Encrypting File Data with Amazon Elastic File System

Encryption of Data at Rest and in Transit

April 2018

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

In today's world of cybercrime, hacking attacks, and the occasional security breach, securing data has become increasingly important to organizations. Government regulations and industry or company compliance policies may require data of different classifications to be secured by using proven encryption policies, cryptographic algorithms, and proper key management. This paper outlines best practices for encrypting shared file systems on AWS using Amazon Elastic File System (Amazon EFS).

Introduction

Amazon Elastic File System (Amazon EFS)[1] provides simple, scalable, highly available, and highly durable shared file systems in the cloud. The file systems you create using Amazon EFS are elastic, allowing them to grow and shrink automatically as you add and remove data. They can grow to petabytes in size, distributing data across an unconstrained number of storage servers in multiple Availability Zones. Data stored in these file systems can be encrypted at rest and in transit using Amazon EFS. For encryption of data at rest, you can create encrypted file systems through the AWS Management Console or the AWS Command Line Interface (AWS CLI). Or you can create encrypted file systems programmatically through the Amazon EFS API or one of the AWS SDKs. Amazon EFS integrates with AWS Key Management Service (AWS KMS)[2] for key management. You can also enable encryption of data in transit by mounting the file system and transferring all NFS traffic over an encrypted Transport Layer Security (TLS) tunnel. This paper outlines best practices for encrypting shared file systems on AWS using Amazon EFS. It describes how to enable encryption of data in transit at the client connection layer, and how to create an encrypted file system in the AWS Management Console and in the AWS CLI. Using the APIs and SDKs to create an encrypted file system is outside the scope of this paper, but you can learn more about how this is done by reading Amazon EFS API in the *Amazon EFS User Guide*[3] or the SDK documentation.[4]

Basic Concepts and Terminology

This section defines concepts and terminology referenced in this whitepaper.

- **Amazon Elastic File System (Amazon EFS)** – A highly available and highly durable service that provides simple, scalable, shared file storage in the AWS Cloud. Amazon EFS provides a standard file system interface and file system semantics. You can store virtually an unlimited amount of data across an unconstrained number of storage servers in multiple Availability Zones.
- **AWS Identity and Access Management (IAM)[5] –** A service that enables you to securely control fine-grained access to AWS service APIs. Policies are created and used to limit access to individual users, groups, and roles. You can manage your AWS KMS keys through the IAM console.

- **AWS KMS –** A managed service that makes it easy for you to create and manage the encryption keys used to encrypt your data. It is fully integrated with AWS CloudTrail to provide logs of API calls made by AWS KMS on your behalf to help meet compliance or regulatory requirements.
- **Customer master key (CMK) –** Represents the top of your key hierarchy. It contains key material to encrypt and decrypt data. AWS KMS can generate this key material, or you can generate it and then import it into AWS KMS. CMKs are specific to an AWS account and AWS Region and can be customer-managed or AWS-managed.
    - **AWS-managed CMK** – A CMK that is generated by AWS on your behalf. An AWS-managed CMK is created when you enable encryption for a resource of an integrated AWS service. AWS-managed CMK key policies are managed by AWS and you cannot change them. There is no charge for the creation or storage of AWS-managed CMKs.
    - **Customer-managed CMK** – A CMK you create by using the AWS Management Console or API, AWS CLI, or SDKs. You can use a customer-managed CMK when you need more granular control over the CMK.
- **KMS permissions –** Permissions that control access to a customer-managed CMK. These permissions are defined using the key policy or a combination of IAM policies and the key policy. For more information, see Overview of Managing Access in the *AWS KMS Developer Guide*.[6]
- **Data keys –** Cryptographic keys generated by AWS KMS to encrypt data outside of AWS KMS. AWS KMS allows authorized entities to obtain data keys protected by a CMK.
- **Transport Layer Security (TLS, formerly called Secure Sockets Layer [SSL]) –** Cryptographic protocols essential for encrypting information that is exchanged over the wire.
- **EFS mount helper –** A Linux client agent (amazon-efs-utils) used to simplify the mounting of EFS file systems. It can be used to setup, maintain, and route all NFS traffic over a TLS tunnel.

For more information about basic concepts and terminology, see AWS Key Management Service Concepts in the *AWS KMS Developer Guide*.[7]

Encryption of Data at Rest

You can create an encrypted file system so all your data and metadata is encrypted at rest using an industry-standard AES-256 encryption algorithm. Encryption and decryption is handled automatically and transparently, so you don't have to modify your applications. If your organization is subject to corporate or regulatory policies that require encryption of data and metadata at rest, we recommend creating an encrypted file system.

