Securing OpenEMR with Disk Encryption

Prepared by
ViSolve Inc.,
15th December 2016

ViSolve, Inc. 6559, Springpath Lane, San Jose, CA 95120, USA Tel: 408 850 2243

Email: services@visolve.com Website: www.visolve.com/hc

Contents

Document Objective:

When it comes to medical and patient records, security and the guidelines related to security take up a major role in protecting the patient and medical records. The data that is handled within the EMR can be either based on network (cloud) or local server. Electronic applications hosted with the cloud environment are almost configured with the service provides enterprises that are HIPAA compliant. These enterprises take care of privacy and other security related issues. But, what about the Electronic Medical Application servers that are hosted locally or running up in a closed environment that is not hosted in the cloud.? How is the data security ensured when the data is as at rest within the disk.

This document speaks about the encryption technique that could be followed up to secure the data residing at the disk is secure.

Business Problem

Assuming the EMR application is hosted at a local environment and data stored at a local disk (not connected to internet). The chances of losing the disk, anonymous entry within the hard disk, hard disk theft are some of the common possible loopholes to take control of the patient and medical record from the system. This takes into a serious security thread for the medical record stored within the local disk.

Proposed Solution

Considering the problem with having the data at rest, the proposed solution helps the user to protect the data from security risks during rest. Here the data at rest refers to the data that is not currently used or active with the environment.

This can be achieved in three different techniques.

- Application level encryption
- Database level encryption and
- File level encryption

With the below mentioned sections, the steps to protect the data at disk is protected through disk encryption.

Before getting into the document section directly, the following is the high level summary that we have tried to encrypt and retrieve data from the encrypted disk.

Initially the data to be secured is encrypted using an encryption algorithm. A virtual directory is created and the encrypted drive is mounted to the newly created virtual directory.

Through this process, the data that is to be fetched from the application is accessed from the encrypted disk using a key and the same is decrypted, processed with the systems local memory and viewed through the application.

This ensures the protection of data at rest and blocks the anonymous access to EMR or patient data.

[**Note:** All the following steps are to be followed on the latest Ubuntu OS]

Partition of the file system with encryption:

In order to encrypt sensitive data and improving its hardness at rest can be achieved by enabling a security tool ecryptfs-utils.

Install ecryptfs using below command:

```
apt-get install ecryptfs-utils
```

Creating folder inside /var/lib/ to encrypt the data, entering below command

```
mkdir folder1 folder2
```

It will ask to set password which is used to decrypt the data. Note down the password securely, else it leads to lose the file and its data.

Mounting the encrypted partition:

Initially we process the required permission for the associated folders. Here folder1 and folder2 are the two secure folders where we progress to have the secure data.

Enter the below mentioned commands such that the newly created folders gets access to MySQL.

Ensure that the current directory is set to /var/lib/.

```
chown mysql:mysql folder1

chown mysql:mysql folder2
```

Enter the below mentioned command to mount:

```
mount -t ecryptfs folder1 folder2
```

The above step mounts the encrypted data to the other folder such that the application reads the secured data.

When doing this for the first time, it's required to fill out encryption specification.

Set and verify the passphrase for encryption.

Select the format to encrypt the data, choose cipher 'aes'.

Select key bytes to encrypt the data, choose 32 bytes.

Set No to plain text passthrough, type n.

Set No to file name encryption , type n.

Next, It tells that directory is attempting to mount with above specification. Finally it ask conformation to proceed with mount and append the sign file, type 'yes' to mount the directory.

Installing and Configuring MySQL in secure partition:

The below mentioned steps guides to install MySQL and configure the database to the secure directory holding the application data.

[Note: If MySQL has already installed, skip the below MySQL installation steps]

Install MySQL using below command:

```
apt-get install mysql-server

mysql_secure_installation
```

Start MySQL using below command:

```
systemctl mysql start
```

Following are the steps to configure MySQL to new directory.

