AltiusIT

Encryption Security Recommendations

Basic Concepts

Sensitive data should be encrypted while in transit and stored.

All communication between clients and servers, and between servers (Web server to app server, app server to database server, etc.) should be authenticated and encrypted. Sensitive data should never be sent over http/ftp since information is sent in clear text.

Recommended encryption strength minimums:

- Certificates: 256-bit SHA-2
- Block Encryption: 256-bit AES
- Hash: SHA-512

Certificates

Certificate options in order of preference:

- Internal PKI
- Third Party signed certificates for all servers
- Self-signed certificates - should be avoided due to their insecure nature

Application Level (My SQL Database Server / Application Server)

The default encryption methods of MySQL only use 128bit encryption. This is not considered to be strong encryption. Furthermore, the MySQL AES_ENCRYPT() functions are also potentially not secure because they haven't documented how they hash the password into the key and the cipher mode used. This may cause the encryption to be significantly weaker depending on the actual implementation. While it is possible to make changes to the source of MySQL and recompile, Altius IT recommends that data be encrypted prior to being sent to MySQL.

Network Level (MySQL Server / Application server)

OpenSSL should be used for all connections to the database. This provides a centralized method for managing keys. Most standard MySQL packages of the common Linux distributions already offer OpenSSL-enabled MySQL services out of the box. If not, compile the sources of MySQL manually and run the configure script with the option --with-vio --with-openssl. OpenSSL activation forces the environment variable have_openssl to be set to YES. This can be checked by:

```
mysql > SHOW VARIABLES LIKE ’%openssl%’;
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_openssl  | YES   |
+---------------+-------+
```

Since the OpenSSL encryption implementation of MySQL sustains upon certificates, we need to create

1. Certificate Authority (CA) key and certificate,
2. Server encryption key, as well as a server certificate request,
3. Client encryption key, as well as a client certificate request.

The following shellscript will do the above (OpenSSL binaries needs to be installed):

```
#!/bin/bash
DIR=`pwd`/openssl
PRIV=$DIR/private
mkdir $DIR $PRIV $DIR/newcerts
cp /usr/lib/ssl/openssl.cnf $DIR
replace ./demoCA $DIR -- $DIR/openssl.cnf
openssl req -new -x509 -keyout $PRIV/cakey.pem
  -out $DIR/cacert.pem -config $DIR/openssl.cnf
openssl req -new -keyout $DIR/server-key.pem
  -out $DIR/server-req.pem -days 3600 -config $DIR/openssl.cnf
openssl rsa -in $DIR/server-key.pem -out $DIR/server-key.pem
openssl ca  -policy policy_anything
  -out $DIR/server-cert.pem -config $DIR/openssl.cnf -infiles $DIR/server-req.pem
openssl req -new -keyout $DIR/client-key.pem
  -out $DIR/client-req.pem -days 3600 -config $DIR/openssl.cnf
openssl rsa -in $DIR/client-key.pem -out $DIR/client-key.pem
openssl ca  -policy policy_anything
  -out $DIR/client-cert.pem -config $DIR/openssl.cnf -infiles $DIR/client-req.pem
```

Line numbers

- Lines 1 - 6 create a useable directory structure for storing the resulting keys and certificates. Be sure to call this script from a safe location; keys are normally stored in /etc/mysql/keys or something similar.
- Line 7 and 8 generate a local Certificate Authority for signing the certificates which are to be created.
- Lines 9 and 10 create an encryption key for the MySQL server and a certificate request, which is to be signed afterwards. The certificate will be valid for 3600 days.
- Line 11 (and line 16) is optional and would remove the passphrase from the server key. This means that it’s not necessary to give the passphrase every time the MySQL server is restarted. This behavior may be seen as security risk, depending on where the (unencrypted) key will be stored.
- Lines 12 and 13 will sign the previously generated server certificate with our local CA instance.
- Lines 14 and 15 create a client key and certificate request.
- The last lines sign the client certificate with our local CA instance.

