CloudMirror is a component of StorageGRID that can be configured to replicate the objects of a bucket to a defined destination as they are ingested into the source bucket and does not replicate deletes. Because CloudMirror is an integrated component of StorageGRID, it cannot be turned off or manipulated by an S3 API-based attack. You can configure this replicated bucket with versioning enabled. In this scenario you need some automated cleanup of the replicated bucket's old versions that are safe to discard. For this, you can use the StorageGRID ILM policy engine. Create rules to manage the object placement based on non-current time for several days sufficient to have identified and recovered from an attack.

One downside to this approach is that it consumes more storage by having a complete second copy of the bucket plus multiple versions of the objects retained for some time. Additionally, the objects that were intentionally deleted from the primary bucket must be manually removed from the replicated bucket. There are other replication options outside of the product, such as NetApp CloudSync, that can replicate deletes for a similar solution. Another downside for the secondary bucket being versioning enabled and not object lock enabled is that there exists a number of privileged accounts that might be used to cause damage on the secondary location. The advantage is that it should be a unique account to that endpoint or tenant bucket and the compromise likely does not include access to accounts on the primary location or vice-versa.

After the source and destination buckets are created and the destination is configured with versioning, you can configure and enable replication, as follows:

Steps

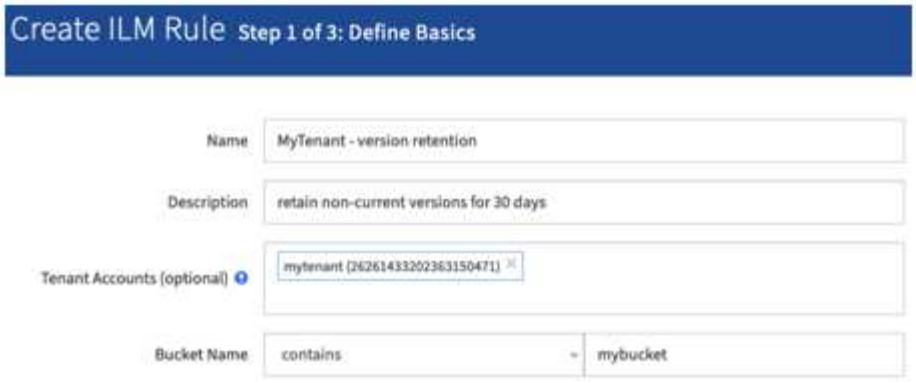
1. To configure CloudMirror, create a platform services endpoint for the S3 destination.

