TDIU25 Exam

Klas Arvidsson

2015-03-17 14-18
TER1, TERD

Admitted material

Dictionary from English to your chosen language.

Jour

Klas Arvidsson (013-282146) visiting after about one hour.

Instructions

- Fill in the exam wrapper and read the instructions on it before you start.
  Read instructions and all assignments carefully and completely before you begin.
- You may answer in either English or Swedish.
- State your interpretation of the question and all assumptions you make.
- Write clearly. Unreadable text will be ignored.
- Be precise in your statements. **Prove your point when possible.**
  Unprecise or ambiguous formulations will lead to reduction of points.
- **Motivate clearly and in depth all statements and reasoning.**
  Explain calculations and solution procedures.
- The exam is 40 points and graded U, 3, 4, 5 (preliminary limits: 21p, 28p, 34p).
  Points are given for motivations, explanations, and reasoning.
- **Use Round Robin with 15 minutes time quanta to solve the exam!**

Definitions

Unless you are more specific, the correcting team will interpret the following terms as follow:

memory Volatile random access memory (DRAM), about 100ns access time.

disk Permanent storage, about 10ms access time.

page A fix size region of virtual memory, possibly on disk.

frame A fix size region of physical memory (DRAM).

block A data block located on disk.

Problem 1 (8p)

It was recently (2015-03-09) shown that a rowhammer attack can give an attacker full control of the target system. In reality the attack triggers memory accesses that (due to crosstalk) induce a higher voltage in adjacent memory cells. Eventually this may cause susceptible bits to flip from zero to one.
In this assignment we will simply assume the attacker can flip a set of random bits in DRAM memory. The kernel stores active parts of every process page table in DRAM memory.
Our target system uses paging in 4 levels (x86-64) with virtual addresses of 48 bits and each page table entry using 64 bits. The page size used is 4 KiB. Consider the system without caches.

(a) How do you calculate the maximum and minimum size required by page tables (all levels) for a process? To get the minimum, assume only one page is used. To get the maximum, assume the process uses its entire address space. (4p)

(b) Consider a page with known content. A kernel mode page table is used to map it to a frame. How can you as attacker distinguish a bit flip that occur in the frame number for the page (in page table) from a bit flip that occur in page data, from no bit flip at all? You have no way to access the page table. Explain in depth. (4p)

Problem 2 (8p)

```
program type P {
  create a 4 KiB memory buffer
  open a file for reading
  until end of file do {
    read data from file to fill the buffer
    calculate checksum from buffer
  }
  close the file
}
```

Assume you simultaneously execute two I/O bound processes, P_1 and P_2, like program P above, and one CPU bound process Q with infinite execution time. P_1 and P_2 were started with two different large files. Each file is sequentially allocated on disk. The two files are located far apart on a disk with 10ms access time. But sequential access time is just 500µs.

(a) Explain with an example schedule in which way *you would like*[1] to execute the processes to optimize overall system performance. Assume a system with no caches. (4p)

Multilevel queue scheduling is used. In particular, you have two ready-queues available. Processes that go to a wait queue will eventually go back to the ready-queue they came from. You are free to choose the scheduling policy to use in each ready-queue, and you can choose the policy that determines which of the two ready-queues that should receive CPU-time. *More precisely, on any type of queue event, you may choose which ready-queue to schedule from next.*

(b) Which scheduling policies would you use to obtain a schedule as close to your suggestion in (a) as possible, and in which queue would you place each process? State specifically how you select next process when an I/O bound process must wait for the disk, and when the I/O is complete. (4p)

[1] You are not bound to any existing algorithm or policy. Make human choices based on all given information. Your example schedule do not have to be complete or to scale. Just show and explain your main point.

Problem 3 (12p)

Consider program type P from previous assignment. You have an eight-core processor and want to perform the user-mode checksum calculation in 8 different threads. Each thread will get 1/8 of the buffer to work on. In the optimal case you will now have 8 calculations going on in parallel, one och each core. Each thread must be possible to pause in favor of any higher priority process in need of that core.

(a) Would you use 1:1 or 1:N mapping of kernel to user threads? Motivate clearly, compare for example how each model handle a tell-tale scenario. (3p)

(b) Consider a system with an operating system capable of the described threading. To eliminate the need for kernel threads, it uses the user mode thread stack during execution of system calls. In particular, the kernel mode functions use the stack to save their respective parameters, return address[2], and local variables. All kernel data are cleared from the stack as soon as the kernel mode execution is done with the system call. Provide either a set of restrictions that you think would make such stack usage safe, or provide example of an attack to get your program kernel mode privileges. (4p)

(c) A program of type P is used to record checksums for all files in the system. Any user should be able to use P to verify the checksum of any file. Explain how you would make sure P can access any file. Your solution should be as secure as possible, and are rated according to *principle of least privilege* and *need-to-know*. The system does **not** support `setcap` (neither Solaris Privileges, Linux Capabilities nor similar privilege system). (5p)

Problem 4 (12p)

Your friend has an idea for a new file system. Each file control block will contain a number of entries. Each entry will consist of (P, N) where P always is a pointer to N sequential blocks of file data. Your friend has great hope to find high access speed, high storage efficiency, low overhead, no limitations, and super reliability. This file system will be used on a system with some very popular files that are used simultaneously by many processes.

(a) Your friend also suggests to clean up the obsolete system call interface. The argument is that `open` and `close` are not needed today: instead we provide the file name to `read` and `write`. If you agree, explain all steps the new implementation of read must go through. If you disagree, provide in depth argumentation that `open` and `close` are important. (5p)

(b) Do you think the described file system could prove a good file system? Motivate in depth. Cover at least the main strength and the main weakness. (5p)

(c) The file system reminds you of a DRAM memory management technique. Which and why? (2p)

[2] An instruction pointer saved before entering a function. Upon completion of the function the CPU will continue execution from this address.