LCA 2007

Disk Encryption on Linux

http://marc.merlins.org/linux/talks/DiskEncryption/

Marc MERLIN
marc_soft@merlins.org

What to expect here

- The talk will look at the several solutions available for disk encryption
- why dmcrypt/cryptsetup is one of the better choices in most cases
- how and why to use LUKS and integrate this with pam-mount
- an example of how I use it with multiple key management

- Oh, I may^H^H^Hwill throw stuff at you, again :)

Why crypt?

- Because your 4 triple core CPUs are sitting idle, even when running a compiz 3D X setup with dancing windows without accelerated 3D :)
- Because if you can't remember the key, you probably didn't really need to get to that information anymore :)
- Because a bad byte on your disk will make sure a good portion of your filesystem goes with it, or if partition magic can't resize an encrypted partition, it must therefore be more secure :)

No, Really, Why Crypt?

- How many millions of user data records were lost on government and other laptops, leading to easy identity theft?
- If you have a laptop, think hard about how you would feel if some unknown person, or a potential competitor got a hold of all your files, mail, and more...
- If you have very important data on your work machine, but it is not physically secure at all times?
- Do you have really private data like credit card numbers that you do not feel safe having on a server in a colo?

How can I crypt?

- application/library built-in crypt (SSL private key, firefox personal security manager, ssh-agent)
- Third party crypto (gpg)
- Filesystems that support encryption (quite few, and it locks you out of your favourite existing filesystem)
- Overlay encryption
  - encfs: http://arg0.net/wiki/encfs/intro2
  - ecryptfs: http://ecryptfs.sourceforge.net/
- Block level encryption

Which Layer to Encrypt at?

- Filesystems with built in encryptions are ideal if you have one that does everything you need. In real life, they are rare
- EncFS is a nice idea, but FUSE and going through user space just is slower and potentially not as reliable (but better with bit flips or partition resizing). Good concept, but not an ideal implementation.
- Block level forces to encrypt the entire partition, but gives the best performance, choice of filesystems, and can encrypt swap

Which Block Level Encryption?

- old crypt over loopback (`losetup -e des /dev/loop0 /file`) considered obsolete and insecure (and slow)
- loop-AES is a fine choice, marginally more secure and faster than dmcrypt, but it's harder to deal with (will be nice when included in default util-linux, kernel, mkinitramfs, and have easier key management)
  - See http://deb.riseup.net/storage/encryption/loop-aes/ for problems
- dmcrypt is conveniently part of the disk mapper layer, and is supported by both initramfs on debian, and pam-mount.
- cryptsetup is required for dmcrypt, but you really want cryptsetup-luks for better key management

Crypto Talk

- IV: Initialization Vector
- ECB: Electronic Codebook
- CBC: Cipher Block Chaining
- CBC = IV + ECB and XOR
- More details at:
  http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation
  http://clemens.endorphin.org/LinuxHDEncSettings

Which Crypto Algorithm?

- Many ciphers can be used in cryptsetup: des, 3des, serpent, blowfish, twofish, aes, rot26, and 3rot26...
- blowfish was a good and fast option, but aes is now the standard, and linux has a fast i586 assembly version.
- You also get to pick your hashing function, and initialization vector...
- Ultimately, forget about everything else, and go with the current crypsetup default of “-c aes-cbc-essiv:sha256” (cbc being better than ecb, essiv being a good initialization vector, and sha256 a currently still good hashing algorithm)

About cryptsetup-LUKS

- cryptsetup now has LUKS built in, in recent Debians and Ubuntu. Hopefully your own distro is recent enough to have it too
- 2.6.10 or better required for proper cryptsetup-LUKS support
- LUKS offers 8 keyslots for passphrases that can unlock the main crypto setup (essential for storing a user key and at least one backup recovery key)
- keyslots are essential for changing passwords (plain cryptsetup requires re-encrypting the whole data)
- LUKS protects against dictionary attacks with PBKDF2 (many computing intensive iterations)

Before Formatting the Partition

- wipe first? This can take a whole day, or more...
- An optimized version would be: `wipe -fk -qQ 1 -R /dev/random /dev/part`
- But is it really necessary?
- Online HOWTOs say yes (note that most are copied from one another)
- I say totally overkill in my non professional opinion as a non licensed cryptographer :)
- Protecting against known plaintext attacks isn't that useful when you have known filesystem structures at known offsets. Relying on a plaintext attack resistant algorithm is a better route.

