New database encryption cipher - AeadBase64Encryptor

1. Introduction

CloudStack has used StandardPBEStringEncryptor from jasypt for more than 10 years.

The encryptor use algorithm "PBEWithMD5AndDes", which is considered as Insecure, because it uses MD5 and DES which has only 56-bits key.

After investigation, we decided to replace it with an implementation of AES-GCM algorithm. The key length of AES-GCM is 256-bit. Please refer to https://github.com/google/tink

Main changes of this feature:

- Introduce new encryptor AeadBase64Encryptor, which is based on AesGcmJce
- Improve cloudstack-setup-databases to set up cloudstack database with different encryptors
- Improve cloudstack-migrate-database to migrate cloudstack properties and database with different management key, database key or encryptor version.
- Improve cloudstack startup process so that both legacy and new encryptors work. Existing environments work well without any change.

Community PR to main/4.18: https://github.com/apache/cloudstack/pull/7003

2. New encryptors

There are 4 new classes introduced in this feature:

- Base64Encryptor:
  - interface of cloudstack encryptors
  - There are two abstract methods: (1) encrypt ; (2) decrypt
  - The output of encrypt and input of decrypt are base64-encoded.
- LegacyBase64Encryptor
  - an implementation of Base64Encryptor
  - It uses "org.jasypt.encryption.pbe.StandardPBEStringEncryptor" to decrypt/encrypt values
  - Same as current encryption
- AeadBase64Encryptor
  - another implementaionof Base64Encryptor
  - It uses "com.google.crypto.tink.subtle.AesGcmJce" to decrypt/encrypt values
- CloudStackEncryptor
  - The is the class ready for use by other java classes in cloudstack
  - properties
    - password: the passphrase used by encryptors
    - LegacyBase64Encryptor encryptorV1: the instance of LegacyBase64Encryptor
    - AeadBase64Encryptor encryptorV2: the instance of AeadBase64Encryptor
    - Base64Encryptor encryptor: the actual encryptor used in encryption and decryption
  - encrypt process
    - if encryptor is not null, use it to encrypt input value
    - if encryptio is null
      - use default encryptor (current encryptorV2) to encrypt input value
      - update encryptor to default encryptor (encryptorV2)
  - decrypt process
    - if encryptor is not null, use it to decrypt input value

        - if encryptio is null
            - use encryptorV2 to decrypt input value. if succeed, update encryptor to encryptorV2 and return result
            - if fail, use encryptorV1 to decrypt input value. if succeed, update encryptor to encryptorV1 and return result
            - if still fail, throw an CloudRuntimeException
    - This class is used by
        - DBEncryptionUtil
        - EncryptionSecretKeyChecker
        - EncryptionSecretKeyChanger
        - EncryptionCLI
        - EncryptionUtil

3. cloudstack-setup-databases changes

"/usr/bin/cloudstack-setup-databases" is used to setup

- db.properties (encrypted by management key)
- cloudstack database (encrypted by database key)

it has been updated to support

- encryption type: this is saved as db.cloud.encryption.type in /etc/cloudstack/management/db.properties
    - file: management key is saved as /etc/cloudstack/management/key
    - web:
    - env: cloudstack-setup-databases reads management key from enviroment variable "CLOUD_SECRET_KEY". (THIS IS NEW)

- encryptor version: This is saved as db.cloud.encryptor.version in /etc/cloudstack/management/db.properties (THIS IS NEW)
    - V1 : The encrypted values in db.properties are encrypted by management key using LegacyBase64Encryptor
    - V2 : The encrypted values in db.properties are encrypted by management key using AeadBase64Encryptor
    - empty/not specified: The encrypted values in db.properties are encrypted by management key using default encryptor (currently AeadBase64Encryptor)

Please note: com.cloud.utils.crypt.EncryptionCLI in /usr/share/cloudstack-common/lib/cloudstack-utils.jar is used to replace org.jasypt.intf.cli.JasyptPBEStringEncryptionCLI