Managing Keys

Amazon EFS is integrated with AWS KMS, which manages the encryption keys for encrypted file systems. AWS KMS also supports encryption by other AWS services such as Amazon Simple Storage Service (Amazon S3), Amazon Elastic Block Store (Amazon EBS), Amazon Relational Database Service (Amazon RDS), Amazon Aurora, Amazon Redshift, Amazon WorkMail, Amazon WorkSpaces, etc. To encrypt file system contents, Amazon EFS uses the Advanced Encryption Standard algorithm with XTS Mode and a 256-bit key (XTS-AES-256).

There are three important questions to answer when considering how to secure data at rest by adopting any encryption policy. These questions are equally valid for data stored in managed and unmanaged services.

Where are keys stored?

AWS KMS stores your master keys in highly durable storage in an encrypted format to help ensure that they can be retrieved when needed.

Where are keys used?

Using an encrypted Amazon EFS file system is transparent to clients mounting the file system. All cryptographic operations occur within the EFS service, as data is encrypted before it is written to disk and decrypted after a client issues a read request.

Who can use the keys?

AWS KMS key policies control access to encryption keys. You can combine them with IAM policies to provide another layer of control. Each key has a key policy. If the key is an AWS-managed CMK, AWS manages the key policy. If the key is a customer-managed CMK, you manage the key policy. These key policies are the primary way to control access to CMKs. They define the permissions that govern the use and management of keys. When you create an encrypted file system, you grant the EFS service access to use the CMK on your behalf. The calls that Amazon EFS makes to AWS KMS on your behalf appear in your CloudTrail logs as though they originated from your AWS account.

For more information about AWS KMS and how to manage access to encryption keys, see Overview of Managing Access to Your AWS KMS Resources in the *AWS KMS Developer Guide*.[8]

For more information about how AWS KMS manages cryptography, see the AWS KMS Cryptographic Details whitepaper.[9]

For more information about how to create an administrator IAM user and group, see Creating Your First IAM Admin User and Group in the *IAM User Guide*.[10]

Creating an Encrypted File System

You can create an encrypted file system using the AWS Management Console, AWS CLI, Amazon EFS API, or AWS SDKs. You can only enable encryption for a file system when you create it. Amazon EFS integrates with AWS KMS for key management and uses a CMK to encrypt the file system. File system metadata, such as file names, directory names, and directory contents, are encrypted and decrypted using an EFS-managed key. The contents of your files, or file data, is encrypted and decrypted using a CMK that you choose. The CMK can be one of three types:

- An AWS-managed CMK for Amazon EFS
- A customer-managed CMK from your AWS account
- A customer-managed CMK from a different AWS account

All users have an AWS-managed CMK for Amazon EFS, whose alias is *aws/elasticfilesystem*. AWS manages this CMK's key policy and you cannot change it. There is no cost for creating and storing AWS-managed CMKs.

If you decide to use a customer-managed CMK to encrypt your file system, select the key alias of the customer-managed CMK that you own or enter the Amazon Resource Name (ARN) of a customer-managed CMK that is owned by a different account. With a customer-managed CMK that you own, you control which users and services can use the key through key policies and key grants. You also control the life span and rotation of these keys by choosing when to disable, re-enable, delete, or revoke access to them. AWS KMS charges a fee for creating and storing customer-managed CMKs. For information about managing access to keys in other AWS accounts, see Allowing External AWS Accounts to Access a CMK in the *AWS KMS Developer Guide*.[11]

For more information about how to manage customer-managed CMKs, see AWS Key Management Service Concepts in the *AWS KMS Developer Guide*.[12]

The following sections discuss how to create an encrypted file system using the AWS Management Console and using the AWS CLI.

Creating an Encrypted File System Using the AWS Management Console

To create an encrypted Amazon EFS file system using the AWS Management Console, follow these steps.

1. On the Amazon EFS console, select **Create file system** to open the file system creation wizard.
2. For **Step 1: Configure file system access**, choose your VPC, create your mount targets, and then choose **Next Step**.
3. For **Step 2: Configure optional settings**, add any tags, choose your performance mode, select the box to enable encryption for your file system, select a KMS master key, and then choose **Next Step**.