cryptsetup/dmsetup overview

```
root@saruman:~# cryptsetup luksFormat /dev/sda5
This will overwrite data on /dev/sda5 irrevocably.
Are you sure? (Type uppercase yes): YES
Enter LUKS passphrase:
Verify passphrase:
Command successful.

root@saruman:~# cryptsetup luksOpen /dev/sda5 _dev_sda5
Enter LUKS passphrase:
key slot 0 unlocked.
Command successful.
root@saruman:~# dmsetup ls
_dev_sda5       (253, 0)
root@saruman:~# dmsetup status _dev_sda5
0 5300355 crypt

root@saruman:~# cryptsetup status _dev_sda5
/dev/mapper/_dev_sda5 is active:
  cipher:  aes-cbc-essiv:sha256
  keysize: 128 bits
  device:  /dev/.static/dev/sda5
  offset:  1032 sectors
  size:    5300355 sectors
  mode:    read/write
root@saruman:~# mount /dev/mapper/_dev_sda5 /mnt/mnt
```

LUKS Key Management

```
root@saruman:~# cryptsetup luksDump /dev/mapper/_dev_sda5
/dev/mapper/_dev_sda5 is not a LUKS partition
root@saruman:~# cryptsetup luksDump /dev/sda5
LUKS header information for /dev/sda5
Key Slot 0: ENABLED
        Iterations:             150151
        Salt:                   7b 11 d4 f9 71 34 0f 59 7e 42 b2 8c f7 bc 65 53
                                c5 64 ba 6b 2f ad 19 bf d8 ba 82 36 87 3d 3e 7d
        Key material offset:    8
        AF stripes:             4000
Key Slot 1: DISABLED
Key Slot 2: DISABLED
(...)
root@saruman:~# cryptsetup luksAddKey /dev/sda5 --key-slot=4
Enter any LUKS passphrase:
Verify passphrase:
key slot 0 unlocked.
Enter new passphrase for key slot:
Verify passphrase:
Command successful.
root@saruman:~# echo -e "test\ntest3" | cryptsetup luksAddKey /dev/sda5 --key-
slot=5
key slot 0 unlocked.
Command successful.
root@saruman:~# echo test4 > /tmp/key;  echo test | cryptsetup luksAddKey
/dev/sda5 –key-slot=6 /tmp/key
```

LUKS Key Management (2)

```
root@saruman:~# cryptsetup luksDelKey /dev/sda5 0
Command successful.
root@saruman:~# cryptsetup luksDump /dev/sda5
LUKS header information for /dev/sda5
Key Slot 0: DISABLED
Key Slot 1: DISABLED
Key Slot 2: DISABLED
Key Slot 3: DISABLED
Key Slot 4: ENABLED
Key Slot 5: ENABLED
Key Slot 6: ENABLED
Key Slot 7: DISABLED

root@saruman:~# umount /mnt/mnt
root@saruman:~# cryptsetup luksClose _dev_sda5

root@saruman:~# echo test | cryptsetup luksOpen /dev/sda5 _dev_sda5
Command failed: No key available with this passphrase.
root@saruman:~# echo test2 | cryptsetup luksOpen /dev/sda5 _dev_sda5
key slot 4 unlocked.
Command successful.
```

Automated Key Management

- Adding/removing/changing keys is unsafe unless you know which slot has which key
- My patch adds a –key-slot option for luksFormat and luksAddKey, and I keep track of slot locations
- Clemens Fruhwirth <clemens@endorphin.org> added the patch to standard cryptsetup
- Change password: add a new key and remove the old one

```
polgara:~$ sudo cryptsetup luksAddKey --key-slot 1 -q -y /dev/sda6
Enter any LUKS passphrase:
key slot 0 unlocked.
Enter new passphrase for key slot:
Verify passphrase:
Command successful.
polgara:~$ sudo cryptsetup luksDelKey /dev/sda6 0
Command successful.
```

Key Recovery / Genuserkey

- You can't recover a lost/forgotten/deleted key, but you can use a backup key to unlock the block device
- You can generate a per user, non user changeable, recovery key. Here's genuserkey:

```
HASHSALT='averydeepsecret' echo "$USER""$HASHSALT" | openssl sha1
```

