Lesson #5 - Disk encryption and data backup

Advanced Linux Administration

Karel Šrot

Motivation

- Keep disk data safe and enable its recovery in case of unexpected or undesired event
    - e.g. data loss or data corruption due to an attacker or accidentally

CAPEC* Domains of Attack

- Software - SW exploitation
- Hardware - HW exploitation
- Communications - exploitation of communications and related protocols
- Supply Chain - disruption by manipulating hw, sw, services
- Social Engineering - manipulation and exploitation of people
- Physical Security - exploitation of weaknesses in physical security

*) Common Attack Pattern Enumeration and Classification by MITRE

Security objectives*

- Resource protection
- Authentication
- Authorization
- Integrity - data/system
- Nonrepudiation - proof that transaction occurred
- Confidentiality
- Auditing security activities

*) according to IBM

Examples of possible attacks

- Device stolen
  - Strong encryption of important data may be enough to protect the data against the brute force attack (if the device was powered off)
- Cold-boot attack / firewire dump attack
  - Attempt to get the encryption/decryption key from the memory
  - Very difficult or impossible to protect
  - https://blog.f-secure.com/cold-boot-attacks/
- Evil maid attack (unattended device)
  - An attacker modifies the device (e.g. /boot) so it reads your password on the next boot
  - Secure boot with signed kernel and initramfs image prevents an attack on boot process

Disk encryption

Data objects

- Physical disk (HDD/SSD)
- Raid
- Portable storage
- ...

Disk encryption - cont.

What to consider / How much security do you need?

- Physical access!!!
- Risk and damage caused by data leak/loss
- Amount of data requiring protection
- Who and how can access the data
- Performance impact
- ...

Disk encryption - cont.

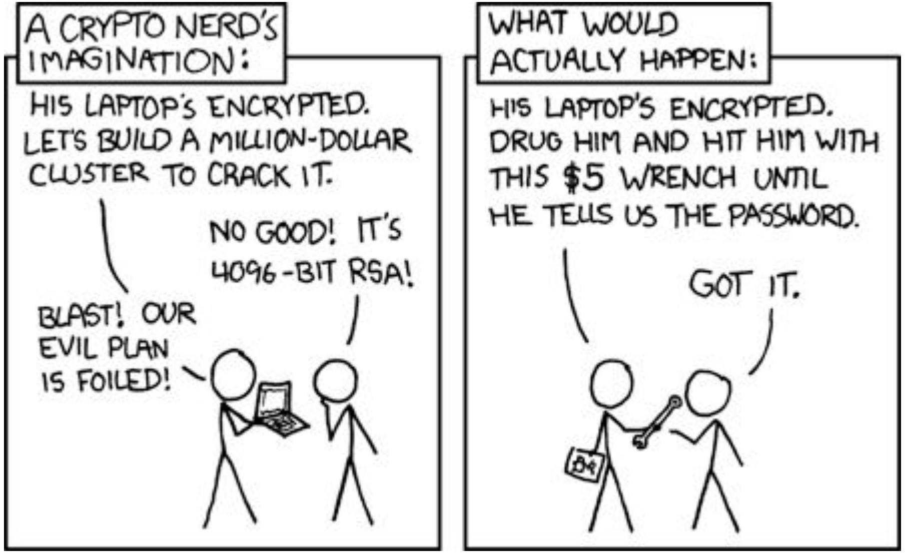
- Depending on the method disk encryption can protect the data in case an intruder gets physical access to the device (theft, insider,...)
- Typically protection of the turned-off device
  - won’t help when an attacker gets access to a (running) system having data decrypted!!!
- We can encrypt whole drive or just a subset (partition, files, directories)
- Remember that data may end up on a different place
  - E.g. in a swap partition after the system hibernation, snapshots, backups, bad sectors relocation on hard drive
- Usually a user is required to provide a password during a system boot
  - But the key can be also obtained from a file
  - Network-Bound Disk Encryption (NBDE) allows the user to encrypt a volume without requiring to manually enter a password when systems are restarted.

Disk encryption - cont.

- Remember! By law/force you may be pushed to reveal your password.

Resources:

- [Disk encryption on Archlinux wiki]


