MASARYK UNIVERSITY

FACULTY OF INFORMATICS

Hardware-encrypted disks in Linux

Master's Thesis

ŠTĚPÁN HORÁČEK

Brno, Spring 2022

MASARYK UNIVERSITY

FACULTY OF INFORMATICS

Hardware-encrypted disks in Linux

Master’s Thesis

ŠTĚPÁN HORÁČEK

Advisor: Ing. Milan Brož, Ph.D.

Department of Computer Systems and Communications

Brno, Spring 2022

FACULTAS ARTIS INFORMATICÆ • UNIVERSITAS MASARYKIANA •

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Štěpán Horáček

Advisor: Ing. Milan Brož, Ph.D.

Abstract

This is the abstract of my thesis, which can
span multiple paragraphs.

Keywords

keyword1, keyword2, ...

Contents

1 Introduction

Just keeping the citations here, like [1], [2], [3].

1.1 Thesis description

The thesis aims to analyze existing approaches to using hardware-encrypted block devices (disks) in Linux and propose integrating such devices into the cryptsetup project.

The implementation should use the Linux kernel interface.

Student should

- get familiar with and study available resources for self-encrypted drives, OPAL2 standard, block layer inline encryption,
- analyze and describe security of such drives,
- provide state-of-the-art overview of existing attacks,
- implement proof-of-concept extension to cryptsetup project (and possibly propose changes for Linux kernel).

The student should be familiar with C code for low-level system programming and cryptography concepts.

2 Hardware disk encryption

Hardware disk encryption is a technology that provides confidentiality of data stored on a storage device using encryption provided by hardware.

Some general overview about what it is, I guess., maybe talk about both sw and hw enc instead

Describe generic stuff: provisioning, locking, key types DEK, KEK, MEK, processes, locking ranges vs. FDE, ... keyslots?,,, (and for opal TPer, SP, ..)

mention LUKS somewhere...

The disk encryption process can be conducted in logically and physically different places, depending on the type. In the following sections of this chapter, we will further describe three such types.

2.1 Self-encrypting drives

at least try to mention ATA security,,, which means also mentioning opal,,,, and TCG Enterprise...

2.2 Inline encryption hardware

Compared to the previously described self-encrypting drives, inline encryption hardware is separated from the disk, and

something about how it provides actually a way to check the encrypted content, right?

2.3 Software encryption

probably just a quick overview, ... maybe change the chapter to just "disk encryption"... or mention this just shortly at the start...

2.3.1 Linux stack

rethink where to put this... maybe tools?

dm-crypt

fscrypt

3 fscrypt

Assuming ext4 in this chapter...

3.1 How to make it work?

As of now, blk-layer inline encryption is supported only by two filesystems in Linux: ext4 and F2FS. Need to use a kernel with `CONFIG_FS_ENCRYPTION_INLINE_CRY` enabled. Need to first mount the filesystem with the inline encryption flag:

```
mount -t ext4 /dev/foo /mnt/foo -o inlinecrypt
```

It is not enough to specify the inline encryption flag, the encryption itself also must be enabled. Assuming ext4 filesystem, the encryption can be enabled after mounting like so:

```
tune2fs -O encrypt "/dev/loop0"
```

After this, `fscrypt` can be used as normal and it will use inline encryption for this filesystem.

In order to encrypt a folder using a `fscrypt`, the following must be done: an encryption key must be added and the encryption policy must be created.

```
int fd = open(pathname, O_RDONLY | O_CLOEXEC);

struct fscrypt_add_key_arg *key_request = calloc
   (1, sizeof(struct fscrypt_add_key_arg) +
   key_len);
struct fscrypt_policy_v2 policy_request = { 0 };

// add a key
key_request->key_spec.type =
   FSCRYPT_KEY_SPEC_TYPE_IDENTIFIER;
key_request->key_id = 0;
key_request->raw_size = key_len;
memcpy(key_request->raw, key, key_len);
ioctl(fd, FS_IOC_ADD_ENCRYPTION_KEY, key_request
   );
```

```
// set a policy
policy_request.version = 2;
policy_request.contents_encryption_mode =
   FSCRYPT_MODE_AES_256_XTS;
policy_request.filenames_encryption_mode =
   FSCRYPT_MODE_AES_256_CTS;
policy_request.flags =
   FSCRYPT_POLICY_FLAGS_PAD_8 |
   FSCRYPT_POLICY_FLAGS_PAD_16 |
   FSCRYPT_POLICY_FLAGS_PAD_32;
memcpy(policy_request.master_key_identifier,
   key_request->key_spec.u.identifier,
   FSCRYPT_KEY_IDENTIFIER_SIZE);
ioctl(fd, FS_IOC_SET_ENCRYPTION_POLICY, &
   policy_request);
```

This code sets up an encryption policy for the file specified by the pathname.