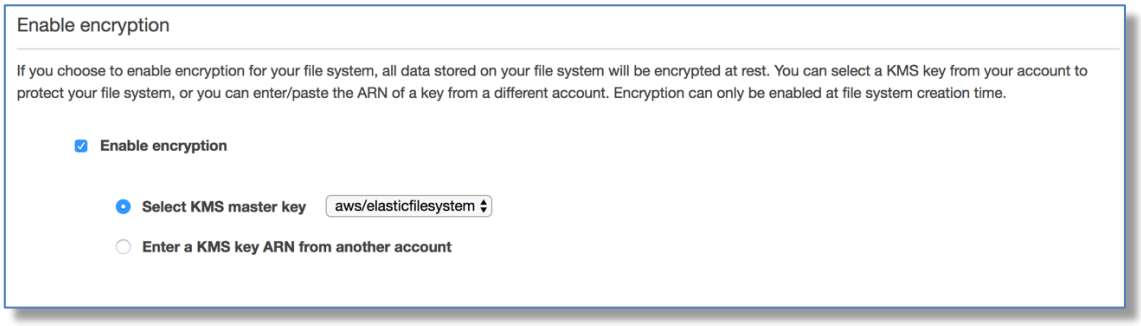

Figure 1: Enabling encryption through the AWS Management Console

4. For **Step 3: Review and create**, review your settings and choose **Create File System**.

Creating an Encrypted File System Using the AWS CLI

When you use the AWS CLI to create an encrypted file system, you use additional parameters to set the encryption status and customer-managed CMK. Be sure you are using the latest version of the AWS CLI. For information about how to upgrade your AWS CLI, see Installing the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.[13]

In the `CreateFileSystem` operation, the `--encrypted` parameter is a Boolean and is required for creating encrypted file systems. The `--kms-key-id` is required only when you use a customer-managed CMK and you include the key's alias or ARN. Do not include this parameter if you're using the AWS-managed CMK.

```
$ aws efs create-file-system \
  --creation-token $(uuidgen) \
  --performance-mode generalPurpose \
  --encrypted \
  --kms-key-id user/customer-managedCMKalias
```

For more information about creating Amazon EFS file systems using the AWS Management Console, AWS CLI, AWS SDKs, or Amazon EFS API, see the *Amazon EFS User Guide*.[14]

Using an Encrypted File System

Encryption has minimal effect on I/O latency and throughput. Encryption and decryption are transparent to users, applications, and services. All data and metadata is encrypted by Amazon EFS on your behalf before it is written to disk and is decrypted before it is read by clients. You don't need to change client tools, applications, or services to access an encrypted file system.

Enforcing Encryption of Data at Rest

Your organization might require the encryption of all data that meets a specific classification or is associated with a particular application, workload, or environment. You can enforce data encryption policies for Amazon EFS file systems by using detective controls that detect the creation of a file system and verify that encryption is enabled. If an unencrypted file system is detected, you can respond in a number of ways, ranging from deleting the file system and mount targets to notifying an administrator.

Be aware that if you want to delete the unencrypted file system but want to retain the data, you should first create a new encrypted file system. Next, you should copy the data over to the new encrypted file system. After the data is copied over, you can delete the unencrypted file system.

Detecting Unencrypted File Systems

You can create an Amazon CloudWatch alarm to monitor CloudTrail logs for the `CreateFileSystem` event and trigger an alarm to notify an administrator if the file system that was created was unencrypted.

Creating a Metric Filter

To create a CloudWatch alarm that is triggered when an unencrypted Amazon EFS file system is created, follow this procedure. You must have an existing trail created that is sending CloudTrail logs to a CloudWatch Logs log group. For more information, see Sending Events to CloudWatch Logs in the *AWS CloudTrail User Guide*.[15]

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Logs**.
3. In the list of log groups, choose the log group that you created for CloudTrail log events.
4. Choose **Create Metric Filter**.
5. On the **Define Logs Metric Filter** page, choose **Filter Pattern** and then type the following:

```
{ ($.eventName = CreateFileSystem) &&
($.responseElements.encrypted IS FALSE) }
```

6. Choose **Assign Metric**.
7. For **Filter Name**, type `UnencryptedFileSystemCreated`.
8. For **Metric Namespace**, type `CloudTrailMetrics`.
9. For **Metric Name**, type `UnencryptedFileSystemCreatedEventCount`.
10. Choose **Show advanced metric settings**.
11. For **Metric Value**, type `1`.
12. Choose **Create Filter**.

Creating an Alarm

After you create the metric filter, follow this procedure to create an alarm.

1. On the **Filters** for *Log_Group_Name* page, next to the **UnencryptedFileSystemCreated** filter name, choose **Create Alarm**.
2. On the **Create Alarm** page, set the parameters shown in Figure 2.

