# S3 Ransomware Part 1: Attack Vector

By Spencer Gietzen

Published: 2019-06-10 · Archived: 2026-04-05 15:01:04 UTC

*This is part one in a two-part series on S3 Ransomware. You can find Part Two: Prevention and Defense here.*

## Introduction to the Attack Vector

Through our cloud security research, we at Rhino Security Labs developed a proof of concept "cloud ransomware" using KMS to encrypt objects within Amazon S3 buckets of a compromised AWS account. Ransomware is when an attacker gains access to a victim's system and encrypts the sensitive data on it. This is accompanied by a threat to delete or publicly release the data if the victim does not give payment within a certain time frame. Typically, these attacks are targeting servers, workstations, and similar setups, but as with many attacks, there is a "cloud-equivalent".

S3 Ransomware can be very damaging, and our research and blog post do not aim to assist attackers by any means. For this reason, only a slow proof-of-concept script will be released with this blog, so defenders are able to test their defenses. For the defenders and blue teamers, we created a second part to this post in which we discuss how to prevent and defend against S3 ransomware.

## Amazon S3 and AWS KMS

For those unfamiliar, Amazon Simple Storage Service (Amazon S3) is a robust static-file hosting service offered through Amazon Web Services. According to Amazon, S3 can be used to "store and protect any amount of data for a range of use cases, such as websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics". At a high level, S3 consists of "buckets" and "objects", in which objects are files that are stored in a bucket.

AWS Key Management Service (AWS KMS) is essentially a managed encryption key service offered by AWS. Amazon defines it as a service that "makes it easy for you to create and manage keys and control the use of encryption across a wide range of AWS services and in your applications".

### S3 Buckets

S3 buckets are not automatically server-side encrypted, meaning that without configuration, all files will be stored in cleartext on Amazon's servers. Buckets can have a "default" encryption setting that can be used as a fallback mechanism for when someone/something tries to upload a file without encryption. In that case, the default encryption method will kick in and ensure the file is not stored in cleartext.

### S3 Objects

Individual objects can also be encrypted separately from the "default" encryption on the bucket, unless enforced in the bucket's policy. This means that if there was a default encryption method set up on a bucket (or none at all), the uploader of a file can decide how to encrypt it on upload, which will override the default setting for the bucket.

### KMS Encryption

AWS KMS integrates with S3 object encryption, in that you can specify a particular KMS key to encrypt an object in a bucket. In that case, a user who tries to view a file in S3 would both need the "s3:GetObject" permission on the specific S3 object and the "kms:Decrypt" permission on the specific KMS key. Proper implementation of a KMS key policy can prohibit attackers from reading sensitive files in S3. If an attacker escalates their access and gains the "s3:GetObject" permission, they still may not be able to read files in S3 because they don't have decrypt permissions on the appropriate KMS key.

KMS keys can also be used cross-account, so if an attacker gains access to your S3 bucket, they may be able to encrypt your objects with a cross-account KMS key that only provides you with "encrypt" permissions. Because you don't control the KMS key that was used to encrypt your files, that means you can't view your own files.

