PERCONA
TRAINING

Disk-Level Encryption

http://www.percona.com/training/

Disk-Level Encryption

OVERVIEW

Introduction

- Security, security, security
- Clients in the PCI, HIPAA, PHI, PII space
- Similar rules and restrictions in other parts of the world
- Demand that data be encrypted "at rest"

States of Data

- Data at Rest
  - Fuzzy definition.
  - Cold storage.
  - In database.
- Data in Use
  - Residing in memory
  - Accessible
- Data in Transit
  - Over the wire

Linux Unified Key Setup (LUKS)

- Created by Clemens Fruhwirth in 2004
- Implemented using the linux device mapper (dm) layer
  - Abstracts any block device to software
- Can encrypt:
  - Removable media
  - RAIDs
  - LVMs
  - Physical, Logic and File-based disks

(*) https://en.wikipedia.org/wiki/Dm-crypt

Disk-Level Encryption

LUKS SETUP

The Deep End!

- Example block device: */dev/sda4*
- The key can be securely stored in a LastPass Secure Note or other secure vault.
- The key is not directly recoverable.
- **WARNING! Loss of the key means complete loss of data! Be sure to have a backup of the key.**
  - See above!
  - Make a backup of the block headers!

Creating a passphrase

- Need something with high entropy
  - Lots of randomness

```
# openssl rand -base64 32

# date | md5 | rev | head -c 24 | md5 | tail -c 32
```

Creating an Encrypted Disk

- Format the disk
  - Unmount beforehand

```
# umount /var/lib/mysql

# cryptsetup -c aes-xts-plain -v luksFormat /dev/sda4

WARNING!
========
This will overwrite data on /dev/sda4 irrevocably.

Are you sure? (Type uppercase yes): YES

Enter passphrase:
Verify passphrase:

Command successful.
```

"Opening" the Disk

- After you provide a passphrase, you now need to "open" the encrypted disk
  - This becomes the device mapper name/alias.
- It can be anything, but for our purposes, we will call it “mysqldata.”

```
# cryptsetup luksOpen /dev/sda4 mysqldata
Enter passphrase for /dev/sda4:
```

- On success, you should see the device show up:

```
# ls /dev/mapper/

lrwxrwxrwx  1 root root       7 Nov  4 11:50 mysqldata -> ../dm-0
```

Format a Filesystem

- You can now format this encrypted block device and create a filesystem:

```
# mkfs.ext4 /dev/mapper/mysqldata
```

- You can now mount the encrypted block device you just formatted:

```
# mount /dev/mapper/mysqldata /var/lib/mysql

# df -h
...
/dev/mapper/mysqldata          26G   45M   25G   1% /var/lib/mysql
```

Gotchas / Caveats

- Unfortunately, you cannot add this to /etc/fstab to automount since the key is needed.
- Do keep this in mind if your server ever reboots
  - MySQL will not start since the data directory is unavailable until opened and mounted.

Make a Backup Now!

- The header of the block device contains information regarding the current encryption key(s).
- Should this ever get damaged, or if you need to recover because you forgot the new password, you can restore this header.

```
# cryptsetup luksHeaderBackup --header-backup-file \
    ${HOSTNAME}_`date +%Y%m%d`_header.dat /dev/sda4

# file node2_20170426_header.dat
node2_20170426_header.dat: LUKS encrypted file, ver 1
    [aes, xts-plain, sha256] UUID: 611391ee-c182-44ab-bbb4-f1ec76fd9fe1

# sha1sum ${HOSTNAME}_`date +%Y%m%d`_header.dat
4a48f2eaf1c65893d2d82681d393690d0a704e0a  node2_20170426_header.dat
```

- GZip the header file. Store the SHA1 and the .gz file in a secure location.

Unmounting and Closing a Disk

- If you know you will be storing a disk and want to make sure the contents are not currently visible (ie: mounted), you can unmount and "close" the encrypted device.

```
# systemctl stop mysql
# umount /var/lib/mysql
# cryptsetup luksClose mysqldata
```

- In order to mount this device again you will be required to "open" it and provide one of the keys.

Disk-Level Encryption

ROTATING KEYS

Rotating Keys (Adding / Removing Keys)

- Various compliance and enforcement rules will dictate how often you need to rotate keys.
- You cannot rotate or change a key directly.
- LUKS supports up to 8 keys per device.
- You must first add a new key to any other slot and then remove the older key.

