Improving disk encryption in Linux

Research Day, Brno, January 2020

Milan Brož

Red Hat

Who am I?

Upstream maintainer of Linux disk encryption

Ph.D. graduate and Red Hat open-source lab head, FI MUNI

Brno Red Hat Research Program Manager

https://www.linkedin.com/in/mbroz/

https://twitter.com/gmazyland

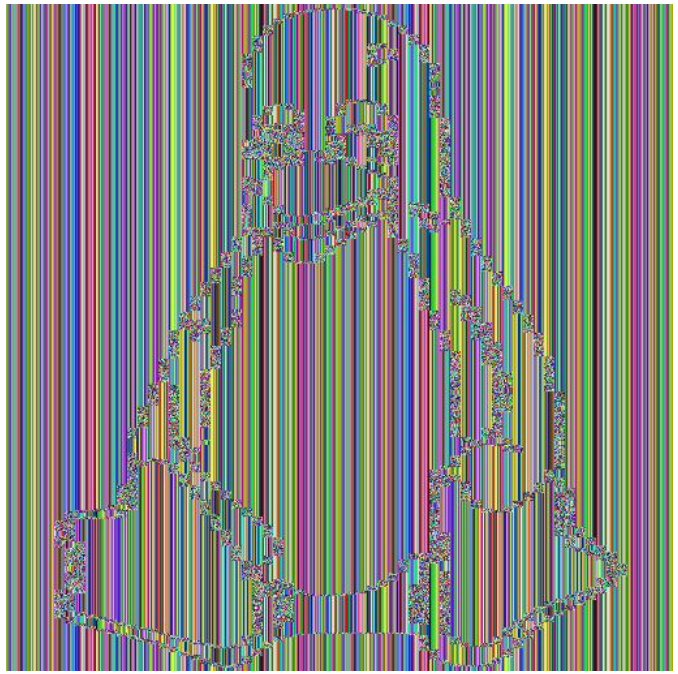
https://scholar.google.com/citations?user=5ugu5pgAAAAJ

Examples in this talk are a demonstration that applied research and collaboration with the academic world can produce both academic publications and open-source enterprise product innovations.

Even just with interns, students, and no budget.

Disk encryption – a very common security feature

What it provides and what can be improved?



Transparent encryption of data

- On the disk-sector level
- Implementation in software or hardware
- Provides data confidentiality
- Length-preserving encryption
  => *cannot provide data integrity protection*

Common threat model

- Primarily for data-at-rest protection
- Stolen device, disk in repair, ...

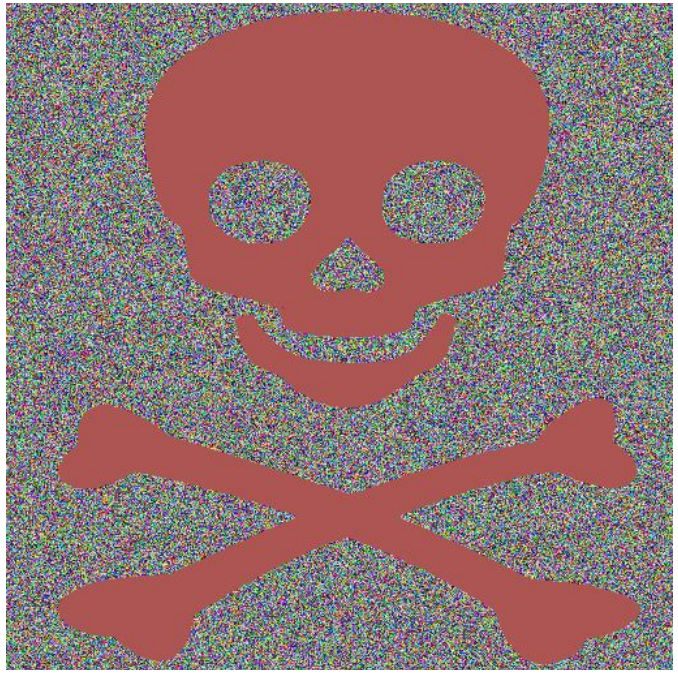
Our threat model: device returns to user