## The Attack Path

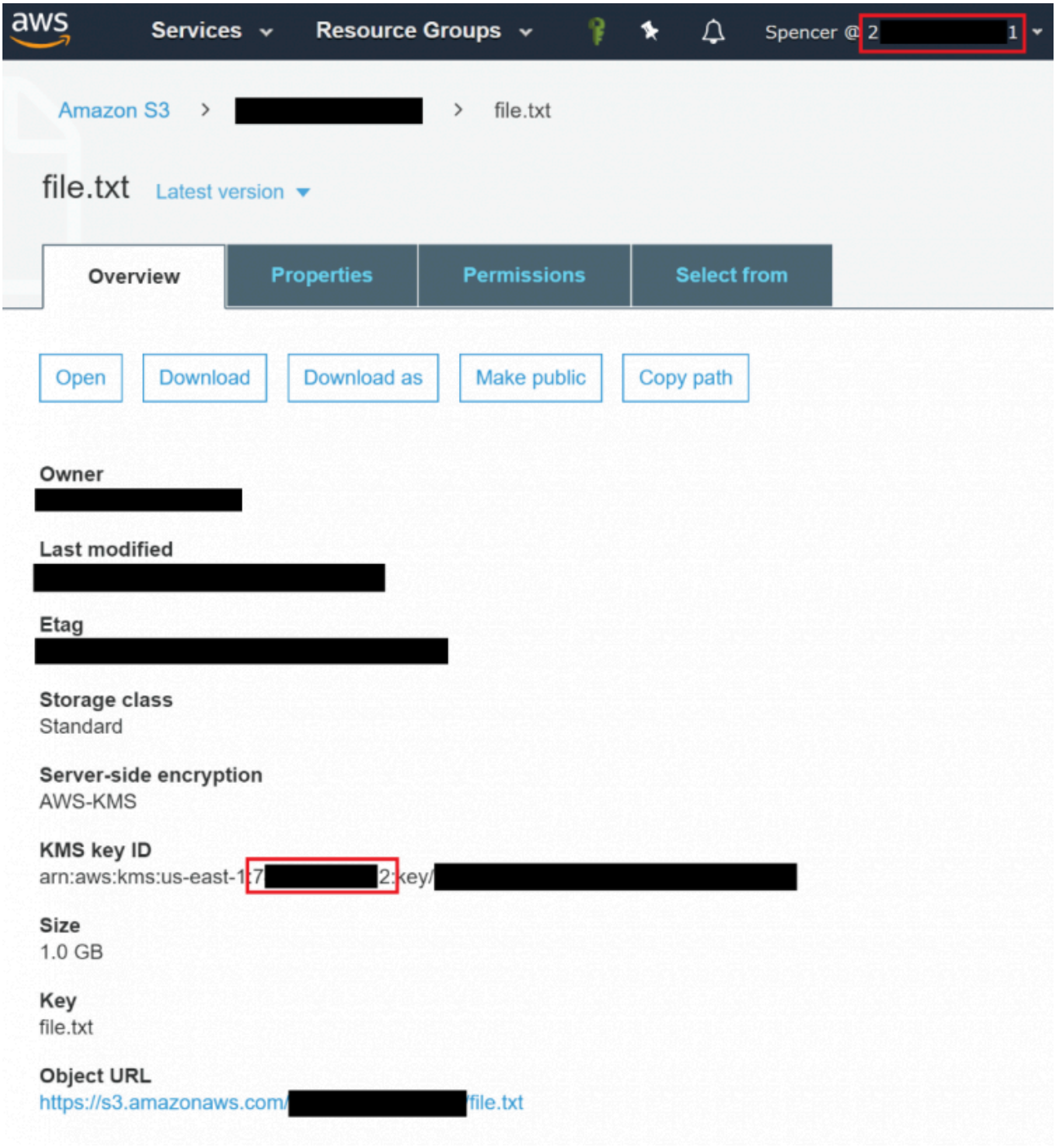
As part of this research, we wrote an advanced tool for performing ransomware attacks against S3 buckets and objects. This tool handles a variety of options and configurations in the target environment, and was written specifically for speed. However, because our research and blog post do not aim to assist attackers, we will not be releasing the tool for this attack. Instead, a script can be found below in the "Testing Your Buckets" section that is designed to assist defenders and blue teamers in protecting against this attack.

### Attack Walkthrough

1. **Attacker creates a KMS key** in their own "personal" AWS account (or another compromised account) and provides "the world" access to use that KMS key for encryption. This means that it could be used by any AWS user/role/account to encrypt, but **not** decrypt objects in S3.
2. **Attacker identifies a target S3 bucket and gains write-level access to it**, which is possible through a variety of different means. This could include poor configuration on buckets that expose them publicly or an attacker gaining access to the AWS environment itself. Typically attackers would target buckets with sensitive information, such as PII, PHI, logs, backups, and more.
3. **Attacker checks the configuration of the bucket** to determine if it is able to be targeted by ransomware. This would include checking if S3 Object Versioning is enabled and if multi-factor authentication delete (MFA delete) is enabled. If Object Versioning is not enabled, then they are good to go. If Object Versioning is enabled, but MFA delete is disabled, the attacker can just disable the Object Versioning. If both Object Versioning and MFA delete are enabled, it would require *a lot* of extra work to be able to ransomware that specific bucket.
4. **Attacker uses the AWS API to replace each object in a bucket with a new copy of itself**, but this time, it is encrypted with the attackers KMS key.
5. **Attacker schedules the deletion of the KMS key** that was used for this attack, giving a 7 day window to their target until the key is deleted and the data is lost forever.

6. **Attacker uploads a final file** such as “ransom-note.txt” without encryption, which instructs the target on how to get their files back.

