Disk Encryption

FreeBSD's GEOM offers excellent ways to protect data from unauthorized access. As long as the operating system is running access permissions and Mandatory Access Controls (MAC) protect the data. Traditional access controls of the operating system do not protect the data from users who have physical access to the disk however. If the disk is physically removed from the system and placed into another one, the data is not protected anymore. This is particularly dangerous for data on notebook disks.

The cryptographic GEOM classes *gbde* and *geli* can protect data on file systems even from highly motivated and skilled attackers with deep pockets. This protection works independent of how the way an attacker gained access to a disk or system, as long as the data is not decrypted because the disk is running in a system. Unlike other systems that are only capable of encrypting single files or data containers, *geli* and *gbde* can encrypt complete disks below the file system layer.

Features common to both systems:

– disjoint and therefore independently changeable passphrases that don't require reencryption of the data

– multiple independet keys (geli: 2, gbde 4)

– possibility to use a random key that is not stored anywhere

– possibility to use a passphrase and/or key as access control

Following we present some differences between the systems:

Encryption - gbde

Only AES is supported as encryption algorithm using a multikey-setup. The key sectors are protected with AES256, each other sector with its own AES128 key.

Zones are defined within the drive to spread logically contiguous data over the whole physical disk.

It supports passphrase and passphrase+key setups, but no configurations just with a keyfile.

Encryption - geli

Supports the crypto(9)-framework - if the system has cryptographic hardware available it is used.

Supports various algorithms with definable key lengths (currently AES, Blowfish and 3DES in a Single-Key Setup).

Offers a simple way to encrypt the root partition when the kernel is loaded from an external media.

Very fast due to simple sector-to-sector encryption and no spreading.

Optional verification of data integrity with HMAC.

Encryption - Swap

Encrypting swap space also protects sensitive information. E.g. if a program asks for a password and FreeBSD swaps data to disk to make space in RAM, your password may be written in cleartext to disk. You can encrypt swap with *gbde* or *geli* – you just need to change the fstab entry.

Additional Information

We strongly recommend using the excellent FreeBSD handbook:

http://www.freebsd.org/doc/en/books/handbook

For detailed information on GEOM please have a look at:

http://phk.freebsd.dk/pubs/bsdcan-04.slides.geomtut.pdf
http://phk.freebsd.dk/pubs/bsdcan-04.slides.geom.pdf
http://phk.freebsd.dk/pubs/bsdcan-04.slides.gbde.pdf
http://events.ccc.de/congress/2005/fahrplan/attachments/586-paper_Complete_Hard_Disk_Encryption.pdf
http://wikitest.freebsd.org/gvirstor

FreeBSD GEOM - English

FreeBSD GEOM

Introduction

GEOM is a modular and object oriented framework in the FreeBSD kernel through which all access to storage media has been controlled since FreeBSD 5.0. It is located between the file system and the two I/O-subsystems CAM and ATA and serves as an abstraction layer to the geometry of the actual media. Please note that GEOM is nearly unlimited in its possibilities and more classes are likely to come soon.

An Overview Of The GEOM Framework

The interesting parts of GEOM, for circles outside academia, are the available classes and their functionality. We will therefor concentrate on describing the, in our humble opinion, most important and most interesting classes and avoid discussing how the *geometry layer* (GEOM) works within the kernel itself. A full description may be found in the Daemon book.

RAID 0 - gstripe

RAID 0 means, as the name already implies, a RAID with zero redundancy. RAID 0 can however offer higher transfer rates by combining multiple hard disks and writing in parallel, also known as *striping*. If one the hard disks fails all data is lost however because parts (stripes) of data are missing. RAID 0 is therefore only recommended in setups where data availability is much less important than speed.

Using gstripe is simple: Load the module (# kldload geom_stripe), create a mount point, create a new media (# gstripe label), write a partition table with bsdlabel and create a file system with newsfs. Unfortunately the FreeBSD installer *sysinstall* is not yet capable of creating GEOM media; booting from gstripe-media is also not possible (yet).