- Silent data corruption
- Implanted data without user knowledge (attacker)

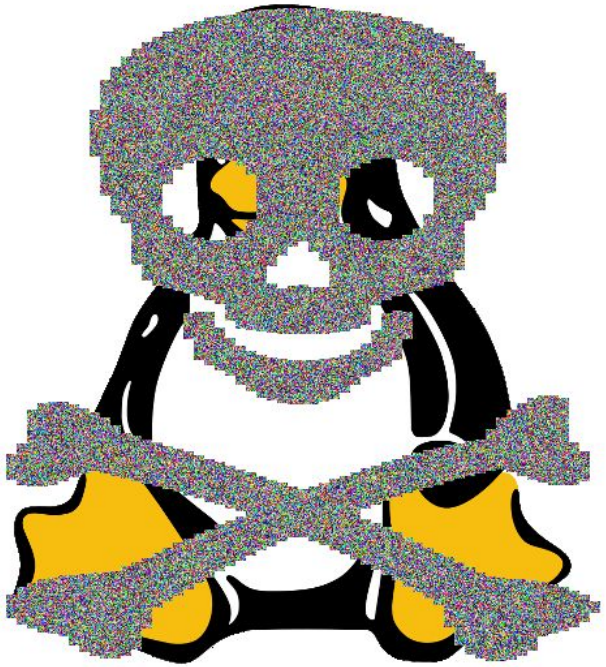
The illustrative image above is a visualization of a real encrypted disk with wrongly used encryption mode (AES-XTS with constant IV).

Disk encryption - data integrity protection?

What it means that data are encrypted but not integrity protected?



Encrypted disk with intentionally corrupted data



Decrypted disk and the effect of data corruption

The illustrative image above is a visualization of a real encrypted disk (dm-crypt with AES-XTS mode used today in most systems). Note pattern propagation as a pseudo-random noise in decrypted data; also note propagation to encryption blocks (one-bit change propagates to the whole encrypted block in XTS mode or to even the entire sector). If the upper layer (like filesystem) does not detect this data corruption, it propagates even further.

Data integrity protection on the disk sector level

side goal: to show that data integrity protection is important

Our data integrity protection requirements

- Reliable but without special HW (commercial of-the-shelf drives)
- Configurable (algorithm agnostic) and aligned to the native sector size
- Use state-of-the-art algorithms (authenticated encryption, AEAD)
- Simple & user friendly configuration - LUKS2 format
- Open-source and free licence, no patents

Layer separation & reuse

- Use existing block-layer integrity extension but emulated in software
- Reliable handling of authentication tags (dm-integrity)
- Cryptographic part (extension of dm-crypt)

Data integrity protection - implementation

dm-integrity - a new device-mapper target

- Emulates per-sector metadata; interleaved data and metadata sectors
- Optionally stand-alone mode (quite unexpected success :-)
- Protection to power-fail (journalling)
- Side-effect - used as QA tool (uncovered several issues in MD RAID)

dm-crypt extension for authenticated encryption

- Stacked above dm-integrity device (cryptographic part is in dm-crypt)
- Linux kernel crypto API
- Not many usable authenticated algorithms suitable here...
  - CAESAR competition finalist: AEGIS
- Some limitations but it was quite well perceived
- Integration to LUKS2
- Motivation: the future is to use authenticated encryption on higher layers

Experimental LUKS2 authenticated encryption

```
# cryptsetup luksFormat /dev/sdb --cipher aegis128-random --key-size 128 --integrity aead
  Enter passphrase for /dev/sdb: ***

# cryptsetup luksOpen /dev/sdb test
  Enter passphrase for /dev/sdb: ***

# cryptsetup status test
/dev/mapper/test is active.
  type: LUKS2
  cipher:  aegis128-random
  keysize: 128 bits
  key location: keyring
  integrity: aead
....
```

