A4: Encrypted Filesystem

2. Encrypting a single file using GnuPG

2.1

Scope of Encryption:

Stacked File System Encryption: This approach encrypts data at the file system level. Each file or directory is individually encrypted, and the encryption and decryption processes occur within the file system.
Block Device Encryption: This approach encrypts entire block devices, such as partitions or disks. The encryption is applied at the block level, and the entire device is treated as a single encrypted unit.

Stacked File System Encryption: Tied to the file system, and different file systems may have different encryption implementations. Changing the file system may impact the encrypted data.
Block Device Encryption: Independent of the file system. The entire block device is encrypted, making it file system agnostic.

2.2

lsmod # list all the loaded module
modprobe <mod-name> # to load a module
modprobe -r <mod-name> # to remove a loaded module.

creating the key-pair

gpg --full-generate-key
real name: shawkot
passphrase: 1234

list gpg key

gpg --list-keys

delete a gpg key

gpg --delete-secret-keys 0C2116AA5488111BE828961CE4ADDF04BDC74F99
gpg --delete-keys 0C2116AA5488111BE828961CE4ADDF04BDC74F99

exporting a gpg key

gpg --export --armor --output server1.pub

exchanging the public key

scp server1.pub vagrant@192.168.1.11:/home/vagrant

importing the key

```
gpg --import server1.pub

# verifying the key

gpg --edit-key blake@cyb.org

fpr
sign
check

# to add digital sign

gpg --output test2.sig --sign test2.txt

# to verify the signature
gpg --verify test2.sig

# to verify and decrypt a file
gpg --output test2 --decrypt test2.sig
```

2.3
The security of GPG is largely depends on the key management and key sharing. How can we verify if the owner of the key is the intended user or not. Also, using key server is another issue. If the server is compromised, then the whole security is gone.

2.4
GnuPG is a complete and free implementation of the OpenPGP standard also known as PGP. Over the past decade, PGP, and later OpenPGP, has become the standard for nearly all of the world's signed or encrypted email.
Both GPG and PGP adhere to the OpenPGP standard, which defines a standard for the encryption and signing of data. This standard ensures interoperability between different implementations.

2.5
Haveged is used to provide user-space entropy for randomness. It provides renewable source of random number to improve the quality of randomness available on a system.

problems with Haveged:
Resource usage and predictable entropy.

3. Crypto filesystem with loopback and device mapper

3.1
```
# creating a file with random bytes of data
dd if=/dev/urandom of=loop.img bs=1k count=32k

# attaching the file to a loopback device
sudo losetup -P /dev/loop5 /home/vagrant/loop.img

# formatting the loopback device with LUKS
sudo cryptsetup luksFormat /dev/loop6
sudo cryptsetup luksFormat --cipher aes-cbc-essiv:sha256 /dev/loop4
```

```
# mapping the device named as "encrypted_loop"
sudo cryptsetup luksOpen /dev/loop6 encrypted_loop

# creating filesystem
sudo mkfs.ext2 /dev/mapper/encrypted_loop

# mounting the filesystem
sudo mount /dev/mapper/encrypted_loop /mnt/lo-mount

# generating some file and directory
sudo touch /mnt/lo-mount/newfile
sudo mkdir /mnt/lo-mount/dir_new

# to close the LUKS volume
sudo cryptsetup luksClose encrypted_loop; for this, the volume should
be un-mount first.

sudo mount /dev/mapper/encrypted_loop /mnt/lo-mount/
sudo umount /mnt/lo-mount
```

3.2

Pseudo-device has no corresponding physical device, they are used by OS for different purposes like dev/mull, dev/random, etc.

A Loopback device is an example of a pseudo-device that is used as a block device by Unix systems to access files. Before that it must be attached to a file in the file system. The file, also referred to as loopback file, can contain an ISO image, a disk image, a file system, or a logical volume image and can be mounted as a disk device.

3.3

LUKS (Linux Unified Key Setup), it is a specification for disk encryption. It can be implemented using cryptsetup and dm_crypt. It is primarily used to encrypt entire disk partitions or volumes, ensuring that data remains confidential even if the physical storage media is compromised.

3.4

Strength:

Full Disk Encryption:

Data Security: Encrypting the entire loopback device provides comprehensive protection for all data stored on it.

System Integration:

Kernel-Level Encryption: Operates at the kernel level, providing a transparent and integrated solution that is accessible to all applications and users on the system.

Weakness:

Size Limitations:

Fixed Size: The size of the encrypted loopback device is determined at creation, and it may not dynamically resize like filesystems on a traditional disk.

single Key for Entire Device:

Single Passphrase: Typically, a single passphrase or key is used for the entire encrypted loopback device, which may not provide the same granularity as file-specific keys.

3.5

- Cryptoloop is deprecated due to security vulnerabilities
- Cryptoloop used a single key for the entire filesystemCryptoloop used a single key for the entire filesystem
- The encryption algorithm used by Cryptoloop (usually DES) was considered outdated
- Cryptoloop was not actively maintained, and as security vulnerabilities were discovered, they were not promptly addressed.

