Server-Side Encryption and KMS

January 3, 2016

The Key Management System (KMS) is a sub-part of the identity management (IM) inside of AWS which allows a user to manage their encrypted keys via Amazon. With it, the user won't have to buy their own encryption key generation devices because Amazon generates, manages, and backs hardware-generated keys. These keys can then be used with various Amazon services (like Red Shift or S3) that store or transmit user data – the kinds of data most users want to keep encrypted.

Transparency is the name of the game when it comes to server-side encryption: data is encrypted as it enters S3 and decrypted as users access it with keys.

This encryption process is especially useful for business data because it prevents the theft of information on disks. If there are two backdrop indications and strong passwords on the Amazon account associated with a disk, a thief won't be able to access the keys protected by the account.

These keys can be protected by a few different methods (or layers) of server-side encryption in Amazon:

1. The standard S3, the AES-256
2. The Amazon KMS managed keys
3. Customer-provided keys

While the standard S3 supports a generic AES-256 encryption, users also have the option to turn on encryption using KMS keys. This second method allows for the encryption of the bucket of data inside of S3. A major benefit to this system is that if a user wants to immediately make a disk's data unavailable, they can just destroy the key, rendering the disk unreadable.

In comparison, just destroying a disk's bucket doesn't destroy the data sitting on the disk. With the right equipment, the bucket data could still technically be accessed if encryption keys aren't used.

By deleting a key out of KMS (an action preceded by heavy warnings), the data is immediately unreadable although it still takes up space on a disk. This process is as good as formatting the disk; however, there is an automatic safeguard in place. A timeframe spanning anywhere from 7 to thirty days can be created during which time a user can reenable a key before the data is gone forever. Similarly, KMS allows new keys to be rotated in - an old key can be kept around for a certain amount of time until the new key rotates in as it is requested and used.

Keys can be accessed via the AWS console, or by using the API's, users can utilize their own keys rather than the standard Amazon S3 AES-256 keys. These keys are still transparent to the user – Amazon S3 will look them up because it has key access when tokens are in place, and the program will be able to decrypt the data as comes back down to the user. With this method, there is no need to specify a key each time data is accessed because Amazon does it all transparently for the operator.

When it comes to the third method, customer-provided keys, a user must provide a key for every request regarding data in S3. Amazon does not have these keys, but if they were ever lost they could be deleted, immediately rendering data useless.

Additionally, if users are working on a team, KMS will allow them to specify policies around which accounts inside a certain AWS account have access to read specific keys. Users can be prevented from receiving two separate buckets of data if they don't have access to the key. Key access is customizable, allowing for the addition and removal of access for certain users. While someone may be given access to a bucket, they may not have access to the key that can decrypt the data in it.

If this all sounds a bit complicated, the bare minimum of turning on the AES-256 encryption is still recommended. This is because even if a bucket is deleted, the data is not completely gone - it remains on a disk, and that is a good enough reason to encrypt it.

Going one step further with KMS is easy and provides an extra layer of control over data encryption. The third layer of customer provided keys, creates extra overhead, that, while probably not necessary for most people, allows for greater data encryption control for those seeking it.