3.2 How does it work?

3.2.1 Setup

During mounting the "inlinecrypt"/`SB_INLINECRYPT` flag is written into the `super_block` structure.

It all starts in `__ext4_new_inode`. This is the internal function used when creating new inodes, called by functions such as `ext4_create` when creating a new file.

The function (if it is not inode used for large extended attributes?) calls `fscrypt_prepare_new_inode`.

`fscrypt_prepare_new_inode -> fscrypt_setup_encryption_info -> setup_file_encryption_key -> fscrypt_select_encryption_impl`

In `fscrypt_select_encryption_impl` there is actually the only place where the `SB_INLINECRYPT` flag is used. ... Calls `blk_crypto_config_supported` to check the device's crypto profile. Afterwards, `fscrypt_select_encryption_impl` function sets the `(fscrypt_info *)ci->ci_inlinecrypt`.

`setup_per_mode_enc_key` then sets the `(fscrypt_info *)ci->ci_enc_key`.

3.2.2 Usage

The bio function is stored in
`(struct bio *)bio->(struct bio_crypt_ctx *)bi_crypt_context`

Function `fscrypt_set_bio_crypt_ctx` changes the file's bio to use inline encryption... simply calls the blk layer `bio_crypt_set_ctx`.

Calling `submit_bio` like normally ... `__submit_bio` calls `__blk_crypto_bio_prep`...

"If the bio crypt context provided for the bio is supported by the underlying device's inline encryption hardware, do nothing."

`__blk_crypto_rq_bio_prep` however sets the context of the request to the one of the bio... After the bio prep `blk_mq_submit_bio` gets called (which calls `blk_mq_bio_to_request`, and after that also `blk_crypto_init_request->blk_crypto_get_keyslot` which updates the devices keyslot to contain the new key..., but does nothing if the device does not have keyslots)..... the info about the key to use then has to be acquired by the driver from `request->`

Most important are probably structures `blk_crypto_ll_ops` and `blk_crypto_profile`... just two operations, program key and evict key.

how to get crypto profile from outside..

Where does the hardware come to play?

`ufshcd_exec_raw_upiu_cmd()->ufshcd_issue_devman_upiu_cmd()->ufshcd_prepare`
sets the header with the correct keyslot.

4 TCG Opal 2.0

TCG Opal SSC (Security Subsystem Class) 2.0 is a specification for storage devices, aiming to provide confidentiality of stored data while the disk implementing this specification is turned off [1]. The Opal standard is one of the representants of the self-encrypting drive approach to hardware disk encryption.

developed by TCG (Trusted Computing Group)

maybe move under the SED chapter, but it is probably going to be too big for that..., and important

structure of the standards described in `https://trustedcomputinggroup.org/wp-content/uploads/TCGandNVMe_Joint_White_Paper-TCG_Storage_Opal_and_NVMe_FINAL.pdf`.

4.1 Technical stuff

I expect stuff here like important headers, codes, ...

SP = service provider, tables and methods that operate upon them

4.1.1 Capability discovery

begin with Level 0 discovery: (4, p. 3.3.6) specifies the general security receives command for all TCG storage devices [1, p. 3.1.1] specifies the Opal 2 specific parts of the header.

The Opal 2 feature header contains several important information usable in the future usage of the device. Most importantly the base ComID [4, p. 3.3.2].

ComID are divided into static and dynamic (managed by ComID management).

4.1.2 sessions

In order to facilitate communication between the host and SP, sessions must be used. During these sessions, methods are used...

example of method:

```
SMUID.StartSession [
HostSessionID : uinteger,
SPID : uidref {SPObjectUID},
Write : boolean,
HostChallenge = bytes,
HostExchangeAuthority = uidref {
   AuthorityObjectUID},
HostExchangeCert = bytes,
HostSigningAuthority = uidref {
   AuthorityObjectUID},
HostSigningCert = bytes,
SessionTimeout = uinteger,
TransTimeout = uinteger,
InitialCredit = uinteger,
SignedHash = bytes ]
=>
SMUID.SyncSession [ see SyncSession definition
   in 5.2.3.2]
```

Each method call is defined by the caller UID, method UID, and values of its parameters. The method calls are coded using tokens... start list, end list, optional argument start, ... short int, long int,....