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

https://xkcd.com/538/

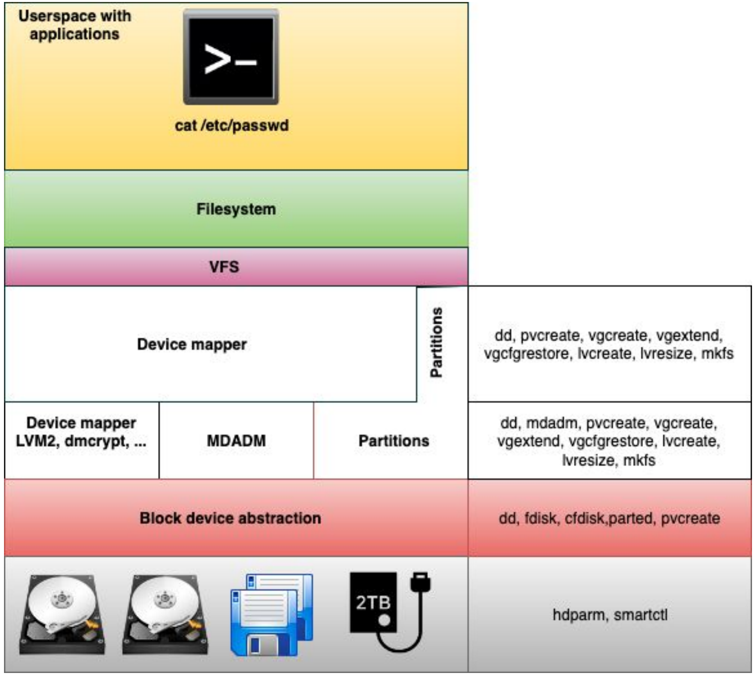
Disk encryption methods

- Filesystem native encryption
  - Ext4, ZFS
  - Similar to the Stacked filesystem encryption below
- Stacked filesystem encryption
  - eCryptfs, EncFS,...
  - implemented as a layer that stacks on top of an existing filesystem (defining specific directories for which the content would be encrypted)
  - Cloud storage optimized tools available - optimized for file transmission over Internet (gocryptfs, cryfs, custom implementations from cloud storage provides,...)
- Block device encryption
  - Loop-AES, dm-crypt / cryptsetup / LUKS, TrueCrypt/VeraCrypt
  - Operate below the filesystem layer and make sure that everything written to a certain block device (disk, partition, loop device,..) is encrypted

Aspects	Block device encryption	Stacked filesystem encryption
Encrypts	Disk, partition, file acting as virtual partition (loop device)	A directory in an existing filesystem
Operates	Block device, does not care about its content (filesystem, partition table, LVM, …)	Additional layer on top of existing filesystem
File metadata encryption	yes	No (names can be encrypted)
Can be used to encrypt whole drive	yes	no
Can be used to encrypt swap	yes	no
Preserve benefit from filesystem level compression	yes	no
Can be used without fixed amount of pre-allocated disk space	no	yes
Can be used to protect existing filesystem without block device (e.g. NFS, samba, cloud storage,..)	no	yes
Allows off-line file-based backups	no	yes

Disk encryption using cryptsetup / LUKS

- LUKS (Linux Unified Key Setup) is a specification for block device encryption
- Establishes an on-disk format for the data, as well as a passphrase/key management policy
  - Provides multiple (8) key slots
- LUKS uses the kernel device mapper subsystem via the dm-crypt module to handle data encryption and decryption
- User-level operations are accomplished through the use of the cryptsetup utility



Userspace with applications

cat /etc/passwd

Filesystem

VFS

Device mapper

Partitions

dd, pvcreate, vgcreate, vgextend, vgcfgrestore, lvcreate, lvresize, mkfs

Device mapper LVM2, dmcrypt, ...

MDADM

Partitions

dd, mdadm, pvcreate, vgcreate, vgextend, vgcfgrestore, lvcreate, lvresize, mkfs

Block device abstraction

dd, fdisk, cfdisk,parted, pvcreate

hdparm, smartctl

DEMO - disk encryption using cryptsetup / LUKS

Resources:

- Fedora Disk encryption user guide
- Archlinux device encryption wiki page

Disk encryption using EncFS

- Provides filesystem level encryption layer
  - Mounting of an EncFS directory refers to mounting an encrypted directory to another unencrypted mount point (directory)
    - Beware of a side-channel leak!!!
- Easy to configure for basic use
  - metadata information stored per-directory configuration file .encfs6.xml
- Works well for network storage mount points (e.g. NFS, SSHFS)
  - For increased security do not store .encfs6.xml along with the encrypted data

Resources:

- EncFS HowTo for Fedora
- EncFS on Archlinux wiki
- EncFS on Wikipedia

Disk encryption using eCryptfs

- Provides filesystem level encryption layer
  - Mounting of an eCryptfs directory refers to mounting an encrypted directory to another unencrypted mount point (directory)
    - Beware of a side-channel leak!!!
- In general a bit harder to configure - OTOH, there are setup scripts available to configure and decrypt ~/Private mount point upon user login
  - However, unmounting upon logout doesn't currently work
- Not recommended for network storage mounts due to a number of known bugs

Resources:

- eCryptfs HowTo for Fedora
- eCryptfs on Archlinux wiki

Data backup

Goals / Objectives

- Ability to restore the data in case they were lost, damaged, accidentally or intentionally modified