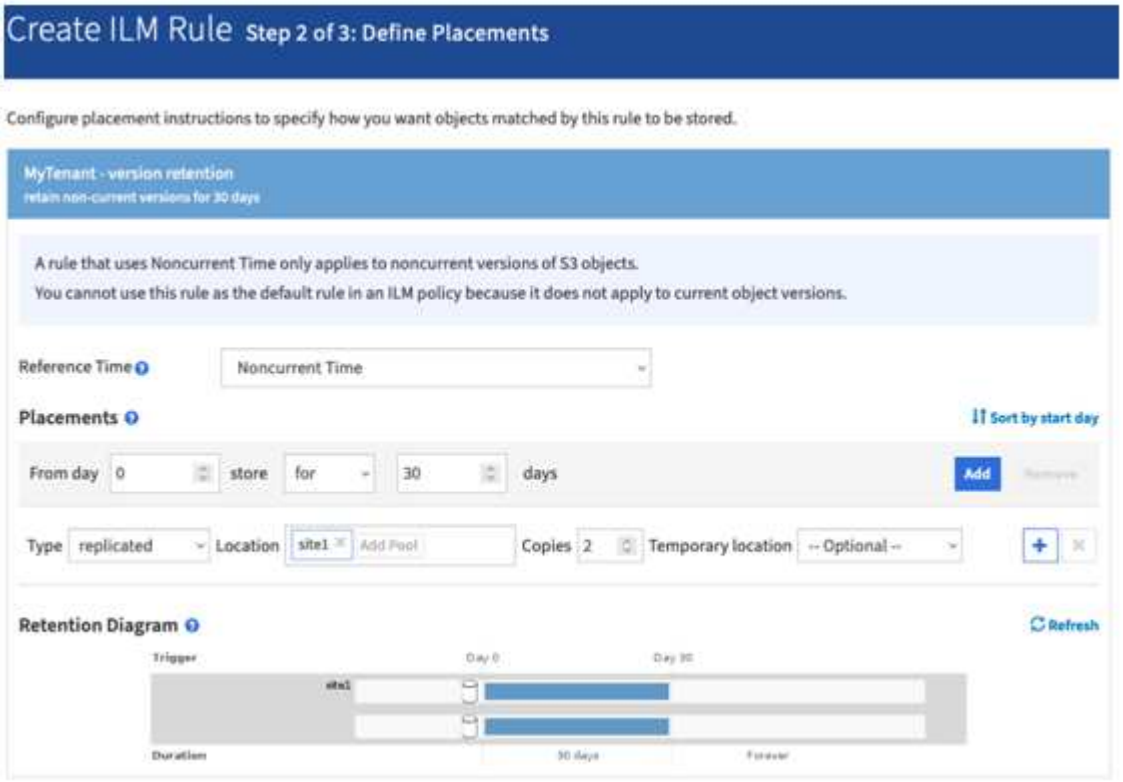


2. On the source bucket, configure replication to use the endpoint configured.

```
<ReplicationConfiguration>
    <Role></Role>
    <Rule>
        <Status>Enabled</Status>
        <Prefix></Prefix>
        <Destination>
            <Bucket>arn:aws:s3:::mybucket</Bucket>
            <StorageClass>STANDARD</StorageClass>
        </Destination>
    </Rule>
</ReplicationConfiguration>
```

3. Create ILM rules to manage the storage placement and version storage duration management. In this example, the non-current versions of the objects to store are configured.





There are two copies in site 1 for 30 days. You also configure the rules for the current version of the objects based on using ingest time as reference time in the ILM rule to match the source bucket storage duration. The storage placement for the object versions can be erasure coded or replicated.

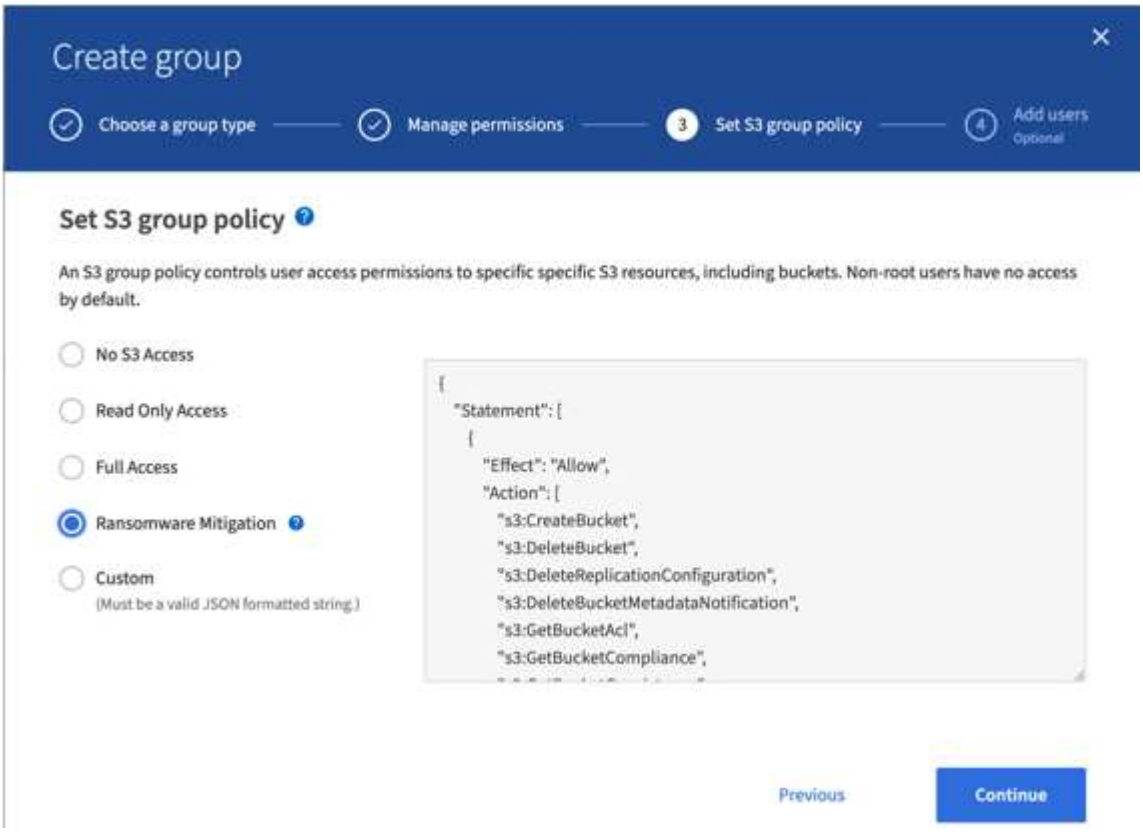
Ransomware defense using versioning with protective IAM policy