To call a method use IF-SEND command, to get result of the method call use IF-RECV command.

In order to authentize the user, one can use many different ...

Methods are sent using packets. There are three types of packets: ComPackets, packets and subpackets. A single ComPacket can contain multiple packets and single packet can contain multiple subpackets. Each ComPacket is associated with a single ComID, each packet is associated with a single session.

4.1.3 setup

4.1.4 set locking range

global locking range is a special locking range that protects LBAs not protected by any of the other locking ranges.

4.1.5 access...

4.1.6 ...

4.2 Comparison with OPAL 1.0

OPAL 2 disks are *optionally* backwards compatible.

cite `https://trustedcomputinggroup.org/wp-content/uploads/TCG_Storage-Opal_SSC_FAQ.pdf`

4.3 Comparision with other TCG's SSC

Enterprise, Opalite, Pyrite, Ruby

5 Existing tools

5.1 Cryptsetup

... tool for disk encryption setup.
nothing for opal, right?

5.2 sedutil

supports opal, but the code seems to be abandoned

6 Security of hardware encryption

specify threat model, ,....,,, offline, online,,, offline multiple times???s

.. maybe change position of this chapter, since it's splitting tools and change to a tool

6.1 Attacks on hardware encryption

7 OPAL extension for cryptsetup

8 Conclusion

9 Notes

Write/read on locked range returns "Input/output error".
tested disks:

- SanDisk SD7UB2Q5
- Samsung SSD 850
    - problems with user authentication

ATA drives need `libata.allow_tpm` set to 1,,, but why?

templates — base, locking,, -> SPs. Service providers of the trusted peripherals are issued based on templates.

Global locking range is a special locking range that controls any area of the disk that is not controlled by any other locking range.

Shadow MBR allows FDE disk pre-boot. Presents "fake" MBR that can unlock the disk, on successful unlocking hands the control over to the real MBR.

key types (Credential Table Group):

- `C_PIN` — password
- `C_RSA_*` — signing arbitrary input???, session startup
- `C_EC_*` — ec-mqv, ec-dh session startup
- `C_AES_*`
- `C_HMAC_*`

5.3.4.1.3
authority operations:

- None — does not authenticate
- Password — authenticated using a pin
- Sign — challenge and response
- Exchange
- SymK

- HMAC
- TPerSign
- TPerExchange

Sessions may use secure messaging: use additionaly a start/sync-trustedsession methods for challange-response/key exchange

CVE-2018-12037 — "An issue was discovered on Samsung 840 EVO and 850 EVO devices (only in "ATA high" mode, not vulnerable in "TCG" or "ATA max" mode), . . . " — wtf is ata high/ata max mode???

CVE-2018-12038

s
e
p
a
r
a
t
o
r

All that follows is from the core standard[4].

single user mode — feature set that "locks" the admin out — admin can do only destructive actions upon the locking range, and only the user can actually unlock it

Trusted Computing Group (TCG), "Storage Work Group Use Case White Paper – v 1.0"
opal — "for corporate" opalite — worse opal, only global range, less admins and uesrs
pyrite — non-encrypted opalite, only logical access control
enterprise — only global range, no psid, no configurable access control, no pre-boot authentication, for scsi data centers devices

"Data confidentiality and access control over TPer features and capabilities: . . . The protection provided by this exclusive access extends to confidentiality of instructions and data in transit between the trusted host application (or a TPM it uses) and the TPer"

secure messaging

sessions are using readers-writers lock.

