OBL4-OS

August 12, 2020

This is a mandatory assignment. Use resources from the course to answer the following questions. **Take care to follow the numbering structure of the assignment in your submission**. Some questions may require a little bit of web searching. Some questions require you to have access to a Linux machine, for example running natively or virtually on your own PC, or by connecting to `gremlin.stud.iie.ntnu.no` over SSH (Secure Shell). Working in groups is **permitted**, but submissions must be **individual**.

1 File systems

1. Name two factors that are important in the design of a file system.

> **Solution:**
>
> - Security - adequate permissions and access control for security and privacy.
> - Efficiency and performance (e.g., block size) - a small block size is more efficient for many small files, while being inefficient for large files. A large block size on the other hand can waste space if there are many small files. Also relelvant here is the idea of spatial locality.
> - Reliability - power failure, resistance to file corruption or data loss, redundancy.
> - Named data - allow users to use logical naming, and maintain a mapping between logical and physical (on disk) layouts.

2. Name some examples of file metadata.

> **Solution:**
>
> - File size
> - Creation, modification, access date/time
> - Owner
> - Permissions

2 Files and directories

1. Consider a Fast File System (FFS) like Linux's ext4.

(a) Explain the difference between a hard link and a soft link in this file system. What is the length of the content of a soft link file?

> **Solution:** Hard links are multiple file directory entries that map different path names to the same file number (e.g., i-number), in other words a mapping between a path name and a file number. The file is only deleted when the last hard link to it has been remove. A soft link is a link from one path name to another path name. The content of a soft link file is the path name it points to, thus the size is equal to the length of that path.
>
> A benifit of soft links over hard links is that they can be used to link to folders and elements which exist in other file systems (e.g., different partitions or disks, network shares, removable devices, etc.).

(b) What is the minimum number of references (hard links) for any given folder?

> **Solution:** A folder is a mapping from its path name to its i-number, so that's one reference, as well as a mapping from the special name ".", which is a second reference. So the minimum is two.

(c) Consider a folder `/tmp/myfolder` containing 5 subfolders. How many references (hard links) does it have? Try it yourself on a Linux system and include the output. Use `ls -ld /tmp/myfolder` to view the reference count (hint, it's the second column in the output).

> **Solution:**
>
> ```
> $ mkdir /tmp/myfolder
> $ cd /tmp/myfolder
> $ for I in `seq 0 4`; do mkdir folder-$I; done
> $ ls -l
> total 20
> drwxr-xr-x 2 donn donn 4096 april  5 13:14 folder-0
> drwxr-xr-x 2 donn donn 4096 april  5 13:14 folder-1
> drwxr-xr-x 2 donn donn 4096 april  5 13:14 folder-2
> drwxr-xr-x 2 donn donn 4096 april  5 13:14 folder-3
> drwxr-xr-x 2 donn donn 4096 april  5 13:14 folder-4
> $ ls -ld .
> drwxr-xr-x 7 donn donn 4096 april  5 13:14 .
> ```
>
> Notice the second column in the output from `ls`. The last line shows the folder `myfolder` has 7 references, one for each of the subdirectories, one for ".", and one for its own folder name.

(d) Explain how spatial locality is acheived in a FFS.

> **Solution:** Spatial locality is acheived during the allocation of i-numbers upon file or directory creation. The file system tries to allocate i-numbers in groups according to logical file layout, such that they may exist near each other physically on the disk.

2. NTFS - Flexible tree with extents

(a) Explain the differences and use of *resident* versus *non-resident* attributes in NTFS.

Solution: The master file table, containing records for each file and directory in the file system, may not have enough space in its record for some larger attributes (e.g., a data extent). In this case an extent is allocated to hold the attribute, and a pointer to that extent is stored in the MFT record.

(b) Discuss the benefits of NTFS-style extents in relation to blocks used by FAT or FFS.

Solution: Extents are variable-sized regions of files that are each stored in contiguous regions on the disk. The depth of the index tree for a particular file can be shallow even for large files, because the extents can be larger. Deeper trees are only necessary if the file becomes badly fragmented.

3. Explain how copy-on-write (COW) helps guard against data corruption.

Solution: Copy-on-write (COW) writes new versions of files to new locations on the disk, so an inherent transaction is performed where the i-node can be updated only when the file has been completely written, thus minimising data corruption by partially overwriting existing portions of the file.

3 Security

1. Authentication

(a) Why is it important to hash passwords with a unique salt, even if the salt can be publicly known?

Solution: Since a hash function (e.g., SHA-256) is one-way, it is computationally infeasible to find the original plain-text of a hashed password. However, due to the large amount of plain-text passwords available online stemming from data leaks, it is easy to generate hash values for these known passwords. Large lists of hash-password pairs are now available, making password cracking much easier. If a unique salt is added to the password (we can just concatinate the salt and the password together), the hash will be unique. We keep the salt with the hashed value, so that we can calculate the password hash again for authentication. If the salt and the hash are leaked, it is infeasible to find the original password, again, because a hash is a one-way function.

(b) Explain how a user can use a program to update the password database, while at the same time does not have read or write permissions to the password database file itself. What are the caveats of this?

Solution: We can give a program setuid permissions, specifically setuid root, such that the program runs as an administrative user, who has permissions to read and write the password file. The caveats are that we must be careful about which programs we allow to be setuid root, and that these programs are themselves free of vulnerabilities which could permit the user to read or write arbitrary files on the system.

2. Software vulnerabilities

(a) Describe the problem with the well-known `gets()` library call. Name another library call that is safe to use that accomplishes the same thing.

Solution: `gets()` receives input from the user into a buffer (located either on the stack or the heap), but does not perform any bounds checking, so is dangerous. A malicious user could send a large amount of input which would overwrite the end of the buffer, clobber other variables on the stack, and in the worst case overwrite the return pointer to point to some user-provided shellcode, permitting arbitrary code execution.

(b) Explain why a microkernel is statistically more secure than a monolithic kernel.

Solution: A microkernel has fewer lines of code that runs in kernel-mode than a monolithic kernel. The downside is there is more user- to kernel-mode switches, but the benefit is that there is a lower chance of vulnerable code executing in kernel-mode.