Learn how to protect your data by enabling versioning on the bucket and implementing IAM policies on user security groups in StorageGRID.

A method to protect your data without using object lock or replication is to enable versioning on the bucket and implement IAM policies on the user security groups to limit users' ability to manage versions of the objects. In the event of an attack, new bad versions of the data are created as the current version, and the most recent non-current version is the safe clean data. The accounts compromised to gain access to the data do not have

access to delete or otherwise alter the non-current version protecting it for later restore operations. Just like the previous scenario, ILM rules manage the retention of the noncurrent versions with a duration of your choice. The downside is that there is still the possibility of privileged accounts existing for a bad actor attack, but all application service accounts and users must be configured with a more restrictive access. The restrictive group policy must explicitly allow each action you want the users or application to be capable of and explicitly deny any actions that you do not want them to be capable of. NetApp does not recommend using a wildcard allow because a new action might be introduced in the future, and you will want to control whether it is allowed or denied. For this solution, the deny list must include DeleteObjectVersion, PutBucketPolicy, DeleteBucketPolicy, PutLifecycleConfiguration, and PutBucketVersioning to protect the versioning configuration of the bucket and object versions from user or programmatic changes.

In StorageGRID The S3 group policy option “Ransomware Mitigation” makes implementing this solution easier. When creating a user group in the tenant, after selecting the group permissions, you can see this optional policy.



The following is the content of the group policy that includes most of the available operations explicitly allowed and the minimum required denied.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:DeleteBucket",
                "s3:DeleteReplicationConfiguration",
"s3:DeleteBucketMetadataNotification",
```

```
                "s3:GetBucketAcl",
                "s3:GetBucketCompliance",
                "s3:GetBucketConsistency",
                "s3:GetBucketLastAccessTime",
                "s3:GetBucketLocation",
                "s3:GetBucketNotification"
"s3:GetBucketObjectLockConfiguration",
                "s3:GetBucketPolicy",
                "s3:GetBucketMetadataNotification",
                "s3:GetReplicationConfiguration",
                "s3:GetBucketCORS",
                "s3:GetBucketVersioning",
                "s3:GetBucketTagging",
                "s3:GetEncryptionConfiguration",
                "s3:GetLifecycleConfiguration",
                "s3:ListBucket",
                "s3:ListBucketVersions",
                "s3:ListAllMyBuckets",
                "s3:ListBucketMultipartUploads",
                "s3:PutBucketConsistency",
                "s3:PutBucketLastAccessTime",
                "s3:PutBucketNotification",
"s3:PutBucketObjectLockConfiguration",
                "s3:PutReplicationConfiguration",
                "s3:PutBucketCORS",
                "s3:PutBucketMetadataNotification",
                "s3:PutBucketTagging",
                "s3:PutEncryptionConfiguration",
                "s3:AbortMultipartUpload",
                "s3:DeleteObject",
                "s3:DeleteObjectTagging",
                "s3:DeleteObjectVersionTagging",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectLegalHold",
                "s3:GetObjectRetention",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging",
                "s3:ListMultipartUploadParts",
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectLegalHold",
                "s3:PutObjectRetention",
                "s3:PutObjectTagging",
```