3.1 Usage

```
# cloudstack-setup-databases -h
Usage: cloudstack-setup-databases user:[password]@mysqlhost:[port] [--deploy-as=rootuser:[rootpassword]] [--auto=/path/to/server-setup.xml] [-e ENCRYPTIONTYPE] [-m MGMTSECRETKEY] [-k DBSECRETKEY] [–debug]
```

```
Options:
  -h, --help            show this help message and exit
  -v, --debug           If enabled, print the commands it will run as they run
  -d ROOTCREDS, --deploy-as=ROOTCREDS
                        Colon-separated user name and password of a MySQL user
                        with administrative privileges
  -s, --schema-only     Creates the db schema without having to pass root
                        credentials - Please note: The databases (cloud,
                        cloud_usage) and user (cloud) has to be configured
                        manually prior to running this script when using this
                        flag.
  -a SERVERSETUP, --auto=SERVERSETUP
                        Path to an XML file describing an automated unattended
                        cloud setup
  -e ENCRYPTIONTYPE, --encrypt-type=ENCRYPTIONTYPE
                        Encryption method used for db password encryption.
                        Valid values are file, web and env. Default is file.
  -m MGMTSECRETKEY, --managementserver-secretkey=MGMTSECRETKEY
                        Secret key used to encrypt confidential parameters in
                        db.properties. A string, default is password
  -k DBSECRETKEY, --database-secretkey=DBSECRETKEY
                        Secret key used to encrypt sensitive database values.
                        A string, default is password
  -i MSHOSTIP, --mshost=MSHOSTIP
                        Cluster management server host IP. A string, by
                        default it will try to detect a local IP
  -r REGIONID, --regionid=REGIONID
                        Region Id for the management server cluster
  -c DBCONFPATH, --db-conf-path=DBCONFPATH
                        The path to find db.properties which hold db
                        properties
  -f DBFILESPATH, --db-files-path=DBFILESPATH
                        The path to find sql files to create initial
                        database(s)
  -j ENCRYPTIONJARPATH, --encryption-jar-path=ENCRYPTIONJARPATH
                        The cloudstack jar to be used to encrypt the values in
                        db.properties
  -n ENCRYPTIONKEYFILE, --encryption-key-file=ENCRYPTIONKEYFILE
                        The name of the file in which encryption key to be
                        generated
  -g ENCRYPTORVERSION, --encryptor-version=ENCRYPTORVERSION
                        The encryptor version to be used to encrypt the values
                        in db.properties
  -b MYSQLBINPATH, --mysql-bin-path=MYSQLBINPATH
                        The mysql installed bin path
```

3.2 Examples

- /usr/bin/cloudstack-setup-databases 'cloud':'Password'@'10.0.33.168' -i 10.0.33.168

create /etc/cloudstack/management/key (content of the file is the management key = password)

create /etc/cloudstack/management/db.properties with default passwords (management key = password, db key = password)

in db.properties (1) db.cloud.encryptor.version is not set; (2) db.cloud.encryption.type=file; (3) db.cloud.password, db.cloud.encrypt.secret, db.usage.password are encrypted by default encryptor (currently V2)

- /usr/bin/cloudstack-setup-databases 'cloud':'Password'@'10.0.33.168' --deploy-as=root:'Password' -i 10.0.33.168 -m mgmtkey -k dbkey --encryptor-version v1

create /etc/cloudstack/management/key (content of the file is the management key = mgmtkey)

create /etc/cloudstack/management/db.properties with keys (management key = mgmtkey, db key = dbkey)

in db.properties (1) db.cloud.encryptor.version=V1; (2) db.cloud.encryption.type=file; (3) db.cloud.password, db.cloud.encrypt.secret, db.usage.password are encrypted by encryptor V1.

recreate database using mysql user root:'Password'

