Encrypting Your Files

Because nobody else will
And would you trust them if they did?

Why?

- Sensitive personal information
- NSA
- Identity thieves

Linux Disk Encryption

- Dm-crypt is default under Linux
  - Full file systems
  - Partial file systems with loop-back device
- LUKS
  - Easier to use
  - More options
  - Supports TrueCrypt volumes

Linux Unified Key Setup

- Extends the dmcrypt program
- Encrypts the partition BEFORE the file system is created
- Works with LVM and RAID partitions
- Almost the whole disk (still need /boot)
  - Or just specific partitions (sda2, sda5, etc.)
- Multiple passphrases for each container
- Built into many installers

Issues

- Everyone knows you have an encrypted partition
- Have to provide a passphrase on boot
  - This can be inconvenient at times
- No in-place encryption
  - Pre-existing data will NOT be removed/encrypted
- Effectiveness on SSD not yet clear
  - Especially if you don't encrypt on initial install

Important Points

- Difference between passphrase and master key
- Difference between container, partition, and file system
- Encryption won't protect you from a disk crash
- Disk encryption only really protects your data when the system is off
- Entropy on system during initial setup
  - Master key and passphrase hash iterations
- LUKS header (first 2 MB on disk)

Setting up encrypted file system

- Create the partition (fdisk, parted)
- Wipe the disk
  - shred -v --iterations=1 /dev/$DEVICE
- Create the container
  - cryptsetup --verbose --verify-passphrase luksFormat /dev/$DEVICE
- Open the container
  - cryptsetup open /dev/$DEVICE $NAME --type luks
    - Provide passphrase when prompted

  - Other option: cryptsetup luksOpen
- Verify, just to be sure
  - ls -l /dev/mapper/$NAME → ../dm-#
- Create the filesystem
  - mkfs -t {ext3/4/btrfs} /dev/mapper/$NAME
- Edit /etc/crypttab
  - $NAME /dev/$DEVICE {-,none,/password/path} options
  - Last 2 columns are optional
  - Device can be specified by UUID, disk-by-id
  - Options include: luks, noauto, nofail, swap, size=, timeout=

- Edit /etc/fstab
  - /dev/mapper/$NAME /mount/point FSTYPE defaults,**nofail**,**noauto**,users,{others} 0 2
  - fsck can't run until the container has been opened
- Mount the filesystem
  - mount /mount/point

cryptsetup commands

- luksFormat /dev/$DEVICE
- open /dev/$DEVICE $NAME {--type luks}
- close $NAME
- status $NAME
- luksSuspend $NAME
- luksAddKey /dev/$DEVICE
- luksRemoveKey /dev/$DEVICE

- luksDump /dev/$DEVICE
  - --dump-master-key
- luksHeaderBackup /dev/$DEVICE --header-backup-file /path/to/file
- luksHeaderRestore /dev/$DEVICE --header-backup-file /path/to/file
- benchmark

- DEMO

What about backups?

- Encrypted file systems don't protect you from failed disks
- If your data is important enough to encrypt on a live system, you should protect the backups
- For best protection, your backups should be at a second site

duplicity

- Built on python
- Encrypts using gnupg
- rsync like backups (block level changes)
- Also compresses files during backup
- Maintains 2 copies of the database
- Uses relative path within backups/restores
- Uses multiple back ends to transfer files to remote location

- Backends:
  - Amazon web service
  - Google cloud storage
  - Google docs
  - Rackspace
  - Dropbox
  - rsync
  - ftp(s)
  - ssh/sftp/scp (2 different backends)
  - Ubuntu One

- Supports both symmetric and public key encryption (or no encryption, if desired)
- Local database files are NOT encrypted
- Full vs. incremental backups
- Multiple named backups
- Deletion of old backups
- Restore to point in time, not just most recent
- Can sign encrypted backups with 2nd key
- Can include/exclude files from backup
- Creates 25MB volumes by default

Lessons Learned

- Full backups support BOTH symmetric passphrase and public keys, but incremental backups do not
- Put /tmp on a tmpfs, rather than disk, to increase speed
- Requires more disk space for restores than for backups

Symmetric Encryption

PASSPHRASE=XXXXXXXXXX

export PASSPHRASE

duplicity --archive-dir /archive --name remotenet --ssh-backend pexpect /home/$USER sftp://user@remote/relative/path

Public Key Encryption

duplicity –archive-dir /archive -name remotenet –encrypt-key 08114727 --ssh-backend pexpect --ssh-option="-l900" /home/$USER sftp://user@remote/relative/path

- This command demonstrations the use of SSH options, in this case, limiting network band width, during the sftp backup

- DEMO

Questions??

This presentation has been brought to you by the revelations of Edward Snowden. If you would like to help, go to https://petitions.whitehouse.gov/petitions, search for Edward Snowden, and sign the petition asking for a pardon.