- The key then lets you replace a forgotten user key

```
polgara:~$ genuserkey
b9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0e
# -q: batch mode / -y:
polgara:~$  sudo cryptsetup luksAddKey --key-slot 0 -q -y /dev/sda6
Enter any LUKS passphrase: <paste password from above>
key slot 2 unlocked.
Enter new passphrase for key slot:
Verify passphrase:
Command successful.
polgara:~$ sudo cryptsetup luksDelKey /dev/sda6 1
Command successful.
```

LuksDump

```
root@polgara:~# cryptsetup luksDump /dev/sda6
LUKS header information for /dev/sda6
Version:        1
Cipher name:    aes
Cipher mode:    cbc-essiv:sha256
Hash spec:      sha1
Payload offset: 1032
MK bits:        128
MK digest:      44 b8 5d 49 00 dc fd 28 6b 52 ec 2b 1e 9a 3a 81 58 cc 34 a5
MK salt:        39 c4 86 eb c4 56 1d 7f d7 51 cf 83 87 c0 46 38
                91 c8 53 85 da fc cf b1 0c bd ef b2 8f 88 8b 13
MK iterations:  10
UUID:           84ff83c7-ecc1-4fce-9825-5e34dd369711

Key Slot 0: ENABLED
        Iterations:             189211
        Salt:                   46 62 96 81 f6 48 92 7a 54 90 de 99 3e fe b9 ab
                                d6 eb 77 34 f3 bd 1e 49 9a e3 2a ec 83 3e 64 d6
        Key material offset:    8
        AF stripes:             4000
Key Slot 1: DISABLED
Key Slot 2: ENABLED
        Iterations:             185640
        Salt:                   37 f3 33 59 f5 fb 09 bb 4e 53 f7 f6 cc 45 c3 74
                                57 8a 6f 34 bf b0 52 16 1f 70 60 6a 2b 8b 4b a3
        Key material offset:    264
        AF stripes:             4000
Key Slot 3: DISABLED
(...)
```

System Crypt

- Two options: crypt all partitions, or all partitions but root
- not crypting root lets the system boot, and initscripts can ask for the decrypt password at boot time
- To avoid entering the password more than once, you can save it in a file and delete it after `/etc/init.d/cryptdisks`

```
# /etc/crypttab
# <target name>      <source device>      <key file>    <options>
part5                /dev/hda5           /passwd       luks
part6                /dev/hda6           /passwd       luks
```

- dmsetup creates /dev/mapper partition that you mount in /etc/fstab

```
# /etc/fstab
/dev/hda1              /       ext3    defaults          0       1
/dev/mapper/part5      /usr    ext3    defaults          0       2
/dev/mapper/part6      /var    ext3    defaults          0       2
```

Full System with / Crypt

- crypting root is harder, it uses a non crypted /boot, and an initrd image that can boot and decrypt the root partition before it is mounted (pivot_root) and the system booted
- To make a root decrypting initrd image on debian:
  `update-initramfs -v -c -k 2.6.19.1`
- the initrd can ask for the password and save it in /passwd on the new root, can be read from /proc/cmdline, a USB key, or somesuch

```
# /etc/crypttab
# <target name>      <source device>      <key file>    <options>
root                 /dev/hda1           none          luks
part5                /dev/hda5           /passwd       luks
part6                /dev/hda6           /passwd       luks

# /etc/fstab
/dev/mapper/root     /       ext3    defaults        0       2
/dev/hda2            /boot   ext3    defaults        0       1
/dev/mapper/part5    /usr    ext3    defaults        0       2
/dev/mapper/part6    /var    ext3    defaults        0       2
```