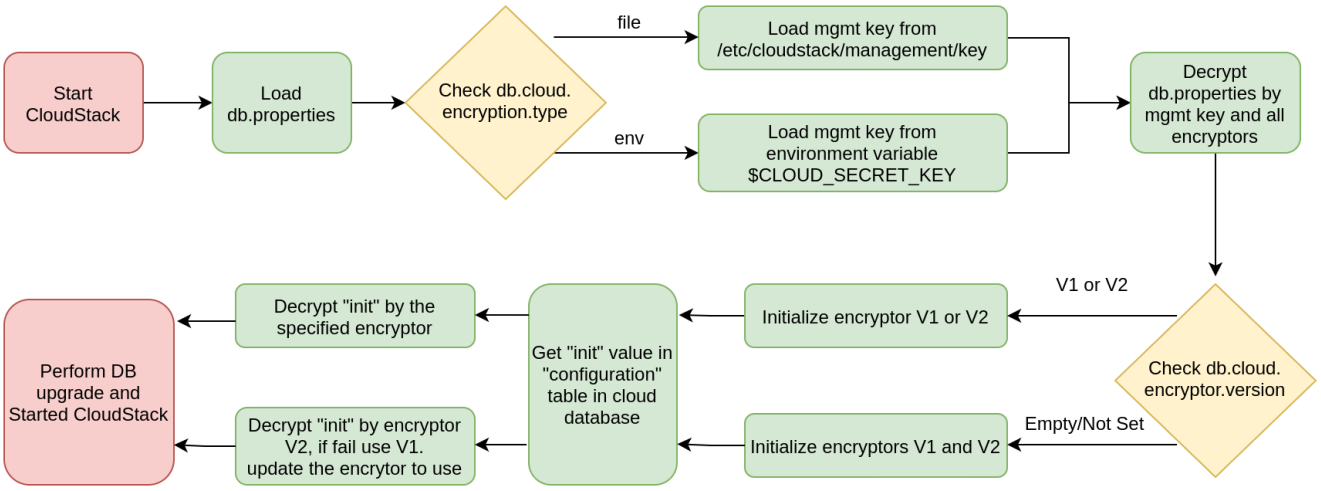
- /usr/bin/cloudstack-setup-databases 'cloud':'Password'@'10.0.33.168' --deploy-as=root:'Password' -i 10.0.33.168 --encryptor-version v2 -e env

create /etc/cloudstack/management/db.properties with keys (management key = $CLOUD_SECRET_KEY, db key = password)

in db.properties (1) db.cloud.encryptor.version=V2; (2) db.cloud.encryption.type=env; (3) db.cloud.password, db.cloud.encrypt.secret, db.usage.password are encrypted by encryptor V2;

recreate database using mysql user root:'Password'

4. New CloudStack startup process



5. cloudstack-migrate-databases changes

"/usr/bin/cloudstack-migrate-databases" are used to migrate the following with new keys/encryptor.

- db.properties (encrypted by management key)
- server.properties (encrypted by management key)
- cloudstack database (encrypted by database key)

It has been updated to support

- version: the new encryption version. Options are V1, V2
  - if it is null, the same encryptor version (=db.cloud.encryptor.version in db.properties) will be used for cloudstack database.
  - if the encryptor version is different, database will be migrated (unless skip-database-migration is passed).
  - if db.cloud.encryptor.version in db.properties is not set, the default encryptor (currently V2) will be used for cloudstack database.
  - Please note: db.properties and server.properties will always use the default encryptor (currently V2) in migration.
- force-database-migration
  - if DB secret key and encryptor version are not changed, DB values will not be migrated (decrypted/encrypted).
  - with this flag, database migration will be performed even if DB Secret key is not changed
  - this is useful for database migration if user want to use same DB secretkey but newer encryptor version.
- skip-database-migration
  - if DB secret key or encryptor version is changed, database migration will be performed by default .
  - with this flag, database migration will be skipped even if DB secret key or encryptor version is changed
  - this is useful for additional cloudstack management servers.
- load-new-management-key-from-env
  - load new management key from environment variable CLOUD_SECRET_KEY_NEW
  - this can be used to replace the option "-e,--newDBKey <newDBKey>"