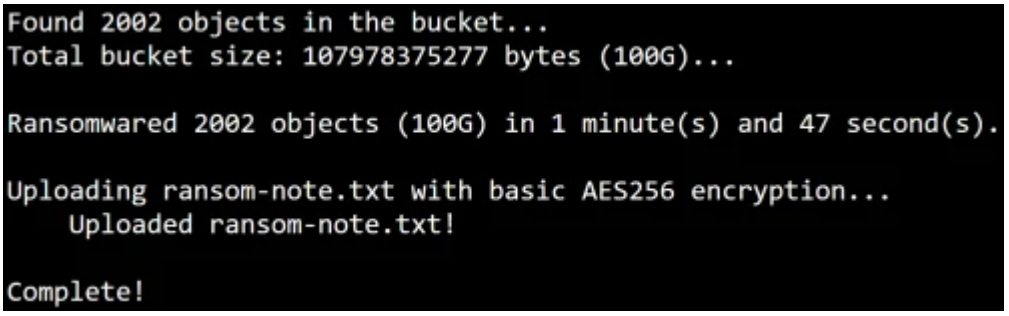
The following screenshot shows an example of a file that was targeted for a ransomware attack. As you can see, the account ID that owns the KMS key that was used to encrypt the object (7**********2) is different than the account ID of the account that owns the object (2**********1).




The next screenshot shows what happens when the object owner uses a pre-signed URL to try to view the object. Access is denied because even though the object owner has permission to view the object, they don’t have permission to use the KMS key to decrypt the encrypted object, thus preventing access to it.


### Attack Metrics

To test this attack and to see the viability of it, we ran Rhino's internal tool against an S3 bucket that was hosting approximately 2000 different files of different sizes, totaling about 100 GB of data. To ransomware the entire bucket (encrypt every individual object) it only took 1 minute and 47 seconds.




Typical CloudTrail logs can take up to 15 minutes to be delivered to an S3 bucket, but in testing, it was found that CloudTrail S3 data event logs were delivered in approximately 5 minutes. Logs delivering in 5 minutes is much better than 15 minutes, but the problem is that 5 minutes is still too long. Our test above shows that approximately 2,000 files, totaling about 100 GB of data can be ransomwared in as little as 1 minute and 47 seconds, which is far under the 5 minute limit. By following those numbers (theoretically—results would be slightly different in practice depending on the size of the individual files), an attacker can ransomware just over 900 MB of data per second. That means in a 5-minute period, they could ransomware approximately 270 GB of data before you even have a chance to see what happened.

The next thing to consider is how fast you can react once you do see what's going on. As an example, it could take you 5 minutes from the first log delivery to act against an attacker trying to ransomware your data and block their access to your environment. At that point, you're 10 minutes into the attack, which comes out to approximately 540 GB of data. To give some perspective, the following list was calculated based on the data from this "How Big is a Gig?" cheat sheet from iClick.

540 GB of data, expressed in common file types:

- 360,000 1.5 MB photos
- 360 1.5 GB movies
- 135,000 4 MB songs

Of course, this is all an estimate and there are many more factors to be considered, but this is to give an idea of how much 540 GB of data really is. The list above shows that no matter how fast you respond, the attacker will still be able to ransomware a large amount of data. This is why it is extremely important to defend against this attack as much as possible, in addition to implementing incident response plans.

## S3 Ransomware Proof-of-Concept

We also wrote a simple proof-of-concept script for demonstrating this attack, so that defenders can test it out and hopefully build detections around it. This script will only work against buckets that have object versioning disabled.

The script, which can be found on our GitHub, does the following:

1. Gathers the first 100 objects in the bucket (or all, if fewer than 100 objects in the bucket)
2. One by one, overwrites each object with itself using the new KMS encryption key

### Script Usage

To use the PoC script, modify the values of “aws_cli_profile”, “bucket_name”, and “kms_key_arn” to match your environment, then run the script with Python 3.6+.

Variables:

- **aws_cli_profile**: The AWS CLI profile to use for authentication with AWS (~/.aws/credentials).
- **bucket_name:** The name of the S3 bucket to target.
- **kms_key_arn:** The ARN of the KMS key to use for the attack.

Here’s a screenshot of the script running and its output:

```
PS C:\Users\      \Documents\S3 Ransomware> py .\s3-ransomware-poc.py
Complete! Encrypted 3 objects!
```

Because you are specifying the KMS key to use and the S3 bucket to target, you shouldn’t lose access to your data (such as if a real attacker used their own KMS key on your S3 bucket). To be cautious, though, **do not run this script against production/important data**.

## Conclusion and Defensive Controls (Part 2)

Ransomware is an extremely threatening attack vector that has become more and more popular in recent years. The amount of data that an attacker can ransomware in a short period of time is very significant. Attackers continue advancing, so defenders need to stay one step ahead to prevent cloud-based attacks that have traditionally been performed elsewhere.

S3 ransomware can be very dangerous to an organization, but there are a variety of both simple and complex defense mechanisms that defenders can put in place to prevent this attack.

These methods are explained in depth in part two of this post.

---