Special Considerations For Laptops

```
# swsusp for laptops (written for my Thinkpad T42p, but ought to work for
# most other configurations). By Marc MERLIN <marc_soft@merlins.org> v1.0

# we can't test the sleep file before the previous invocation is gone
shlock -f /var/run/sleep.lock -p $$ # shlock comes from INN

DIFF=$(( `date "+%s"` - `stat -c "%Z" /var/run/sleep 2>/dev/null` ))
if [ $DIFF -lt 30 ]; then
   echo "detected possible signal bounce, skipping" >&2; exit 0
else
   echo "going to sleep, last time was $(( `date "+%s"` - `stat -c "%Z"
/var/run/sleep 2>/dev/null` )) secs ago" >&2
fi
touch /var/run/sleep

/etc/init.d/irda-utils stop; /etc/init.d/hwclock.sh stop
killall xscreensaver /usr/bin/xscreensaver-command 2>/dev/null

echo -n mem > /sys/power/state # or hibernate: echo -n disk > /sys/power/state
/etc/init.d/hwclock.sh start; touch /var/run/sleep; /etc/init.d/irda-utils start

# lock if laptop slept more than 20mn
if [ $DIFF -gt 1200 ]; then
    echo "Slept for $DIFF, locking" >&2; su - merlin -c /usr/local/bin/xlock
   # xlock is: pidof xscreensaver || (xscreensaver &) ; sleep 1; xscreensaver-
command -lock
else
    echo "Slept for $DIFF, not locking" >&2
fi
/bin/rm /var/run/sleep.lock
```

Data Partition Crypt

- /etc/init.d/cryptdisks and root crypting might be fine for a laptop, but not a server/partially unattended workstation: it would hang when you reboot, or not reboot with all its partitions mounted
- Your OS data may likely not be that vital or secret
- Crypting /data allows the system to boot unattended
- You can also use the user's login password to unlock the data partition, and avoid multiple passwords
- This is where pam-mount comes in: use the password from login/xdm/gdm/kdm to decrypt and mount the data partition

Pam-mount

- pam-mount is the solution to making a crypted data partition as transparent as possible
- pam-mount re-uses your password to decrypt and mount a partition listed in /etc/security/pam_mount.conf:

```
volume $USERLOGIN crypt - /dev/part /dir/data - - -
```

Pam-mount Setup

- Only insert pam-mount auth in services that require passwords each time (#su - nobody would fail and can even cause cron to segfault)
- pam-mount session uses the saved auth password to decrypt and mount partitions in /etc/security/pam_mount.conf

```
# /etc/pam.d/common-auth.pammount
#
auth    sufficient    pam_unix.so
auth    optional      pam_mount.so use_first_pass
auth    required      pam_deny.so

#
# /etc/pam.d/common-session.pammount
#
session       required       pam_unix.so
session       required       pam_limits.so
session       optional       pam_mount.so

# source pam-mount common files for _passworded_ login services
sudo bash -c 'for i in kdm gdm login ssh ; do \
perl -p -i -e "s#common-(auth|session)#common-\$1.pammount#" /etc/pam.d/$i; done'
```

Caveats

- Encryption speed and CPU usage (aes-586)
- disk corruption causes much harsher damage
- even a simple bit flip can cause widespread damage
- encrypting your root partition is tricky
- It would be nice if LUKS cryptsetup allowed for more than 8 keys to allow for an almost arbitrary amount of user passwords to unlock the data partition on a given machine

References

- This talk: http://marc.merlins.org/linux/talks/DiskEncryption/talk/
- dmcrypt/cryptsetup pages:
  http://www.saout.de/tikiwiki/tiki-index.php
  http://deb.riseup.net/storage/encryption/dmcrypt/#dmcrypt_home_partitions
  http://luks.endorphin.org/
- Loop-AES and Loop-AES vs dmcrypt:
  http://loop-aes.sourceforge.net/loop-AES.README
  http://deb.riseup.net/storage/encryption/loop-aes/
  https://wiki.boum.org/TechStdOut/LinuxCryptoFS
  https://docs.indymedia.org/view/Local/UkCrypto#Filesystem
- Different crypto and hash algorithms
  http://clemens.endorphin.org/LinuxHDEncSettings
  http://en.wikipedia.org/wiki/User:Davidgothberg/Crypto_steps
  http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

What's missing / TODO

- Cryptsetup-LUKS could support more than 8 keys
- Switch to full speed layered encryption with good key management
- ecryptfs (2.6.19) is an exciting new option

Questions

Are you a good sysadmin,
or programmer?

Need a job?

(Silicon Valley, Sydney, Santa Monica, Zürich,
New York, Dublin, Arizona, Tokyo, India, and others)

(several of my coworkers came from LCA)

Email: marc@merlins.org