
Sneaker

build unknown license Apache V2

Setec Astronomy? Keynote Shogun.

sneaker is a utility for storing sensitive information on AWS using S3 and the Key Management Service (KMS) to provide durability, confidentiality, and integrity.

Secrets are stored on S3, encrypted with AES-256-GCM and single-use, KMS-generated data keys.

Table Of Contents

- WARNING
- Installing
- Using
 - Configuring Access To AWS
 - Setting Up The Environment
 - Basic Operations
 - Packing Secrets
 - Unpacking Secrets
 - Encryption Contexts
 - Maintenance Operations
- Implementation Details
- Architecture
- Threat Model
 - Assumptions
 - Threats From A KMS Compromise
 - Threats From An S3 Compromise
 - Threats From Seizure Or Compromise Of The User's Computer

WARNING

This project has not been reviewed by security professionals. Its internals, data formats, and interfaces may change at any time in the future without warning.

Installing

```
1 go get -d -u github.com/codahale/sneaker
2 cd $GOPATH/src/github.com/codahale/sneaker
3 make install
4 sneaker version
```

Using

Configuring Access to AWS

`sneaker` requires access to AWS APIs, which means it needs a set of AWS credentials. It will look for the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` environment variables, the default credentials profile (e.g. `~/.aws/credentials`), and finally any instance profile credentials for systems running on EC2 instances.

In general, if the `aws` command works, `sneaker` should work as well.

If you have multi-factor authentication enabled for your AWS account (**and you should**), you may need to provide a token via the `AWS_SESSION_TOKEN` environment variable.

If you're using IAM instance roles, you may need to set the `AWS_REGION` environment variable to the AWS region you're using (e.g. `us-east-1`).

Setting Up The Environment

Sneaker requires two things: a KMS master key and an S3 bucket.

You can create a KMS key via the AWS Console or using a recent version of `aws`. When you've created the key, store its ID (a UUID) in the `SNEAKER_MASTER_KEY` environment variable:

```
1 export SNEAKER_MASTER_KEY="9ed356fb-5f0f-4792-983d-91866faa3705"
```

As with the key, you can create an S3 bucket via the AWS Console or with the `aws` command. You can either use a dedicated bucket or use a directory in a common bucket, but we recommend you do two things:

1. Use a `Private` ACL. In addition to the cryptographic controls of `sneaker`, access control is critical in preventing security breaches.
2. Enable access logging, ideally to a tightly-controlled, secure bucket. While Amazon's CloudTrail provides audit logging for the vast majority of AWS services, it does not do so for S3 access.

Once you're done, set the `SNEAKER_S3_PATH` environment variable to the location where secrets should be stored (e.g. `s3://bucket1/secrets/`):

```
1 export SNEAKER_S3_PATH="s3://bucket1/secrets/"
```

(That will store the encrypted secrets in the bucket `bucket1` prefixed with `secrets/`.)

Basic Operations

Once you've got `sneaker` configured, try listing the secrets:

```
1 sneaker ls
```

This will print out a table of all uploaded secrets. You haven't uploaded anything yet, so the table will be empty.

Let's create an example secret file and upload it:

```
1 echo "This is a secret!" > secret.txt
2 sneaker upload secret.txt example/secret.txt
```

This will use KMS to generate a random, 256-bit data key, encrypt the secret with AES-256-GCM, and upload the encrypted secret and an encrypted copy of the data key to S3. Running `sneaker ls` should display a table with the file in it.

If your file is so sensitive it shouldn't be stored on disk, using `-` instead of a filename will make `sneaker` read the data from `STDIN`.

You can download a single file:

```
1 sneaker download example/secret.txt secret.txt
```

Finally, you can delete the file:

```
1 sneaker rm example/secret.txt
```

Packing Secrets

To install a secret on a machine, you'll need to pack them into a tarball:

```
1 sneaker pack example/*,extra.txt example.tar.enc
```

This will perform the following steps:

1. Download and decrypt all secrets matching any of the patterns: `example/*` or `extra.txt`.

-
2. Package all the decrypted secrets into a `TAR` file in memory.
 3. Generate a new data key using KMS.
 4. Use the data key to encrypt the `TAR` file with AES-GCM.
 5. Write both the encrypted data key and the encrypted `TAR` file to `example.tar.enc`.

Using `-` as the output path will make `sneaker` write the data to `STDOUT`.

The result is safe to store and transmit – only those with access to the `Decrypt` operation of the KMS key being used will be able to decrypt the data.

You can also use a different KMS key than your `SNEAKER_MASTER_KEY` when packing secrets by using the `--key-id` flag:

```
1 sneaker pack example/* example.tar.enc --key-id=deb207cd-d3a7-4777-aca0-01fbceb4c927
```

This allows you to unpack your secrets in environments which don't have access to the key used to store your secrets.

Unpacking Secrets

To unpack the secrets, run the following:

```
1 sneaker unpack example.tar.enc example.tar
```

This will perform the following steps:

1. Read `example.tar.enc`.
2. Extract the encrypted data key and encrypted `TAR` file.
3. Use KMS to decrypt the data key.
4. Decrypt the `TAR` file and write the result to `example.tar`.

Using `-` instead of a filename will make `sneaker` read the data from `STDIN`. Likewise, using `-` as the output path will make `sneaker` write the data to `STDOUT`. This allows you to pipe the output directly to a `tar` process, for example.

Encryption Contexts

KMS supports the notion of an Encryption Context: semi-structured data used in the encryption of data which is then required for resulting decryption operations to be successful.

`sneaker` uses the `SNEAKER_MASTER_CONTEXT` environment variable as the default encryption context for the secrets which are stored in S3. In addition, `sneaker` also includes the full S3 path, including bucket and prefix. Because of this, secrets in S3 cannot be renamed; they can only be deleted and re-uploaded.

