Instituto Superior de Engenharia de Lisboa

BEng in Computer Science and Engineering

System Virtualization Techniques, Autumn/Winter 2023/2024

Partial Test #1

ATTENTION: Answer questions **1** and **2** on one set of sheets and questions **3**, **4**, and **5** on another set.

1. [6] Consider the code that will be compiled into the executable **denum** and, on the right, the file **in.txt**:

```
int main(int argc, char * argv[]) {
  int c;
  while ((c = fgetc(stdin)) != EOF)
    if (!(c >= '0' && c <= '9'))    // c is not a digit
      fputc(c, stdout);
  return 0;
}
```

```
ISEL-LEIC-TVS
31! T1V3S73 20?
Lisbon, PT
31-Oct-2023
```

   a. [1.5] Explain what is printed on the *standard output* with the execution of the command line below:

      `denum < in.txt | grep TVS` ← *same command for subquestions **b** e **c***

   b. [1.5] For the command line above, how many processes are created, and for each of them, what do the entries 0 and 1 in their respective file descriptor tables refer to?

   c. [3] Write a C program to execute the command line in ***a***, using the POSIX API for creating/opening files, launching processes, connecting them as a shell would do, and then waiting for their completion.
      NOTA: *Using, directly or indirectly, a shell instance in this answer will result in a rating of zero.*

2. [4] In a Linux system with an x86-64 processor, a process has the address space configured such that the following virtual addresses are accessible in user mode with the permissions indicated next to them:
   – 0x0002468ACE0468AC r__ *(Read OK , Write NOK , Execute NOK)*
   – 0x0002468ACE246321 rw_ *(Read OK , Write OK , Execute NOK)*
   – 0x0002468ACE247753 r_x *(Read OK , Write NOK , Execute OK)*

   a. [1] Is the processor operating with 4 or 5 levels of translation? Justify.

   b. [3] What are the available permissions in user mode for the following addresses? Justify.
      0x0002468ACE2468AC 0x0002468ACE247321 0x1112468ACE247753

3. [2.5] The sentences below are taken from the Architecture Reference Manual for ARM64 processors:

   «The SVC instruction causes a Supervisor Call exception. This provides a mechanism for unprivileged software to make a system call (...).» (B1.3.1)

   «A Supervisor Call enables software executing at EL0 to make a call to (...) software executing at EL1.» (D1.3.9)

   a. [1] Knowing that EL0 and EL1 correspond to privilege levels in ARM64 processors, based on the text, indicate which of the values is the most privileged and which is the least privileged.

   b. [1.5] The SVC instruction is described as an exception and it does **not** have as an argument the code address where to transfer execution. What is the relevance of this and also, in the specific case of Linux, how does the system know which of the various possible system calls is to be executed?

4. [3.5] Consider the following source code of a library (file with `.so` extension or *shared object*) that is intended to be loaded using the `dlopen` function :

```
int count = 1;
void * xalloc(size_t npages) { ++count; return mmap(NULL, npages * 4096,
                  PROT_READ | PROT_WRITE, MAP_PRIVATE | MAP_ANONYMOUS, -1, 0); }
void   xfree(void * ptr) { --count; munmap(ptr); }
```

   a. [1.5] A process loaded the library and only later invoked `xalloc(4)`. What changed in the process's address space (which is visible in `/proc/<pid>/smaps`) with this invocation?

   b. [2] If the process invokes `dlclose` when the value of count is 2, what will change in the address space (visible in `/proc/<pid>/smaps`)?

5. [4] The following C program was compiled and executed on a Linux system with an x86-64 processor :

```
 1  const char * test = "ISEL - LEIC - TVS";
 2  char u[5000];
 3  char x;
 4  char y;
 5  int main(int argc, char * argv[]) {
 6     x = 10;
 7     y = 20;
 8     getchar();
 9     test[5] = '+';
10     return 0;
11  }
```

   a. [1] It is observed that the execution of line 6 and the execution of line 7 do not cause an equal change (in *bytes*) in the *resident set size* (`Rss`) of the memory region corresponding to `.bss`. What is the value of the change for line 6, and what is the value for line 7? Why are the values different?

   b. [2] With the execution paused at line 8, another instance of the same program was executed in another terminal, which also stops at line 8. In both processes (of the same program), the *resident set size* (`Rss`) of the memory region corresponding to the `.text` section is 4KiB..

      i. [1] What is the sum of the *proportional set size* (`Pss`) of the two **`.text`** regions? Explain why.

      ii. [1] How does the `Pss` value compare to `Rss` in the regions corresponding to **`.bss`** in each of the processes, with both still paused at the same point? You may write the answer as a ratio `(Rss / Pss)` or similar.

   c. [1] The execution of line 9 results in the abrupt termination of the program's execution. In machine code, this instruction consists of only a memory write with a `movb` instruction. What check fails in the CPU, allowing the detection of the invalid access to index 5 of the constant string?

Duration: 75 minutes

ISEL, October 31st 2023