The Default installation directory of MySQL is '/var/lib/mysql/'. Change this directory to a new secure directory '/folder1/'

- Stop MySQLl Service

  Shut down MySQL before making changes in data directory by using the below command

```
systemctl stop mysql
```

- Move the existing database into a new directory by using the below command,

```
rsync -av /var/lib/mysql /files/ /var/lib/folder2/
```

- Edit mysqld.conf file to point out location changes.

```
nano Check /etc/mysql/mysql forthelinedatadir= .conf.d/mysqld inmysqld .cnf .cnf and change it to point to the new
```

directory. Once completed, save the file and exit.

datadir=/var/lib/folder2/

File: /etc/mysql/mysql.conf.d/mysqld.cnf

```
[mysqld]
#
# * Basic Settings
#
user            = mysql
pid-file        = /var/run/mysqld/mysqld.pid
socket          = /var/run/mysqld/mysqld.sock
port            = 3306
basedir         = /usr
datadir         = /var/lib/mysql
tmpdir          = /tmp
lc-messages-dir = /usr/share/mysql
skip-external-locking
```

Note: With the above picture, change the /var/lib/mysql to /var/lib/folder2

- Configure apparmor to allow access.

Configure apparmor to tell the changes in data directory. These changes are progressed such that the SQL fetches the required data from the newly configured files.

- Open the terminal enter below command to configure apparmor profile.

```
vim /etc/apparmor.d/usr.sbin.mysqld
```

Add the new directory location in the apparmor.d as follow,

Adding data directory access:

```
Folder1/files/ r,

Folder1/files/** rwk,

Folder2/files/ r,

Folder2/files/** rwk,
```

```
# Allow data dir access
  /var/lib/mysql/ r,
  /var/lib/mysql/** rwk,
  Folder1/files/ r,
  Folder1/files/**  rwk,
  Folder2/files/ r,
  Folder2/files/**  rwk,
```

Addding data files directory access :

Folder1/files/mysql-files/ r,

*Folder1/files/mysql-files** rwk,*

Folder2/files/mysql-files r,

*Folder2/files/mysql-files/** rwk,*

```
# Allow data files dir access
  /var/lib/mysql-files/ r,
  /var/lib/mysql-files/** rwk,
  Folder1/files/mysql-files/ r,
  Folder1/files/mysql-files**  rwk,
  Folder2/files/mysql-files r,
  Folder2/files/mysql-files/**  rwk,
```

Adding keyring directory access:

Folder1/files/myfiles/mysql-keyring/-keyring/r, r,

*Folder1/files/myfiles/mysql-keyring**-keyring**rwk, rwk,*

Folder2/files/myfiles/mysql-keyring-keyringr, r,

*Folder2/files/mysqlfiles/mysql -keyring/** rwk -keyring/** rwk*

```
# Allow keyring dir access
  /var/lib/mysql-keyring/ r,
  /var/lib/mysql-keyring/** rwk,
  Folder1/files/mysql-keyring/ r,
  Folder1/files/mysql-keyring**  rwk,
  Folder2/files/mysql-keyring r,
  Folder2/files/mysql-keyring/**  rwk
```

- Restart the Apparmor by entering below command.

```
/etc/init.d/apparmor restart
```

- Restart MySQL by entering below command,

```
/etc/init.d/mysql restart
```

Unmounting the disk and having the data over other disk:

Prior to unmount, it's necessary to stop MySQL service, running below command will stop the MySQL service,

```
systemctl mysql stop
```

Enter the below command to unmount the directory:

```
umount folder2
```

If it's required to mount again enter the below mentioned command, which includes all encryption specification in the single command.

```
mount -t ecryptfs folder1 folder2 -o
ecryptfs_unlink_sigs,ecryptfs_key_bytes=32,ecryptfs_cipher=aes,ecryptfs_passthrough=n,ecryptfs_ena
ble_filename_crypto=n
```

Disclaimer

Whilst every effort has been made to ensure the accuracy of all information and statements contained in this document, the functionality, the service levels, performance and capabilities referred to are best estimates and best practices only, based on the general understanding of ViSolve Consultants. Contents of this document are subject to change without notice.

ViSolve does not hold any responsibilities for any kind of Confidentiality or Security standards with the current document. The development, release, and timing of any features or functionality in this document remains at the sole discretion of ViSolve Inc. All copyrights and trademarks are the property of their respective owners

Proprietary Notice

For Services and Support Contact:

ViSolve, Inc.
6559, Springpath Lane,
San Jose,
California - 95120.
Tel : +1-408-850-2243
Email : services@visolve.com
Skype : visolve.inc