Note: there is currently no way to change the contents of `SNEAKER_MASTER_CONTEXT` in place. If you want to change it, you'll need to download all your secrets and re-upload them with the new context.

For packing and unpacking secrets you can specify a different encryption context on the command line:

```
1 sneaker pack example/* secrets.tar.enc --context="hostname=web1.example.com,version=20"
```

That same context (`hostname=web1.example.com,version=20`) **must** be used to unpack those secrets:

```
1 sneaker unpack secrets.tar.enc secrets.tar --context="hostname=web1.example.com,version=20"
```

All data in the encryption contexts are logged via CloudTrail, which allows you to track when and where particular secrets are packed and unpacked.

Maintenance Operations

A common maintenance task is key rotation. To rotate the data keys used to encrypt the secrets, run `sneaker rotate`. It will download and decrypt each secret, generate a new data key, and upload a re-encrypted copy.

To rotate the KMS key used for each secret, simply specify a different `SNEAKER_MASTER_KEY` and run `sneaker rotate`.

Implementation Details

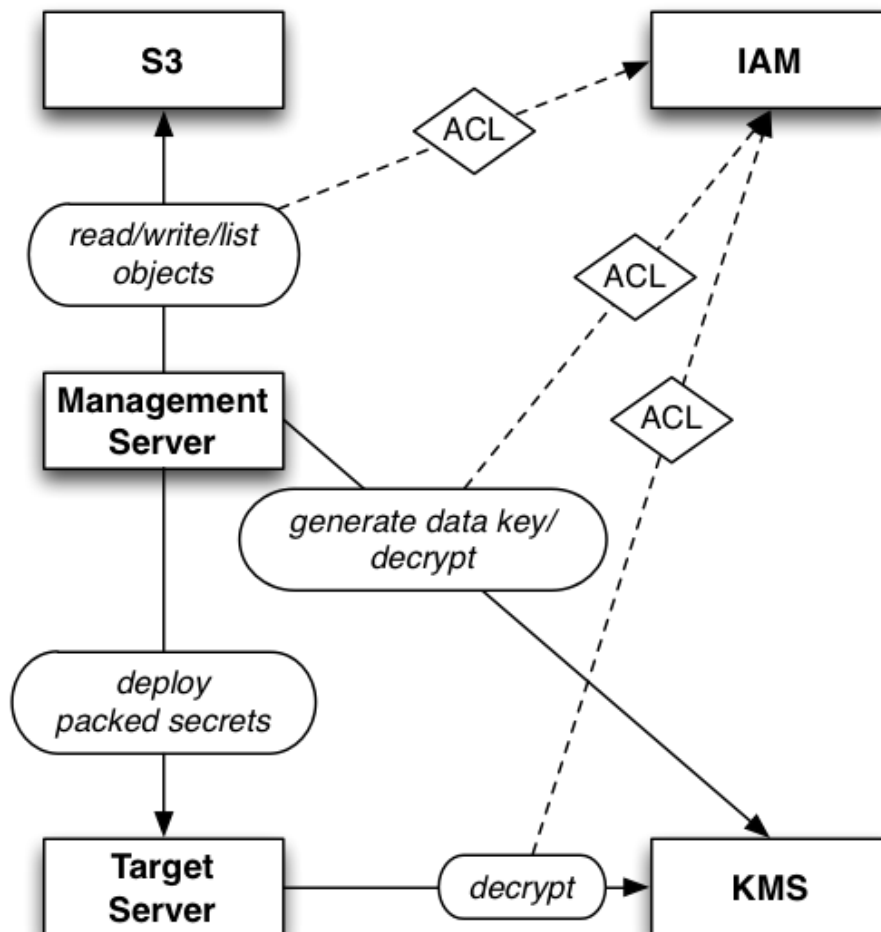
All data is encrypted with AES-256-GCM using random KMS data keys and random nonces. The ID of the KMS key is used as authenticated data.

The final result is the concatenation of the following:

- A four-byte header of the length of the encrypted KMS data key, in bytes, in network order.
- The encrypted KMS data key, verbatim. (This is an opaque Amazon format which includes the key ID.)

-
- The AES-256-GCM ciphertext and tag of the secret.

Architecture



Threat Model

The threat model is defined in terms of what each possible attacker can achieve. The list is intended to be exhaustive, i.e. if an entity can do something that is not listed here then that should count as a break of Sneaker.

In broad strokes, the confidentiality and integrity of content stored with Sneaker is predicated on the integrity of Sneaker, the confidentiality and integrity of KMS, and the integrity of S3.

Assumptions

- The user must act reasonably and in their best interest. They must not reveal secrets or allow unauthorized access to secrets.
- The user must run a copy of Sneaker which has not been suborned.
- The user's computer must function correctly and not be compromised by malware.
- Communications with Amazon have confidentiality and integrity ensured by the use of TLS.
- Key Management Service is reasonably secure: its keys will not be compromised, its random numbers are unguessable to adversaries, its cryptographic algorithms are correct.
- The authentication and access control functionality of both KMS and S3 are secure.
- AES-256 and GCM's security guarantees are valid.

Threats From A KMS Compromise

An attacker who suborns KMS can:

- Create forged secret packages.
- Decrypt packaged tarballs.
- Deny the ability to decrypt secrets, either temporarily or permanently.

Threats From An S3 Compromise:

An attacker who suborns S3 can:

- Delete or modify secrets such that they are no longer valid.

Threats From Seizure Or Compromise Of The User's Computer

An attacker who physically seizes the user's computer (or compromises the user's backups) or otherwise compromises it can:

- Recover AWS credentials and pose as the user. If multi-factor authentication is not enabled, this would allow the attacker to extract all secrets, modify secrets, etc.