Rotating Example

- Take a look at the existing header information:

```
# cryptsetup luksDump /dev/sda4

LUKS header information for /dev/sda4
Version: 1
Cipher name: aes
Cipher mode: cbc-essiv:sha256
Hash spec: sha1
Payload offset: 4096
MK bits: 256
MK digest: 81 37 51 6c d5 c8 32 f1 7a 2d 47 7c 83 62 70 d9 f7 ce 5a 6e
MK salt: ae 4b e8 09 c8 7a 5d 89 b0 f0 da 85 7e ce 7b 7f
47 c7 ed 51 c1 71 bb b5 77 18 0d 9d e2 95 98 bf
MK iterations: 44500
UUID: 92ed3e8e-a9ac-4e59-afc3-39cc7c63e7f6
Key Slot 0: ENABLED
Iterations: 181059
Salt: 9c a9 f6 12 d2 a4 2a 3d a4 08 b2 32 b0 b4 20 3b
69 13 8d 36 99 47 42 9c d5 41 35 8c b3 d0 ff 0e
Key material offset: 8
AF stripes: 4000
Key Slot 1: DISABLED
Key Slot 2: DISABLED
Key Slot 3: DISABLED
Key Slot 4: DISABLED
Key Slot 5: DISABLED
Key Slot 6: DISABLED
Key Slot 7: DISABLED
```

Rotating Example (cont.)

- We can see a key is currently occupying 'Key Slot 0'.
- We can add a key to any *DISABLED* key slot.

```
# cryptsetup luksAddKey --key-slot 1 -v /dev/sda4

Enter any passphrase:
Key slot 0 unlocked.

Enter new passphrase for key slot:
Verify passphrase:
Command successful.
```

- LUKS will accept 'any' passphrase to authenticate us.
- Had there been other keys in other slots, we could have used any one of them.
  - As only one was currently saved, we had to use it.
  - We can then add a new passphrase for slot 1.

Rotating Example (cont.)

- Now that we have saved the new key in slot 1, we can remove the key in slot 0.

```
# cryptsetup luksKillSlot /dev/sda4 0

Enter any remaining LUKS passphrase:
No key available with this passphrase.
```

- In the example above, the existing password stored in slot 0 was used.
  - This is not allowed.
- You cannot provide the password for the same slot you are attempting to remove.

Rotating Example (cont.)

```
# cryptsetup luksKillSlot /dev/sda4 0

Enter any remaining LUKS passphrase:

# cryptsetup luksDump /dev/sda4

LUKS header information for /dev/sda4
Version: 1
Cipher name: aes
Cipher mode: cbc-essiv:sha256
Hash spec: sha1
Payload offset: 4096
MK bits: 256
MK digest: 81 37 51 6c d5 c8 32 f1 7a 2d 47 7c 83 62 70 d9 f7 ce 5a 6e
MK salt: ae 4b e8 09 c8 7a 5d 89 b0 f0 da 85 7e ce 7b 7f
47 c7 ed 51 c1 71 bb b5 77 18 0d 9d e2 95 98 bf
MK iterations: 44500
UUID: 92ed3e8e-a9ac-4e59-afc3-39cc7c63e7f6
Key Slot 0: DISABLED
Key Slot 1: ENABLED
Iterations: 229712
Salt: 5d 71 b2 3a 58 d7 f8 6a 36 4f 32 d1 23 1a df df
cd 2b 68 ee 18 f7 90 cf 58 32 37 b9 02 e1 42 d6
Key material offset: 264
AF stripes: 4000
Key Slot 2: DISABLED
Key Slot 3: DISABLED
Key Slot 4: DISABLED
Key Slot 5: DISABLED
Key Slot 6: DISABLED
Key Slot 7: DISABLED
```

Disk-Level Encryption

ALTERNATIVES

InnoDB Tablespace Encryption

- MySQL 5.7 InnoDB
  - CREATE TABLE t1 (c1 INT) ENCRYPTION='Y';
  - ALTER TABLE foo ENCRYPTION = 'Y'
  - Doesn't cover encryption for bin-logs, slow-logs, and other places your data may be exposed.
  - As a side note, encryption on these other parts is currently work-in-progress.

MySQL Encryption Functions

```
CREATE TABLE `foo` (`secrets` varbinary(512));

INSERT INTO foo VALUES (AES_ENCRYPT('{"glossary": {"title": "example glossary",
"GlossDiv": {"title": "S","GlossList": {"GlossEntry": {"ID": "SGML","SortAs": "SGML",
"GlossTerm": "Standard Generalized Markup Language","Acronym": "SGML",
"Abbrev": "ISO 8879:1986","GlossDef": {"para": "A meta-markup language, used to create
 markup languages such as DocBook.","GlossSeeAlso": ["GML", "XML"]},
 "GlossSee": "markup"}}}}}', UNHEX(SHA2('My secret passphrase', 512))));

SELECT AES_DECRYPT(secrets, UNHEX(SHA2('My secret passphrase', 512))) FROM foo;

-- binlog_format=STATEMENT
mysqlbinlog -vv --base64-output=decode-rows mysqld-bin.000001 | grep passphrase

-- slow log
grep passphrase slow.log
```

(*) https://dev.mysql.com/doc/refman/5.7/en/encryption-functions.html

Final Notes

- After you rotate passwords, it's a good idea to repeat the header dump steps we performed above and store the new passwords in your password vault.
- Make sure all MySQL config values point to this partition.
- Check AES hardware acceleration
  - cat /proc/cpuinfo l grep -i aes
  - modprobe aesni-intel
- Use SSL with MySQL 5.7
  - Certificates created automatically
  - At minimum, distribute ca.pem to client applications

PERCONA
TRAINING

Questions? Thanks for Attending!