5.1 Usage

```
# cloudstack-migrate-databases -h
usage: cloudstack-migrate-databases -d <oldDBKey> -m <oldMSKey> [-e <newDBKey>] [-n <newMSKey>] [-v
       <version>] [-f] [-h] [-l] [-s]
Options:
 -d,--oldDBKey <oldDBKey>                (required) Current DB Secret Key
 -m,--oldMSKey <oldMSKey>                (required) Current Mgmt Secret Key
 -e,--newDBKey <newDBKey>                New DB Secret Key
 -n,--newMSKey <newMSKey>                New Mgmt Secret Key
 -v,--version <version>                  New DB Encryptor Version. Options are V1, V2.
 -f,--force-database-migration           Force database migration even if DB Secret key is not
                                         changed
 -h,--help                               Show help message
 -l,--load-new-management-key-from-env   Load new management key from environment variable
                                         CLOUD_SECRET_KEY_NEW
 -s,--skip-database-migration            Skip database migration even if DB Secret key is changed
```

5.2 Examples

- cloudstack-migrate-databases -m password -d password -n newmgmtkey -v V2
  Migrate cloudstack properties (db.properties and server.properties) with new management key and encryptor V2.

- cloudstack-migrate-databases -m password -d password -n newmgmtkey -e newdbkey
  Migrate cloudstack properties and databases with new management key and database secret key.

- cloudstack-migrate-databases -m password -d password -n newmgmtkey -e newdbkey -s -v V2
  Migrate cloudstack properties with new keys and encryptor V2, but skip database migration.

- cloudstack-migrate-databases -m password -d password -l -f
  Migrate cloudstack properties with new management key (load from $CLOUD_SECRET_KEY_NEW), and migrate database with old db key.

Return codes:
0 - Succeed to change keys and/or migrate databases
1 - Fail to parse the command line arguments
2 - Fail to validate parameters
3 - Fail to migrate database

5.3 Table and Column changes during migration

The following tables and columns will be migrated during database migration.

Table	Coiumn	Column type	Conditions	Note
configuration	value	varchar (8191)	category IN ('Hidden', 'Secure')	
host_details	value	varchar (255)	name = "password"	
cluster_details	value	varchar (255)	name = "password"	
storage_pool_details	value	varchar (255)	name in ("password", "powerflex.gw.username", "powerflex.gw.password")	ScaleIO
image_store_details	value	varchar (255)	name in ("key", "secretkey")	
user_vm_details	value	varchar (5120)	name = "password"	
user_vm_deploy_as_is_details	value	varchar (255)	(1) get OVFPropertyTO from template_deploy_as_is_details by detail name (2) migrate value if property is password.	
storage_pool	path	varchar (255)	pool_type = 'SMB'	
image_store	url	varchar (2048)	protocol = 'cifs'	
vpn_users	password	varchar (255)		

sslcerts	password	varchar (255)		
sslcerts	key	text		
account_details	value	varchar (255)		if https://github.com/apache/cloudstack/pull/6812 is merged, extra methods need to be added to cloudstack-migrate-databases to migrate the values (1) in account_details and domain_details (2) the configs are Hidden/Secure global configurations.
domain_details	value	varchar (255)		
s2s_customer_gateway	ipsec_psk	varchar (256)		
virtual_supervisor_module	password	varchar (255)		
ucs_manager	password	varchar (255)		
vm_instance	vnc_password	varchar (255)		
keystore	key	text		
passphrase	passphrase	varchar(64)		Need test: volume encryption Modify to varchar(255)
external_stratosphere_ssp_credentials	password	varchar (255)		
vmware_data_center	password	varchar (255)		
storage_pool	user_info	varchar (255)		
remote_access_vpn	ipsec_psk	varchar (256)		
user	secret_key	varchar (255)		
oobm	password	varchar (255)		

6. EncryptionCLI in cloudstack-utils.jar

A standalone jar package cloudstack-utils.jar is shipped with cloudstack-common. It will be placed as /usr/share/cloudstack-common/lib/cloudstack-utils.jar.

with the jar, we can encrypt and decrypt values using cloudstack encryptors.

