Under Lock and Key: Using Hardware Protected Keys with the Linux Crypto API

Gilad Ben-Yossef
30 October 2019
Who Am I?

• I'm Gilad ben-Yossef.
• I'm a principal Software Engineer at Arm.
• I work on applied cryptography and security of the upstream Linux kernel in general and maintain the arm® TrustZone® CryptoCell® Linux device driver.
• I have been working in various forms with and on the Linux kernel and other Open Source projects for over twenty years.
• I have co-authored “Building Embedded Linux Systems” 2nd edition from O’Reilly.
The Linux Cryptography Sub-System
Or the Linux Crypto API, in short
Crypto API Usage Example

tfm = crypto_alloc_skcipher("xts(aes)", 0, 0); // Get a handle of a transformation that handles XTS mode of AES.

err = crypto_skcipher_setkey(tfm, key, sizeof(key)); // Set the key to be used for all subsequent operations

req = skcipher_request_alloc(tfm, GFP_KERNEL); // Get a request handle

skcipher_request_set_callback(req, CRYPTO_TFM_REQ_MAY_BACKLOG | CRYPTO_TFM_REQ_MAY_SLEEP, crypto_req_done, &wait); // Set the callback function to be called when done

skcipher_request_set_crypt(req, &sg, &sg, datasize, iv); // Set the input, output and initial vector buffers
ret = crypto_skcipher_encrypt(req); // Start the operation

err = crypto_wait_req(ret, &wait); // Wait for the operation to finish

crypto_free_skcipher(tfm); // Free things up

skcipher_request_free(req);
Display of data from `/proc/crypto`:

<table>
<thead>
<tr>
<th>Generic name</th>
<th>Driver's name</th>
</tr>
</thead>
<tbody>
<tr>
<td>name         : crc32</td>
<td>name         : xts(aes)</td>
</tr>
<tr>
<td>driver       : crc32-pcmul</td>
<td>driver       : xts-aes-aesni</td>
</tr>
<tr>
<td>module       : crc32-pcmul</td>
<td>module       : aesni_intel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority : 200</td>
</tr>
</tbody>
</table>

refcnt : 1
seltest : passed
type : shash
blocksize : 1
digestsize : 4

async : yes
blocksize : 16
min keysize : 32
max keysize : 64
ivsize : 16
geniv : <default>
Wait, back up a little...

// Set the key to be used for all subsequent operations
err = crypto_skcipher_setkey(tfm, key, sizeof(key));

... where is the key stored?
In RAM, like everything else.
What if someone gains access to my device and steals my key?
Hardware Protected keys

The Crypto Sub-Subsystem

Transformation provider 3

Dedicated Hardware

Dedicated Key Store

Transformation provider 2

Specialized Instructions

Transformation provider 1

Software

Crypto User API

DM-Crypt

IPsec
From Big Iron to much smaller embedded iron
Protected Keys Usage

Plain Key Decryption

Ciphertext → Key → Plaintext

Protected Key Decryption

Ciphertext → Tag → Storage → Key → Secure Domain → Key → Plaintext
Not a silver bullet

- Actual security depends on the security of the so called "Secure Domain"...
- Ability to use the key might be enough for attacker.
- Key provisioning and management is a problem as always.
- A good component in a "Defense in Depths" strategy.
Interface Details

Note: some parts are Arm CryptoCell specific

• The letter "p" is used as prefix to the generic algorithms name.
  • E.g. Use "paes" for Protected Key AES

• Because the tag value are implementation specific, use of a driver specific name is preferred over generic name, where possible.
  • E.g. Use "xts-paes-ccree" instead of "xts(paes)"

• Instead of the normal key, provide a tag appropriate for the implementation
  • E.g.

```c
struct cc_hkey_info {
    u16 keylen;       /* Length of actual key in bytes*/
    u8  hw_key1;      /* First key index */
    u8  hw_key2;      /* Second key index (optional) */
} __packed;
```

#define CC_HW_KEY_SIZE sizeof(struct cc_hkey_info)
Example: DM-Crypt with protected keys

```
# dmsetup create my_encrypted_volume \  <-- name of volume
  --table "0 $(blockdev --getsz /dev/sdb) \  <-- start and end offset of volume
    crypt \  <-- use the DM-Crypt target
    crypt:xts(paes)-plain64 \  <-- use the xts(paes) kernel cipher with 64 bit IV
    00200100 \  <-- use 256 bit AES keys from indices 0 and 1
    0 /dev/sdb 0"  <-- No IV offset, no volume offset
```

**Note:** cryptsetup can also be used if you want to password protect the key index with PKDF
Thank You
Danke
Merci
多谢
ありがとうございました
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה