# Security features for UBIFS

Richard Weinberger
sigma star gmbh

# /me

- Richard Weinberger
- Co-founder of sigma star gmbh
- Linux kernel developer and maintainer
- Strong focus on Linux kernel, lowlevel components, virtualization, security

# Disk Encryption on Linux

- Multiple existing solutions
- Kernel: dm-crypt, cryptoloop, eCryptFS, fscrypt
- Userspace: encFS, VeraCrypt, ...

# dm-crypt, cryptoloop

- Work on top of block devices
- Encrypt individual blocks
- Single key for every block
- Full file contents and metadata encryption
- Not suitable for MTD
- But stacking is possible: MTD, UBIFS, container file, dm-crypt/cryptoloop, block filesystem

# eCryptFS

- Stacked on top of your regular filesystem
- Works on filesystem-level (i.e. “sees” files)
- Individual key per file
- File contents and filename encryption
- Stacking filesystems can be problematic
- Also overhead in performance and memory
- Usable on MTD/UBIFS

# encFS, VeraCrypt, ...

- Userspace filesystems
- Work on top of block devices
- Usually not what you want on (deeply) embedded systems

# fscrypt

- A better eCryptFS
- Fairly new, added for ext4
- Encryption baked directly into filesystem (no stacking overhead)
- Intended to encrypt individual directories (e.g. home directory)
- Google use case: Android, ChromeOS

# Why no dm-crypt-like Solution for MTD?

- No individual key per inode
- On NAND, empty pages need special handling
- Stacking
- On a second thought it didn't feel right

# fscrypt: History

- Initially merged as ext4 (filesystem-level) encryption in 4.1
- Intended for Android N to replace dm-crypt and get more flexibility for the smartphone use case
- Soon also added (basically copy-pasted) to F2FS
- Extracted and re-labeled as ‘fscrypt‘ by Jaegeuk Kim
- Lives in **fs/crypto**
- Currently maintained by Ted Ts’o and Jaegeuk Kim

# fscrypt: Features

- Encrypt individual directories by setting encryption policy
- Policy defines crypto algorithms and master key to be used
- Master key provided by userspace
- KEYS for key management (**keyctl**)
- **lookup()** and **readdir()** work without master key present
- Will return encrypted files names
- Useful to allow root to list & delete files without a user's key
- Integrated with **xfstests**

# fscrypt: Security

- Each inode has its own key derived from master key
- This is better than using a single key for everything (less encryption key re-use)
- If an attacker manages to break one file, only that file is exposed
- No metadata encryption (e.g. xattr), except for filenames
- No authentication of data
- Supported ciphers:
- **AES-256-XTS** for contents, **AES-256-CBC-CTS** for filenames
- **AES-128-CBC-ESSIV** for contents, **AES-128-CBC-CTS** for filenames, added by us for embeeded systems

# fscrypt: Filename encryption

- Ciphertext is non-ASCII (i.e. binary)
- **readdir()**, **lookup()** need strings not binary
- **fscrypt** uses base64-like encoding
- Problem: Base64 makes data longer, what if plaintext reaches **NAME_MAX**?
- Solution: Encoded name is: _ + **readdir()** cookie + truncated ciphertext

# fscrypt: Userspace API

- New ioctls: **FS_IOC_SET_ENCRYPTION_POLICY** and **FS_IOC_GET_ENCRYPTION_POLICY**
- Master key provided via **keyctl** syscall
- Key has to be of type **logon** and reside in an accessible keyring (e.g. session keyring)
- Userspace can only set or remove key, never read it back

# fscrypt: Userspace Tooling

- Responsible for deriving master key from password or similar
- **e4crypt**: as part of e2fsprogs, very basic
- **fscrypt**: (WIP) fully featured tool written in Go (https://github.com/google/fscrypt)
- **fscryptctl**: (WIP) minimal tool for embedded use case (https://github.com/google/fscryptctl)
- DIY

# fscrypt: UBIFS support

- Added by David Gstir and myself
- Landed with 4.10
- Actually more work than expected (as always ;-)
- Some changes to UBIFS and fscrypt were required
- Let's look at some of them

# fscrypt: Filenames can be binary (encrypted)

- UBIFS internally assumes that all filenames are strings, ever
- A simple **s/strcmp/memcpy/g** didn't do the job, obviosly ;-)

# fscrypt: **readdir()** cookies required

- UBIFS uses 32bit hash of filename
- Collisions do occur quite easily (birthday paradox)
- Resolved by **strcmp()** with filename
- Not possible when master key not present and encrypted filename is long (fscrypt truncates encoded name)
- Solution: extend hash to 64bit by concatenating random 32bit value
- Byproduct: NFS support (**seekdir()**, **telldir()**) for UBIFS!