4. Gocryptfs

4.1

creating directories, mydir.crypt will use for encrypted data and mydir will be the mount point for decrypted data
mkdir mydir.crypt mydir

initializing gocryptfs to mydir.crypt directory
gocryptfs -init mydir.crypt

mounting the encrypted filesystem to 'mydir'
gocryptfs mydir.crypt mydir

unmounting the filesystem

fusermount -u mydir

4.2

encrypted loopback device with dm-crypt	**Encryption Layer with gocryptfs:**
dm-crypt is a kernel-level encryption module that provides transparent disk encryption functionality in Linux.	gocryptfs is a user-space encryption tool that operates at the file system level.
It operates at the block device level, encrypting entire block devices (e.g., partitions, disk images) rather than individual files or directories.	It creates an encrypted filesystem within an existing directory, encrypting individual files and directories rather than entire block devices.
Encrypted loopback devices are created by associating a regular file (or a block device) with a loop device and then encrypting the loopback device with dm-crypt.	gocryptfs uses FUSE (Filesystem in Userspace) to create a virtual encrypted filesystem on top of an existing directory.
Can be used for various purposes, including	Provides granular control over which files or

encrypting system partitions, external drives, or disk images.	directories are encrypted, allowing for selective encryption within a directory hierarchy.

gocryptfs is designed to be fully compatible with standard file systems
encFS provides a virtual file system that appears like any other file system. It may leak metadata or file size. Use old encryption method.

5. TrueCrypt and alternatives

5.1
Truecrypt is no longer maintained, it has some known vulnerabilities, it uses 1000 iterations for KDF.
-- Encryption keys stored in memory
-- TrueCrypt cannot secure data on a computer if it has any kind of malware installed. Malware may log keystrokes, thus exposing passwords to an attacker.
-- According to a study released 29 September 2015, TrueCrypt includes two vulnerabilities in the driver that TrueCrypt installs on Windows systems allowing an attacker arbitrary code execution and privilege escalation via DLL hijacking.

Veracrypt is regularly maintained, audited by third parties and the number of iteration is way more larger than Truecrypt.

5.2

Create a new volume:
veracrypt -t -c vercry
Mount a volume:
veracrypt vercry veramount/

Mount a volume without mounting its filesystem:
veracrypt --filesystem=none vercry veramount/

List all mounted volume:
veracrypt -l
Dismount a volume:
veracrypt -d vercry

When using the text user interface, the following procedure must be followed to create a hidden volume:

1) Create an outer volume with no filesystem.
2) Create a hidden volume within the outer volume.
3) Mount the outer volume using hidden volume protection. (--protect-hidden=yes|no)
4) Create a filesystem on the virtual device of the outer volume.
5) Mount the new filesystem and fill it with data.
6) Dismount the outer volume.

If at any step the hidden volume protection is triggered, start again from 1).

--volume-type=TYPE
Use specified volume type when creating a new volume. TYPE can be 'normal' or 'hidden'. See option -c for more information on creating hidden volumes.

A hidden volume can be mounted the same way as a standard VeraCrypt volume: Click Select File or Select Device to select the outer/host volume (important: make sure the volume is not mounted). Then click Mount, and enter the password for the hidden volume. Whether the hidden or the outer volume will be mounted is determined by the entered password (i.e., when you enter the password for the outer volume, then the outer volume will be mounted; when you enter the password for the hidden volume, the hidden volume will be mounted).

--protect-hidden=yes|no
Write-protect a hidden volume when mounting an outer volume.
Before mounting the outer volume, the user will be prompted for a password to open the hidden volume. The size and position of the hidden volume is then determined and the outer volume is mounted with all sectors belonging to the hidden volume protected against write operations. When a write to the protected area is prevented, the whole volume is switched to read-only mode. Verbose list(-v -l) can be used to query the state of the hidden volume protection. Warning message is displayed when a volume switched to read-only is being dismounted.

5.3
In case an adversary forces you to reveal your password, VeraCrypt provides and supports two kinds of plausible deniability:

1. Hidden volumes (see the section Hidden Volume) and hidden operating systems (see the section Hidden Operating System).

2. Until decrypted, a VeraCrypt partition/device appears to consist of nothing more than random data (it does not contain any kind of "signature"). Therefore, it should be impossible to prove that a partition or a device is a VeraCrypt volume or that it has been encrypted (provided that the security requirements and precautions listed in the chapter Security Requirements and Precautions are followed).

A possible plausible explanation for the existence of a partition/device containing solely random data is that you have wiped (securely erased) the content of the partition/device using one of the tools that erase data by overwriting it with random data (in fact, VeraCrypt can be used to securely erase a partition/device too, by creating an empty encrypted partition/device-hosted volume within it). However, you need to prevent data leaks (see the section Data Leaks) and also note that, for system encryption, the first drive track contains the (unencrypted) VeraCrypt Boot Loader, which can be easily identified as such (for more information, see the chapter System Encryption). When using system encryption, plausible deniability can be achieved by creating a hidden operating system (see the section Hidden Operating System).