We finally have to tell MySQL where our encryption keys and certificates are stored, which is done in my.cnf. We need entries for both, server and client. Note that the client configuration as well as the client and CA certificates have to be available on all clients who wish to encrypt MySQL related network traffic.

```
    ssl-ca=$DIR/cacert.pem
    ssl-cert=$DIR/client-cert.pem
    ssl-key=$DIR/client-key.pem
    <...>
    [mysqld]
    ssl-ca=$DIR/cacert.pem
    ssl-cert=$DIR/server-cert.pem
    ssl-key=$DIR/server-key.pem
    <...>
```

$DIR is to be replaced by the chosen key and certificate directory.

Server (block level) Level (all servers)

Block level encryption should be used for all servers that contain sensitive information. The following are two examples of how this can be accomplished:
Linux: dmcrypt

The following steps need to have losetup and cryptsetup installed on the System, as well as a kernel which has been built with CONFIG_DM_CRYPT and CONFIG_BLK_DEV_DM support (which most of the current kernels have). Most Unices offer the use of encryption, but most of them are not platform independent.

MySQL stores its data in the $MYSQL_CHROOT/data directory, we will now encrypt. We will proceed with the following steps:

1. We generate a file with completely randomized content, with the maximum size of the MySQL storage tables (in the following example, 100MiB). If the reserved space points out to be too few, we can simply create a bigger one and transfer the encrypted data later.
2. We create a new loopback-device, which is capable of handling our encrypted data-image as hard disk partition.
3. We connect the loopback-device with a so called crypto-target, which encrypts everything which is written onto the target, and decrypts everything which is read from the target, as long as the crypto-target is enabled.
4. Format the encrypted data container with a filesystem of our choice (ReiserFS in this case).
5. Mount the encrypted container, as it’s ready to use.

These above steps are performed via the following commands:

```
    # dd if=/dev/urandom of=$MYSQL_CHROOT/data.crypt
    # losetup /dev/loop0 $MYSQL_CHROOT/data.crypt
    # cryptsetup -y create mysql_data /dev/loop0
     Enter passphrase: Passphrase
     Verify passphrase: Passphrase
    # mkreiserfs /dev/mapper/mysql_data
    # mount /dev/mapper/mysql_data $MYSQL_CHROOT/data
```

Before starting up the MySQL database server for everyday use, enforce step 2, 3 and 7. Detailed information about the theoretical backgrounds to cryptography may be found in the wonderful reference of Bruce Schneier [Sch05], as well as [Ert03] and [Wae03]. Information on practical file system encryption is found in [Pac05].

TrueCrypt

The installation of TrueCcrypt is quite easy since precompiled binaries are available for all supported platforms, including Linux binaries as well as Debian (and Ubuntu) packages.

When starting the software, an easy-to-use graphical dialog appears which should be quite self-explanatory.

Like dmcrypt, also TrueCrypt offers two possibilities of creating encrypted volumes:

- Format a whole partition which is to be filled with encrypted content, or
- Creating a fixed-size container archive file; this file will again be looped back to a pseudo-device which can be accessed by the operating system just like a normal partition.

File level encryption. If sensitive data, either in production or backup, is stored in shared directories it should be encrypted using file level encryption. This can be accomplished using something like AES Crypt for Linux.

Refer to Altius IT's Recommended Software Security Controls for additional safeguards and security best practices.

Resources

Advanced Encryption Standard - www.aescrypt.com/linux_aes_crypt.html

National Institute of Standards and Technology:

- http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexa.pdf
- http://csrc.nist.gov/publications/nistpubs/800-66-Rev1/SP-800-66-Revision1.pdf

Open Web Application Security Project -
www.owasp.org/index.php/OWASP_Backend_Security_Project_MySQL_Hardening

TrueCrypt - www.truecrypt.org/docs/

Publication Information
Altius IT is a security audit, security consulting, and risk management firm. We are certified by the Information Systems Audit and Control Association (ISACA) as a Certified Information Systems Auditor (CISA), Certified in Risk and Information Systems Controls (CRISC), and Certified in the Governance of Enterprise Wide IT (CGEIT). For more information, please visit www.AltiusIT.com.