# fscrypt: struct bio

- fscrypt was designed for the ext4 writeback code
- UBIFS writeback codes does not deal with BIOs
- fscrypt needed some small changes to work with plain buffers

# Usage with UBIFS

- Enable **CONFIG_UBIFS_FS_ENCRYPTION**
- Apply policy to (root) directory
- Load master key into the kernel (i.e. in an initramfs)

# Pitfalls: dentry and page-cache are shared, globally

- User **foo** can read **/home/bar/secret.txt** if it has the file permission to do so
- Even if **foo** does not possess the fscrypt key for **/home/bar/secret.txt**
- Requirement: user **bar** loaded his key once and read the file. It stays in the cache
- Even after bar logged out and the key was revokked
- Solution: You still need proper file permissions, also consider private mount namespaces

# Pitfalls: KEYS session keyring reset

- Key is shared across all users (encrypted root filesystem)
- **pam_keyinit.so** creates a new (empty) session keyring upon login
- Solution: Disable **pam_keyinit.so**
- Alternate solution: Implement a new keyring type for fscrypt/UBIFS (being discussed as of now)

# Pitfalls: deadlock with module loader

- Can happen upon first file access if **/sbin/modprobe** (kernel.modprobe sysctl) is encrypted
- If a cipher is requested for the first time the kernel tries to find the best implementation and calls **request_module()**
- **request_module()** itself runs the usermode helper **/sbin/modprobe** to find/load modules
- . . . which will trigger the cipher request again since we want to decrypt **/sbin/modprobe**
- Solution: Make sure that ciphers are loaded before you access an encrypted file
- In an initramfs a simple **stat /new_root/sbin/modprobe** before **switch_root** does the trick
- No, having all AES modules as builtin does not help
- Problem is known to crypto folks

# Pitfalls: Broken AES-CBC-CTS implementations

- ⋆ fscrypt uses the seldomly used ciphertext stealing mode for filename encryption
- ⋆ We found that the CAAM AES driver didn't decrypt such ciphertexts correctly, fix merged
- ⋆ Same bug present in Atmel AES driver, fix by Romain Izard under review
- ⋆ So if your filenames are suddenly in klingon, retry without crypto hardware drivers ;-)

# File authentification on Linux

- What to authentificate?
- File content?
- File attributes?
- Directory structure?
- Whole filesystem structure?
- Whole storage?

# File authentification on Linux

- Integrity Measurement Architecture (IMA)
- dm-integrity
- dm-verity

# File authentification on Linux

- Using DM on MTD is not possible (without ugly stacking)
- IMA works on UBIFS, thanks to Oleksij Rempel, Sascha Hauer!
- Future: Integrate authentification into fscrypt

# File authentification with fscrypt

- Was one of the goals of fscrypt but didn't materialize as of now
- Using authenticated encryption (AEAD) would help
- But just for file contents
- We needs more generic functions to auth structure

# Possible changes to UBIFS: HMAC

- Add HMAC to UBIFS common header
- We cannot enlarge the common header, it is part of the super block header
- UBIFS versioning will break
- A common footer would help
- But still bloats UBIFS
- I want it to be integrated with fscrypt (common key management and being generic!)

# Possible changes to UBIFS: Trusted index

- Utilize fscrypt
- Use authenticated encryption for data
- Just sign UBIFS branch nodes (and journal)
- Looks promising, needs to be prototyped

# VFS integration

- If state is bad, just returning **EIO** is not enough
- We need a way to query why data is no longer trusted
- Also a way is needed to inspect untrusted data
- Not UBIFS specific, needed for other filesystems too
- Think of btrfs data checksumming

# FIN

- Questions? Comments?
- **richard@sigma-star.at**