Data backup - things to consider

- Why do we need the backup (use case)
- Data to backup - whole disk or partition vs individual files, whole system vs specific data
- Backup type - full, incremental, differential,...
- Backup history (frequency, number of (old) backups preserved)
- Backup storage/media - HDD, tape, cloud storage,...
- Backup storage location(s) - onsite + remote location
- Backup verification
- Backup protection - Data integrity, Encryption, physical security,...
- Recovery needs - what needs to be recovered, recovery time,..
- Backup methods - copy/RAID/snapshots,... usually multiple methods

Backup software - examples

- Backup multiple systems running on multiple platforms:
  - Amanda
  - Bacula
  - backuppc
- Backup multiple Linux systems
  - rdiff-backup
- Single system backup
  - Kbackup
  - BackInTime
  - Timeshift
- Disk cloning & recovery
  - Clonezilla
  - mondorescue

rdiff-backup

- Python script for creating directory backups, based on rsync
- Easy to use
- Combine the best features of a mirror and an incremental backup
  - You have the latest directory backup directly available but you can also restore it to any (backed up) version in history
  - Incremental backups are stored in the `rdiff-backup-data` subdirectory of the mirror (backup) directory
  - You should not write to the mirror directory except with rdiff-backup
- Not a daemon, automated backups can be achieved by using `crond`
- `rdiff-backup` must be installed on a remote system in order to initiate the backup remotely (ideally with passwordless ssh login)

Resources:

- rdiff-backup official documentation

rdiff-backup - DEMO

```
rdiff-backup [options] user1@host1::/source-dir user2@host2::/dest-dir
```

Limiting the set of files:

```
--include, --include-filelist,
--include-globbing-filelist, ...
--exclude, --exclude-filelist,
--exclude-globbing-filelist, ...
--max-file-size, --min-file-size
```

Console output:

```
--print-statistics
--verbosity [0-9]
```

Checking changes:

```
--compare, --compare-at-time TIME
--compare-full,
--compare-full-at-time TIME
```

Backup verification:

```
--verify, --verify-at-time TIME
```

Restore

```
--restore-as-of TIME, --force
```

Managing incremental backups:

```
--list-increments
--list-at-time TIME
--list-changed-since TIME
--remove-older-than TIME
```

Time formats:

- now or number followed by `[smhDWMY]`, e.g. `1h78m`
- Datetime format `2018-01-25T07:00:00+02:00`

Workshop

60 min

Lab 1 - Partition encryption with Cryptsetup + LUKS [20 min]

1. Using `dd` create a disk file `/var/tmp/disk0` with random content (use `/dev/urandom`)
2. Format `/var/tmp/disk0` as dm-crypt/LUKS encrypted device, manually providing the password
3. Access the decrypted disk and format it with EXT4 filesystem
4. Create a mount point `/mnt/disk0` and mount it there
5. Create a text file `/mnt/disk0/secret` with some content
6. Configure `/var/tmp/disk0` to be decrypted during a boot (providing a password manually) and mounted under `/mnt/disk0`
7. Reboot a system (providing the password during the boot) and verify the disk was properly mounted and `/mnt/disk0/secret` file is available

Lab 1 - Partition encryption with Cryptsetup + LUKS - cont.

8. Generate a random keyfile `/root/disk0key` and configure it as an additional key for `/var/tmp/disk0`
9. Configure system to use that keyfile during the boot
10. Reboot and verify the disk had been decrypted and mounted without having to manually provide the password

Resources:

- `man cryptsetup; man crypttab`
- Fedora disk encryption user guide

Lab 2 - Simple backup scenario using rdiff-backup [15 min]

1. Create a `/test` directory containing text files `file0`, `file1`, `file2`.
2. Create a backup of `/test` to `/backup` directory, excluding `file0`.
3. In `/test` modify content of `file1`, remove `file2` and create `file3`.
4. Do the backup again and list available incremental backups.
5. Modify file1 once more and do one more backup.
6. List available backups for `file1` and `file3`.
7. Restore the modified `file1` to its original state using the oldest backup
8. Remove the oldest incremental backup. Confirm by listing available backups.
9. Verify the backup. Modify content of `file3` directly in `/backup` directory and verify the backup again.

Resources:

- `man rdiff-backup`, rdiff-backup HowTo

Lab 3 - Setting up encrypted ~/Private with EcryptFs [10 min]

Following the HowTo set up encrypted `~/Private` directory that gets decrypted when user logs in in GUI. For console access one has to run `ecryptfs-mount-private` manually

1. Configure EcryptFS encryption for `~/Private` directory
2. Log out (or reboot) and log in again and confirm with the `mount | grep Private` command that the `~/Private` directory was mounted automatically
3. As a user create some content in the `~/Private` directory and also observe the content of `~/.ecryptfs` directory.
4. As `root` try to access the content of `~/Private`.
5. As user, manually unmount the `~/Private` directory and verify the decrypted content is not available.

Resources:

- eCryptfs on Fedora HowTo