RAID 1 - gmirror

A RAID 1 consists of a minimum of two hard disks, each containing exactly the same data (hence *mirror*). RAID 1 offers full redundancy of data. The smallest disk defines the maximum amount of storage available in a mirror. If one disk fails the other can still deliver data. RAID 1 offers the maximum in data availability, only the failure of all disks causes a loss of all data.

Using gmirror is as simple as using gstripe, can be used for root partitions and replacements of failed disks is easily done. All operations like error detection, disk failure detection and rebuilding the RAID 1 are done automatically.

RAID 3 - graid3

RAID 3 is a RAID 0 with redundancy, stored on a separate dedicated disk. Since a parallel and not an independent method is used, this parity disk is not a bottleneck like with RAID 4, a distribution of parity data is not necessary as with RAID 5.

Just like gstripe and gmirror, using graid3 is also very easy. The parity data is stored on the last disk, so the required number of disks is calculated with the formula (2^n + 1), e.g. 3, 5, 9, 17 etc.

Combined RAID-Levels

The classes gstripe, gmirror and graid3 can be freely combined. This way RAID levels 0+1, 10, 0+3 and 30 are all feasible.

RAID 5 - gvinum

RAID 5 offers higher throughput reading and writing data as well as redundancy at a low cost, making it one of the most popular RAID levels. User data and parity information are distributed over all disks. Since parity information is not required for reading operations, all disks are then available in parallel.

This advantage is however hardly noticeable reading small amounts of data, reading large files is however significantly accelerated. RAID 5 allows for the failure of one disk for data to be available.

Network - Geomgate

ggate is a GEOM class used to export providers via network. In contrast to NFS those drives shall not be mounted on the server and cannot be used by different concurrent clients.

The client is completely responsible for managing Geomgate, only pure I/O-requests are handled via the network.

Virtualisation - Gvirstor

gvirstor is a GEOM class which can define a virtual device independent of the physical devices and their usage. This is useful for server consolidation. You can define virtual devices of an arbitrary size, e.g. terabytes, consisting of any amount or size of physical media, e.g. disks, GEOM devices or RAID volumes.

The storage space is split into small chunks, e.g. 4 MB each, and allocated as needed. If the available storage space is full, any number of additional disks and GEOM devices can be easily and flexibly added. gvirstor is an important component of a Logical Volume Manager (LVM) later to come.

It is a loadable kernel module with its own utility and will later include an optional GUI. Emphasis has been placed on easy administration and monitoring of the physical media. gvirstor is in its final testing phase and will soon be included in FreeBSD.

FreeBSD: The Power To Serve

Journaling - gjournal

UFS 2, FreeBSD's standard file system, doesn't support journaling since it has Softupdates. Softupdates is an extension to UFS which enables meta data to be consistent without journaling and this with very small speed loss writing user data. This is done by sorting the meta data that needs to be written in a consistent order and then writing the whole block in one operation.

After a system crash it looks as if the meta data had been wholly written or not at all. Meta data is written in an atomic (indivisible) fashion. The file system is therefore always consistent.

The only errors that can occur are blocks in the file system that are marked as used although they're actually free. Therefore a file system check is required at each system startup. To shorten system startup time this check is done in the background (bgfschk). This can still be annoying with large disks however.

Therefore a journaling GEOM class has been developed which will probably be integrated in FreeBSD 6.3. gjournal runs below the file system layer on one or two disk devices and is independent of file systems and the write cache of a hard disk.

It is also flexibly combinable with other GEOM classes and faster than Softupdates, especially with small files because write operations of meta data are optimized. If it's run on a gmirror or graid3 even the synchronization is unnecessary because the data remains consistent.

Compression - geom_uzip

The geom_uzip framework allows read access to compressed disk images. You can save a lot of space on your disks while using only few CPU cycles reading the data.

Compressed images created with mkuzip(8) are offered to the kernel with the GEOM label geom_uzip as RAM disks with md(4).

geom_zip creates a unique md#.uzip device for each image which can then be used by the FreeBSD kernel to read it. Writing is not supported though.