6.1 Usage

```
# java -classpath /usr/share/cloudstack-common/lib/cloudstack-utils.jar com.cloud.utils.crypt.EncryptionCLI
Missing required options: i, p
usage: Usage:
 -d,--decrypt                          Decrypt instead of encrypt
 -e,--encryptorversion <encryptorversion>   The encryptor version
 -i,--input <input>                    The input string to process
 -p,--password <password>                   The encryption password
 -v,--verbose                          Verbose output
```

6.2 Examples

- encrypt db secret key using management key (using default encryptor V2)

```
# java -classpath /usr/share/cloudstack-common/lib/cloudstack-utils.jar com.cloud.utils.crypt.EncryptionCLI -p mgmtkey -i dbkey

+X4wvxJcDzp2DhD8tUyyJa+4Ije5VAELOYC75QfEQBGI
```

- decrypt encrypted db secret key (encrypted by V2) using management key

```
# java -classpath /usr/share/cloudstack-common/lib/cloudstack-utils.jar com.cloud.utils.crypt.EncryptionCLI -p mgmtkey -i
+X4wvxJcDzp2DhD8tUyyJa+4Ije5VAELOYC75QfEQBGI -d

dbkey
```

- encrypt db secret key using management key (using encryptor V1)

```
# java -classpath /usr/share/cloudstack-common/lib/cloudstack-utils.jar com.cloud.utils.crypt.EncryptionCLI -p mgmtkey -i dbkey -e V1

sjlu6z6CSYCKitpVqgG0rw==
```

- decrypt encrypted db secret key (encrypted by V1) using management key with encreyptor V2

```
# java -classpath /usr/share/cloudstack-common/lib/cloudstack-utils.jar com.cloud.utils.crypt.EncryptionCLI -p mgmtkey -i sjlu6z6CSYCKitpVqgG0rw== -d -
e V2

Exception in thread "main" com.cloud.utils.exception.CloudRuntimeException: Error decrypting value: sjlu6*****
    at com.cloud.utils.crypt.CloudStackEncryptor.decrypt(CloudStackEncryptor.java:107)
    at com.cloud.utils.crypt.EncryptionCLI.main(EncryptionCLI.java:50)
Caused by: com.cloud.utils.crypt.EncryptionException: Failed to decrypt sjlu6z6CSYCKitpVqgG0rw==. Error: ciphertext too short
    at com.cloud.utils.crypt.AeadBase64Encryptor.decrypt(AeadBase64Encryptor.java:40)
    at com.cloud.utils.crypt.CloudStackEncryptor.decrypt(CloudStackEncryptor.java:105)
    ... 1 more
```

- decrypt encrypted db secret key (encrypted by V1) using management key with all encreyptors (try with V2, if fail try with V1)

```
# java -classpath /usr/share/cloudstack-common/lib/cloudstack-utils.jar com.cloud.utils.crypt.EncryptionCLI -p mgmtkey -i sjlu6z6CSYCKitpVqgG0rw== -d

dbkey
```

- decrypt encrypted db secret key (encrypted by V1) using management key with encreyptor V1

```
# java -classpath /usr/share/cloudstack-common/lib/cloudstack-utils.jar com.cloud.utils.crypt.EncryptionCLI -p mgmtkey -i sjlu6z6CSYCKitpVqgG0rw== -d -
e V1

dbkey
```

7. References

- https://github.com/google/tink
- http://www.jasypt.org/api/jasypt/1.8/org/jasypt/encryption/pbe/StandardPBEStringEncryptor.html