GELI — Disk Encryption in FreeBSD

Michał Borysiak
borysiam@gmail.com

November 15, 2018

Disk encryption facilities in FreeBSD

- GBDE (GEOM-based Disk Encryption)
  - FreeBSD 5, 2003
  - Poul-Henning Kamp
  - GEOM module in the kernel `gbde(4)`
  - User space tool `gbde(8)`
  - Creates new device with `.bde` suffix
- GELI (GEOM eli)
  - FreeBSD 6, 2005
  - Paweł Jakub Dawidek
  - GEOM module in the kernel
  - User space tool `geli(8)`
  - Creates new device with `.eli` suffix
- Operates on sector level
- New devices are created to allow plain text access to the data

The GEOM framework

- Standardized way to access storage layers
- FreeBSD 5, 2003
- Poul-Henning Kamp
- Set of GEOM classes
- Classes can be freely stackable in any order
- Abstraction of an I/O request transformation
- Transformations: striping, mirroring, partitioning, encryption
- Providers and consumers
- Auto discovery

GBDE

- **Master key** (2048 random bits) is located in a random place on the GEOM provider, and its location is stored in a **lock file**
- The **lock file** is encrypted using a **user password** and should be stored separately
- Up to 4 independent user secrets (**lock sectors**)
- Each sector is encrypted using AES-CBC-128 and a random **sector key**
- The **sector key** is encrypted using a key derived from the **master key** and the sector number
- Disk space overhead to store per-sector keys
- Non-atomic disk updates, since sector keys are stored separately from data
- Does not support mounting encrypted device in the / file system

GELI

- Simple sector-to-sector encryption
- To perform symmetric cryptography on sectors a random **master key** is chosen
- The **master key** is encrypted using **user key** and stored in the last sector of the GEOM provider
- Up to two encrypted copies of the **master key** can be stored in the sector
- **User key** consists of up to two components: a user passphrase and a key file
- Passphrase is strengthened using PKCS #5: Password-Based Cryptography Specification 2.0 (RFC 2898)
- Can perform verification of data integrity

GELI

- Automatically takes advantage of hardware acceleration of cryptographic operations thanks to utilization of the `crypto(9)` framework
- Supports multiple encryption algorithms (AES-XTS, AES-CBS, Blowfish-CBC, Camellia-CBC, 3DES-CBC) and different key lengths
- Allows to mount encrypted device in the / file system
- Since FreeBSD 11 supports booting from encrypted partitions

GELI full disk encryption before FreeBSD 11

- Some part of the system had to be left unencrypted (i.e. /boot directory)
- Together with a key file, this part was placed on a separate device which user always carried around (e.g. flash memory)
- Swap partition encrypted using one-time key

FS type	Mount point	Device
freebsd-boot		/dev/da0p1
freebsd-zfs	/boot	/dev/da0p2
freebsd-swap		/dev/ada0p1
		/dev/ada0p1.eli
freebsd-zfs		/dev/ada0p2
	/	/dev/ada0p2.eli

GELI full disk encryption since FreeBSD 11

- Thanks to Allan Jude boot loader can now perform GELI decryption
- Whole system can be installed on one ZFS pool
- Allows ZFS BE to be used with full disk encryption

FS Type	Mount point	Device
`freebsd-boot`		`/dev/ada0p1`
`freebsd-zfs`		`/dev/ada0p2`
	/	`/dev/ada0p2.eli`
`freebsd-swap`		`/dev/ada0p3`
		`/dev/ada0p3.eli`

GELI encryption in a ZFS volume

```
# Create a block device.
zfs create -V 256M zroot/test

# Create a random 4k key file.
dd if=/dev/random of=/tmp/test.key bs=4k count=1

# Initialize and attach encrypted disk.
geli init -K /tmp/test.key /dev/zvol/zroot/test
geli attach -k /tmp/test.key /dev/zvol/zroot/test

# A new device appeared.
ls /dev/zvol/zroot/test.eli

# We can create a new filesystem on the device.
zpool create -m /tmp/ztest ztest /dev/zvol/zroot/test.eli
```

GELI backup and restore metadata

```
# Backup GELI metadata.
geli backup /dev/zvol/zroot/test /tmp/test.eli

# Clear GELI metadata.
geli clear /dev/zvol/zroot/test

# Try to attach GELI device. It is not possible, since GELI
# cannot find its metadata on the device.
geli attach -k /tmp/test.key /dev/zvol/zroot/test

# Restore GELI metadata.
geli restore /tmp/test.eli /dev/zvol/zroot/test

# Now we can attach GELI device and import the pool.
geli attach -k /tmp/test.key /dev/zvol/zroot/test
zpool import
```

GELI resize the provider

```
# Resize ZFS volume.
zfs set volsize=512M zroot/test

# Now we cannot attach GELI device, because GELI cannot
# find its metadata on the device.
geli attach /dev/zvol/zroot/test

# We need to inform GELI about previous size of the device.
geli resize -s 256M /dev/zvol/zroot/test

# Now we can attach GELI device and import the pool.
geli attach -k /tmp/test.key /dev/zvol/zroot/test
zpool import
```

Thank you for your attention!