Data integrity protection – standalone dm-integrity

```
# integritysetup format /dev/sdb [ -I crc32c ]
Formatted with tag size 4, internal integrity crc32c.
Wiping device to initialize integrity checksum.

# integritysetup open /dev/sdb test [ -I crc32c ]

# mkfs -t xfs /dev/mapper/test
....

# integritysetup status test
  type: INTEGRITY
  tag size: 4
  integrity: crc32c
  device: /dev/sdb
....
```

New memory-hard key derivation function for LUKS2

- LUKS2 is the new version of Linux native disk encryption format
- State-of-the-art key derivation using memory-hard function
- Argon2 - winner of Password Hashing Competition
  - all candidates were measured for this practical use case
- Argon2 provides much better resistance to dictionary (offline) attacks
  - on parallel platforms
  - on GPUs and ASICs
- Ongoing effort to implement it in OpenSSL library
- Cost analysis for improved security margin in LUKS2

Another example – Linux cryptsetup native BitLocker support

- BitLocker is a proprietary Microsoft Windows Disk Encryption technology
- Part of specification is public, something is already reverse engineered
- Our Read/Write support use dm-crypt in kernel and cryptsetup in userspace
- Allows sharing native windows encrypted disks with Linux
- Unlocking with passphrase or recovery passphrase (no TPM support)
- Support for AES-XTS mode but also for CBC (also with Elephant diffuser)
- Cryptsetup release candidate just now (in Fedora rawhide)
- For more info see talk at DevConf 2020

Opening BitLocker compatible device

```
# cryptsetup bitlkOpen bitlocker_xts_ntfs.img test
 Enter passphrase for bitlocker_xts_ntfs.img: ***

# cryptsetup status test
 /dev/mapper/test is active.
 type: BITLK
 cipher:  aes-xts-plain64
 keysize: 128 bits
...

# blkid /dev/mapper/test
 /dev/mapper/test: UUID="..." TYPE="ntfs"

# mount /dev/mapper/test /mnt/tst

...
```

Opening BitLocker compatible device

```
# cryptsetup bitlkDump bitlocker_xts_ntfs.img
 Info for BITLK device bitlocker_xts_ntfs.img.
 Version:       2
 GUID:          …
 Created:       Wed Oct 23 17:38:15 2019
 Description:   DESKTOP-xxxxxxx E: 23.10.2019
 Cipher name: aes
 Cipher mode: xts-plain64
 Cipher key:    128 bits

 Keyslots:
 0: VMK
          GUID:            …
          Protection:      VMK protected with passphrase
          Key data size:   44 [bytes]

1: VMK
          GUID:            …
          Protection:      VMK protected with recovery passphrase
...
```

Cooperation & people (mentioned examples only)

- Milan Brož - Red Hat, Masaryk University
  - part of Ph.D. thesis, kernel and LUKS2 code, generic grumbling
- Mikuláš Patočka, Red Hat
  - reliable kernel code for dm-integrity
- Ondrej Mosnáček, Masaryk University student (now Red Hat)
  - (master thesis) CAESAR candidates authenticated encryption algorithms in kernel
  - (bachelor thesis) analysis of PBKDF2 and GPU accelerated attacks
- Vojtěch Polášek, Masaryk University student (now Red Hat)
  - (master thesis) analysis of Argon2 KDF security margin
- Ondřej Kozina, Red Hat
  - LUKS2 code, userspace code co-maintainer
- Vojtěch Trefný, Red Hat, Tomas Bata university in Zlín student
  - (master thesis) BitLocker compatible code and implementation
- Čestmír Kalina, Red Hat, Masaryk university student
  - (bachelor thesis) Argon2 implementation for OpenSSL
- 5 academic (peer reviewed) publications
- 6 developer conference talks (DevConf, FOSDEM, EurOpen)

Thank you

research.redhat.com

Red Hat