ESG Social: Cryptography in the Linux Kernel

Josue Hernandez: 12 Nov 2024 @ HackerGarage

Overview

- *Cryptography basic concepts*
- *Linux Crypto Subsystem concepts*
- *Show me the code*
- *Q&A*

Cryptography basic concepts

Why do we need cryptography?

- Authentication
  - Message author is who pretends to be
- Data Integrity
  - Communication has not been modified/corrupted
- Confidentiality
  - Communication is not being listening for someone else

Cryptography basic concepts

Authentication

- MAC Message Authentication Codes
  - Mechanism used to authenticate the sender of a message
- HMAC hash based MAC
  - TLS Transport Layer Security

Cryptography basic concepts

Data Integrity

- It operates on random number of input data
- Generates unique fix-sized input
- SHA1, MD5 …

Cryptography basic concepts

Confidentiality

- Use one or more number of **keys**
- Can be **stream** or **block** oriented (CBC)
- Could be **Symmetric** or **Asymmetric**
  - Symmetric means the same key is used for encrypt and decrypt (AES)
  - Asymmetric means a different address is used for encrypt and decrypt (RSA)

Linux Crypto Subsystem concepts

Base structures

- Transformation implementation: represents an implementation of a specific algorithm (struct crypto_alg)
- Transformation object: an instance of a specific algorithm (struct crypto_tfm)
- Example: struct skcipher_alg is an extension of struct crypto_alg and struct crypto_skcipher is an extension of struct crypto_tfm

Linux Crypto Subsystem concepts

- Supports a whole bunch of algorithms Cipher, Hash, AEAD, HMAC and why not Compression.
- A transformation algorithm can be a template using basic building blocks
  - hmac(sha1): HMAC using SHA1 hash
  - cbc(aes): CBC using AES
  - authenc(hmac(sha1),cbc(aes)): AEAD using HMAC based on SHA1 and CBC based on AES
- Linux does not distinguish between hardware and software implementations.

Show me the code

Caesar Cipher

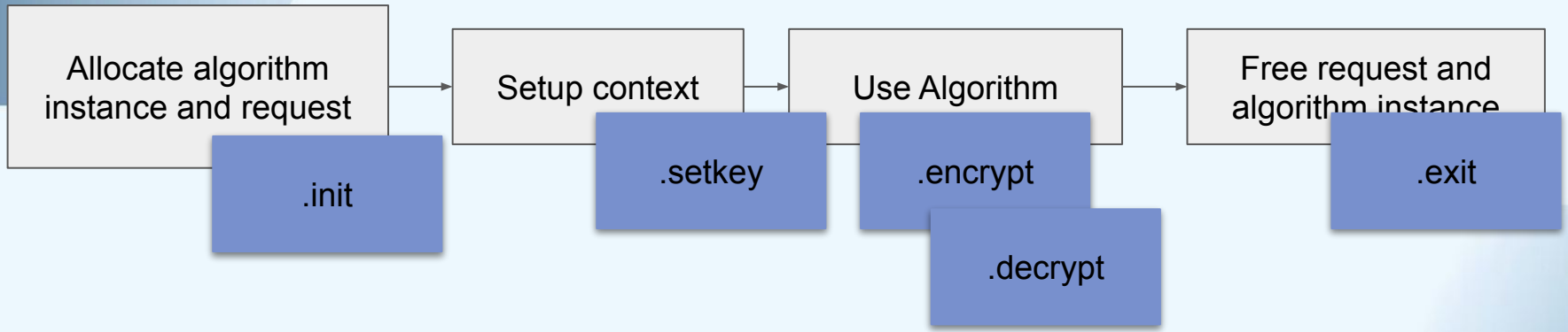
- It shifts letters a number of places right on encryption and left in decryption
- Use single key
- Could work on data streams

Show me the code

Steps to create a cipher algorithm

- Fill the cipher_alg structure with the needed data
  - For single key cipher struct skciper_alg
- Create functions to be called in every cipher step
- Function are called in this order
  - Request init (.init)
  - Setting the key (.setkey)
  - Encrypt function (.encrypt)
  - Decrypt function (.decrypt)
  - Request end (.exit)

Showcase


Allocate algorithm instance and request
.init
Setup context
.setkey
Use Algorithm
.encrypt
.decrypt
Free request and algorithm instance
.exit

Code

How do I use that?