SPs are combinations of templates, must contain base template (). other templates admin, clock, crypto, locking, log

"All SPs incorporate at least a subset of the Base Template's tables and methods." -> is ∅ enough???

issuance = new SP from templates, personalization = customization of newly created SP (but can happen anytime...), initial data, authorities,

SSC = ... "A TPer MAY have only some of the capabilities (tables, methods, access controls, etc.) defined in this Core Specification and MAY include additional capabilities through table definitions and/or methods. A Security Subsystem Class SHALL NOT replace a capability called out in the Core Specification with the same capability implemented in different tables, methods, and access controls."

stream encoding - methods and tokens...,TLV...

tables – bytes tables – raw data, just bytes (rows addressed by row number); object tables (rows addressed by uid, SP-unique, non-reusable, for anti-spoofing)

object = row of table

SP templates – base, admin, clock, crypto, locking, log,

interface – protocol-independent, IF-SEND, IF-RECV commands

COMID

ComID identifies the caller, each application has different ComID and therefore can communicate simultaneously, dynamic, static. Not a session, multiple sessions may use single comid. Then there are some states and transitions, active, associated, issued... Extended ComID for reusing ComID so that we can see if it is the old ComID or the new one (using extra two bytes).

PROTOCOL LAYERS

protocol layers — session, management, communication, tper, interface, transport (... the standard seems to imply that the comid is always managed??)

DISCOVERY

discovery levels : 3.3.5 : level 0 , level 1 Properties method of TPer, level 2 of SP

SESSIONS

sessions: regular and control (not going to care about control much, just between TPer session manager and host session manager); read/read-write mutexes..., session manager methods - properties, start/sync session, start/sync trusted session, close session

*trustedsession - used with PuK, SymK, and HMAC authorities, secure messaging

authorities – used during session startup„, host exchange authority – exchange of session keys, implicit authentication;;; host signing authority – challenge response authentication/startup method integrity, `C_PIN` (password), , „, same for host->SP

METHODS

TRANSACTIONS

— like in database, commiting , etc.... optional (or implicit transaction on method level...), nested transaction commited when outermost finished, possible exceptions to rollbacks e.g. logs

Stream Flow Control – interface (handles sending IF-SEND/RECV commands across the interface) and stream data (handles not overwhelming host and TPer) types;;; uses Credit Control Subpacket to signify the opposite device that it is ready to receive data (and how much)

Session Reliability — ACKs, NACKs (SeqNumbers), timeouts„, all optional

Synchronous Interface Communications — alternative to the previous asynchronous communication... to make it more simple„„ so IF-SEND to make TPer do something, IF-RECV to get the result, repeat ad nauseam

SP OPERATIONS

Special SP - admin SP – maintains info about TPer and other SPs

SP – Cartesian product of template (defining tables and methods, pretty sure I wrote about it earlier, but it is mentioned in the standard again) subsets,

ACCESS CONTROL

invoker may have to know secret (secret + public part = Credentials, operation of proving knowledge = Authentication Operation, proving knowledge = Authentication)

explicit authentication - e.g. password validation, challenge response„, implicit authentication - e.g session key exchange

authority — object in authority table, type is individual or class (afaik just a bunch of individual authorities„ so that they may be easily changed at once, OR of all the individuals)

Access control – access control list consists of access control elements (boolean combination of authorities)„, each SP has AdminExch authority„, ACEs seem to be able to have different "owners"„,

ISSUANCE

issuance – creation of SP from template, from admin SP, „, personalization — after issuance, ussing the adminexch authority of the new sp, fill in tables, set access control... etc

LIFE CYCLE

... can be disabled (still can authenticate, deleteSP and set get on SPinfo (to enable)), frozen (any attempt to open session fails)

s
e
p
a
r
a
t
o
r

... And that§s the introduction... what follows now is the method and table overview...

Bibliography

1. *TCG Storage Security Subsystem Class: Opal*. 2022-01. Version 2.02. Standard. Trusted Computing Group. Available also from: `https://trustedcomputinggroup.org/resource/storage-work-group-storage-security-subsystem-class-opal`.
2. BIGGERS, Eric; TANGIRALA, Satya. *Inline Encryption*. 2021. Available also from: `https://kernel.org/doc/html/v5.17/block/inline-encryption.html`.
3. MÜLLER, Tilo; LATZO, Tobias; FREILING, Felix C. *Self-Encrypting Disks pose Self-Decrypting Risks: How to break Hardware-based Full Disk Encryption*. Available also from: `https://faui1-files.cs.fau.de/filepool/projects/sed/seds-at-risks.pdf`.
4. *TCG Storage Architecture Core Specification*. Tech. rep. Available also from: `https://trustedcomputinggroup.org/resource/tcg-storage-architecture-core-specification`.