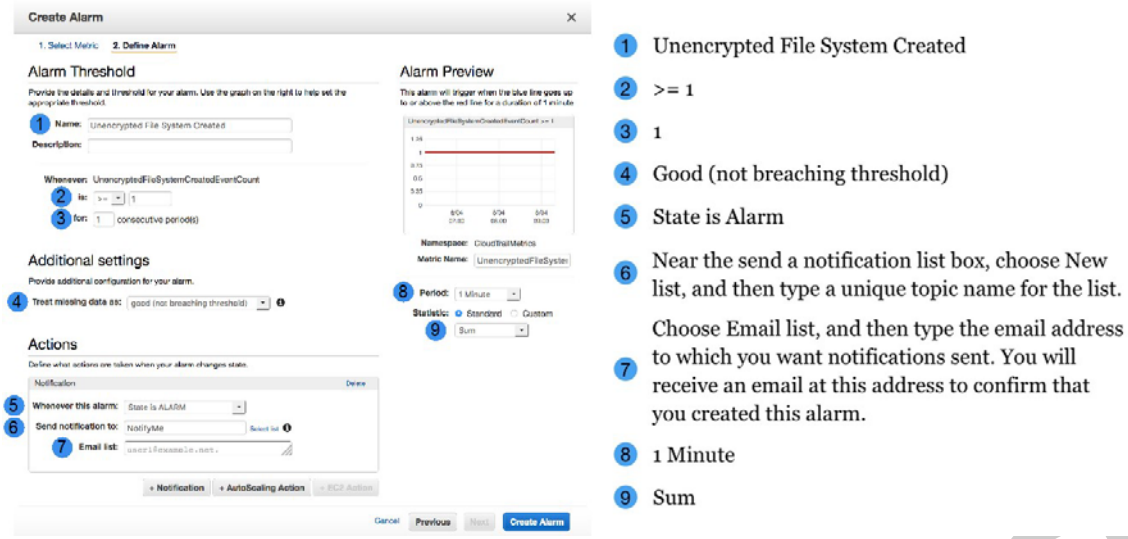

Figure 2: Create a CloudWatch alarm

3. Choose **Create Alarm**.

Testing the Alarm for Unencrypted File System Created

You can test the alarm by creating an unencrypted file system, as follows.

1. Open the Amazon EFS console at https://console.aws.amazon.com/efs.
2. Choose **Create File System**.
3. From the **VPC** list, choose your default VPC.
4. Select the check boxes for all the Availability Zones. Be sure that they all have the default subnets, automatic IP addresses, and the default security groups chosen. These are your mount targets.
5. Choose **Next Step**.
6. Name your file system and keep **Enable encryption** unchecked to create an unencrypted file system.
7. Choose **Next Step**.
8. Choose **Create File System**.

Your trail logs the `CreateFileSystem` operation and delivers the event to your CloudWatch Logs log group. The event triggers your metric alarm and CloudWatch Logs sends you a notification about the change.

Encryption of Data in Transit

You can mount a file system so all NFS traffic is encrypted in transit using Transport Layer Security 1.2 (TLS, formerly called Secure Sockets Layer [SSL]) with an industry-standard AES-256 cipher. TLS is a set of industry-standard cryptographic protocols used for encrypting information that is exchanged over the wire. AES-256 is a 256-bit encryption cipher used for data transmission in TLS. If your organization is subject to corporate or regulatory policies that require encryption of data and metadata in transit, we recommend setting up encryption in transit on every client accessing the file system.

Setting up Encryption of Data in Transit

The recommended method to setup encryption of data in transit is to download the EFS mount helper on each client. The EFS mount helper is an open-source utility that AWS provides to simplify using EFS, including setting up encryption of data in transit. The mount helper uses the EFS recommended mount options by default.

1. Install the EFS mount helper
    - Amazon Linux:

```
sudo yum install -y amazon-efs-utils
```

    - Other Linux distributions: download from GitHub (https://github.com/aws/efs-utils) and install
    - Supported Linux distributions:
        - Amazon Linux 2017.09+
        - Amazon Linux 2+
        - Debian 9+
        - Red Hat Enterprise Linux / CentOS 7+
        - Ubuntu 16.04+
    - The amazon-efs-utils package automatically installs the following dependencies:

- NFS client (nfs-utils)
- Network relay (stunnel)
- Python

2. Mount the file system:

```
sudo mount -t efs -o tls file-system-id efs-mount-point
```

- `mount -t efs` invokes the EFS mount helper
- Using the DNS name of the file system or the IP address of a mount target is not supported when mounting using the EFS mount helper, use the file system id instead.
- The EFS mount helper uses the AWS recommended mount options by default. Overriding these default mount options is not recommended but we provide the flexibility to do so when the occasion arises. We recommend thoroughly testing any mount option overrides so you understand how these changes impact file system access and performance.
- Below are the default mount options used by the EFS mount helper.
  - nfsvers=4.1
  - rsize=1048576
  - wsize=1048576
  - hard
  - timeo=600
  - retrans=2