- AF_ALG
  - Native in linux since 2.6
  - Easy to extend
  - To use it with openssl needs some extra work
- Cryptodev
  - It is the way used by openssl
  - “Say to be faster”
  - To extend needs to recompile Cryptodev kernel module
  - It is not in the kernel main repo

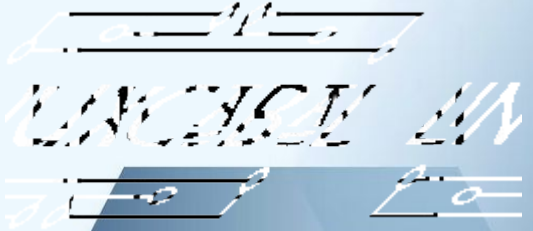
Code

References

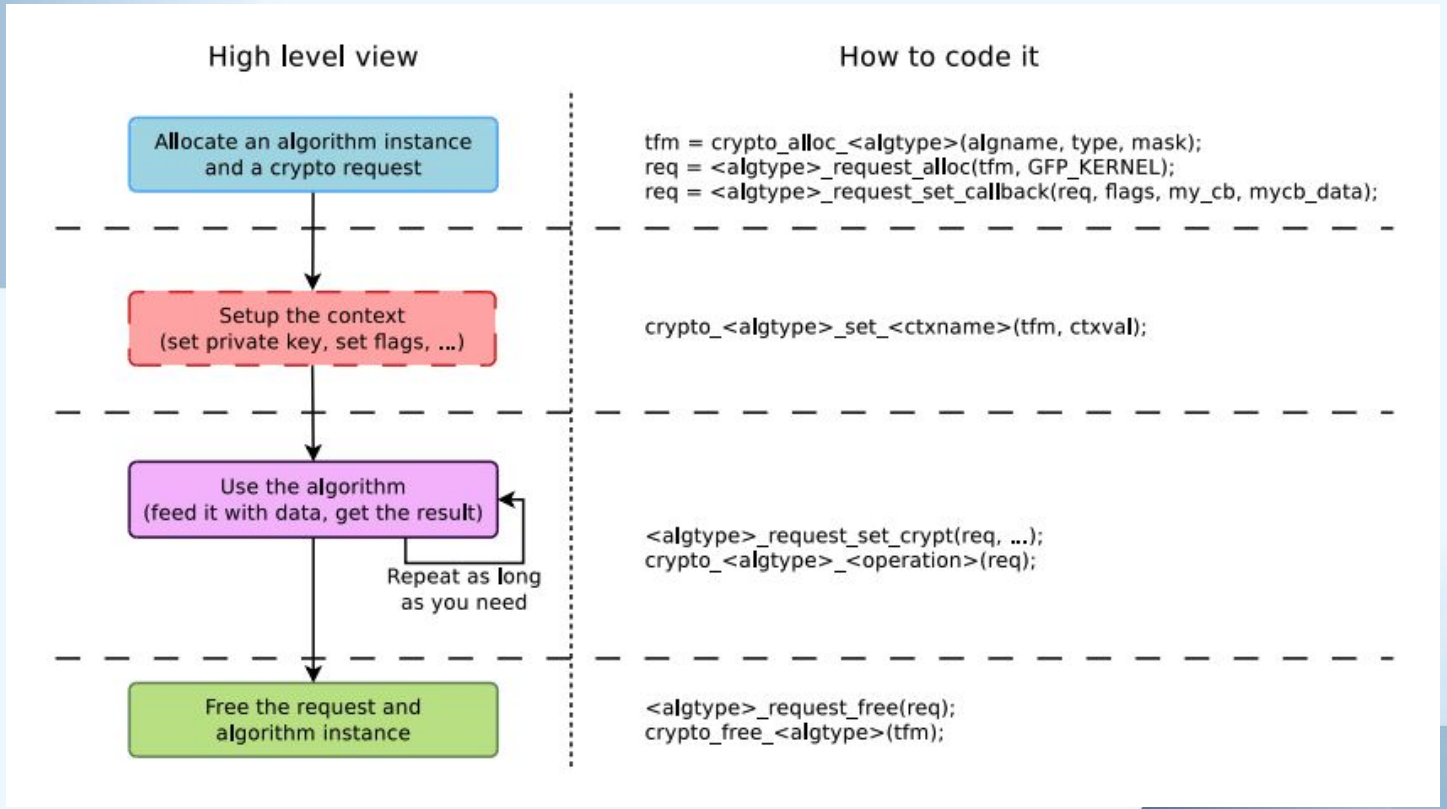
- Crypto API Linux https://www.kernel.org/doc/html/latest/crypto/index.html
- An Overview of the Linux Kernel Crypto Subsystem - Boris Brezillon, Free Electrons

Thank you

Contact:
me@josuedhg.com
https://github.com/josuedhg
josuedhg | josue_dhg on internet
Acruth for videogames



Showcase


High level view
How to code it
Allocate an algorithm instance and a crypto request
tfm = crypto_alloc_<algtype>(algname, type, mask);
req = <algtype>_request_alloc(tfm, GFP_KERNEL);
req = <algtype>_request_set_callback(req, flags, my_cb, mycb_data);
Setup the context (set private key, set flags, ...)
crypto_<algtype>_set_<ctxname>(tfm, ctxval);
Use the algorithm (feed it with data, get the result)
Repeat as long as you need
<algtype>_request_set_crypt(req, ...);
crypto_<algtype>_<operation>(req);
Free the request and algorithm instance
<algtype>_request_free(req);
crypto_free_<algtype>(tfm);