```
                "s3:PutObjectVersionTagging",
                "s3:RestoreObject",
                "s3:ValidateObject",
                "s3:PutBucketCompliance",
                "s3:PutObjectVersionAcl"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "s3:DeleteObjectVersion",
                "s3:DeleteBucketPolicy",
                "s3:PutBucketPolicy",
                "s3:PutLifecycleConfiguration",
                "s3:PutBucketVersioning"
            ],
            "Resource": "arn:aws:s3:::*"
        }
    ]
}
```

Ransomware investigation and remediation

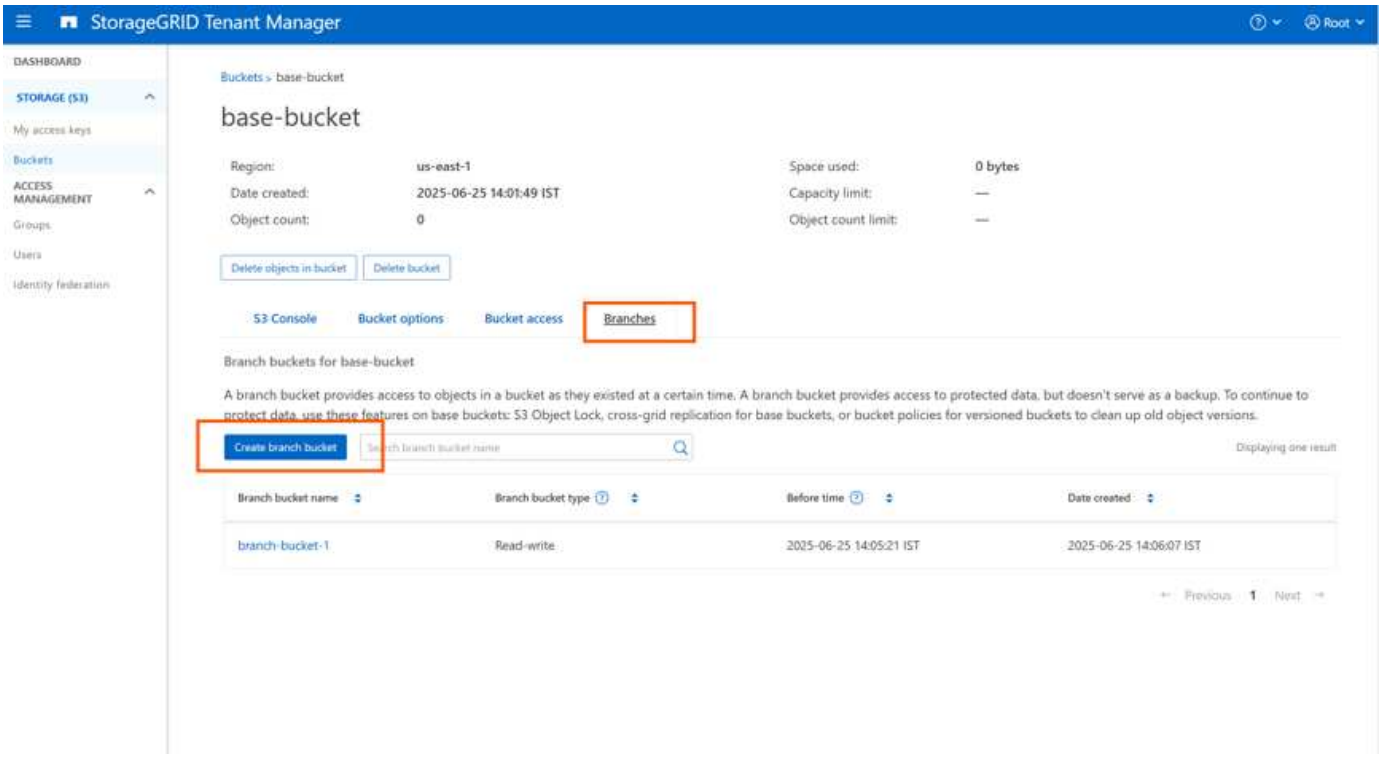
Learn how to investigate and remediate buckets after a possible ransomware attack with StorageGRID.

In StorageGRID 12.0, the new branch bucket feature has been added to extend the usefulness of versioning for ransomware defense. A branch bucket provides access to objects in a bucket as they existed at a certain time provided they still exist in the bucket. Branch buckets can only be created for versioning-enabled base buckets.

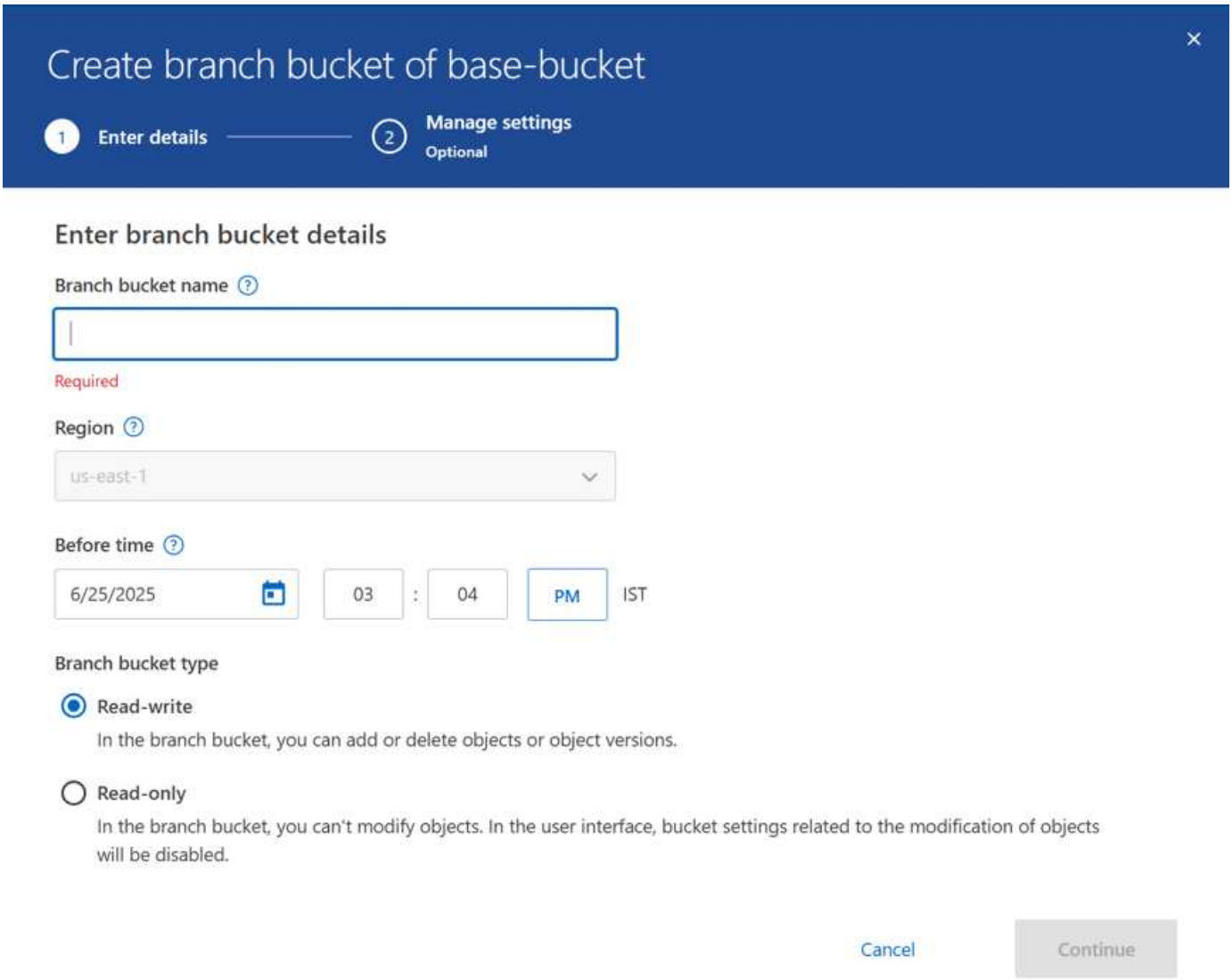
This means if you suspect a ransomware attack has occurred, you can create a read/write, or read-only branch bucket containing all objects and versions that existed prior to the initial attack time. You can use this branch bucket to compare against the base bucket contents to figure out what objects have changed and if the change was part of the attack or not. You could also use a branch bucket to continue client operations using the clean branch while investigating the attack.

Creating a Branch bucket

- Navigate to the base bucket details page and the Branches tab to create a branch bucket.



- Once the Create branch bucket button is clicked, a popup will open with prefilled details of the region associated with the base bucket.
- provide the branch bucket name, before time, and select what type of branch bucket to create.


Create branch bucket of base-bucket
1 Enter details
2 Manage settings
Optional
Enter branch bucket details
Branch bucket name
Required
Region
us-east-1
Before time
6/25/2025
03
04
PM
IST
Branch bucket type
Read-write
In the branch bucket, you can add or delete objects or object versions.
Read-only
In the branch bucket, you can't modify objects. In the user interface, bucket settings related to the modification of objects will be disabled.
Cancel
Continue

Copyright information

Trademark information