3. Use the file fstab to automatically remount your file system after any system restart.
   - Add the following line to /etc/fstab

```
file-system-id efs-mount-point efs _netdev,tls 0 0
```

Using Encryption of Data in Transit

If your organization is subject to corporate or regulatory policies that require encryption of data in transit, we recommend using encryption of data in transit on every client accessing the file system. Encryption and decryption is configured at the connection level and adds another layer of security. Mounting the file system using the EFS mount helper sets up and maintains a TLS 1.2 tunnel between the client and the Amazon EFS service, and routes all NFS traffic over this encrypted tunnel. The certificate used to establish the encrypted TLS connection is signed by the Amazon Certificate Authority (CA) and trusted by most modern Linux distributions. The EFS mount helper also spawns a watchdog process to monitor all secure tunnels to each file system and ensures they are running. After using the EFS mount helper to establish encrypted connections to Amazon EFS, no other user input or configuration is required. Encryption is transparent to user connections and applications accessing the file system.

After successfully mounting and establishing an encrypted connection to an EFS file system using the EFS mount helper, the output of a `mount` command shows the file system is mounted and an encrypted tunnel has been established using the localhost (127.0.0.1) as the network relay. See sample output below.

```
127.0.0.1:/ on efs-mount-point type nfs4
(rw,relatime,vers=4.1,rsize=1048576,wsize=1048576,namlen=255,har
d,proto=tcp,port=20059,timeo=600,retrans=2,sec=sys,clientaddr=12
7.0.0.1,local_lock=none,addr=127.0.0.1)
```

To map an ***efs-mount-point*** to an EFS file system, query the mount.log file in /var/log/amazon/efs and find the last successful mount operation. This can be done using a simple grep command like the one below.

```
grep -E "Successfully mounted.*efs-mount-point"
/var/log/amazon/efs/mount.log | tail -1
```

The output of this grep command will return the DNS name of the mounted EFS file system. See sample output below.

```
2018-03-15 07:03:42,363 - INFO - Successfully mounted
file-system-id.efs.region.amazonaws.com at efs-mount-point
```

Conclusion

Amazon EFS file system data can be encrypted at rest and in transit. You can encrypt data at rest by using CMKs that you can control and manage using AWS KMS. Creating an encrypted file system is as simple as selecting a check box in the Amazon EFS file system creation wizard in the AWS Management Console, or adding a single parameter to the `CreateFileSystem` operation in the AWS CLI, AWS SDKs, or Amazon EFS API. Using an encrypted file system is also transparent to services, applications, and users, with minimal effect on the file system's performance. You can encrypt data in transit by using the EFS mount helper to establish an encrypted TLS tunnel on each client, encrypting all NFS traffic between the client and the mounted EFS file system. Encryption of both data at rest and in transit is available to you at no additional cost.

Contributors

The following individuals and organizations contributed to this document:

- Darryl S. Osborne, storage specialist solutions architect, AWS
- Joseph Travaglini, sr. product manager, Amazon EFS

Further Reading

For additional information, see the following:

- AWS KMS Cryptographic Details Whitepaper[16]
- Amazon EFS User Guide[17]

Document Revisions

Date	Description
April 2018	Added encryption of data in transit

Date	Description
September 2017	First publication

Notes

[1] https://aws.amazon.com/efs/

[2] https://aws.amazon.com/kms/

[3] https://docs.aws.amazon.com/efs/latest/ug/API_CreateFileSystem.html

[4] https://aws.amazon.com/tools/ - sdk

[5] https://aws.amazon.com/iam/

[6] https://docs.aws.amazon.com/kms/latest/developerguide/control-access-overview.html

[7] https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html

[8] https://docs.aws.amazon.com/kms/latest/developerguide/control-access-overview.html - managing-access

[9] https://d0.awsstatic.com/whitepapers/KMS-Cryptographic-Details.pdf

[10] https://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started_create-admin-group.html

[11] https://docs.aws.amazon.com/kms/latest/developerguide/key-policy-modifying.html - key-policy-modifying-external-accounts

[12] https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html - master_keys

[13] https://docs.aws.amazon.com/cli/latest/userguide/installing.html

[14] https://docs.aws.amazon.com/efs/latest/ug/whatisefs.html

[15] https://docs.aws.amazon.com/awscloudtrail/latest/userguide/send-cloudtrail-events-to-cloudwatch-logs.html

[16] https://aws.amazon.com/whitepapers/

[17] https://docs.aws.amazon.com/efs/latest/ug/whatisefs.html