Übungsblatt 07

Maurice Donner, Ise Glade

Aufgabe 1

1 Pagefault for accessing the word

There are no other interactions with RAM

Aufgabe 2

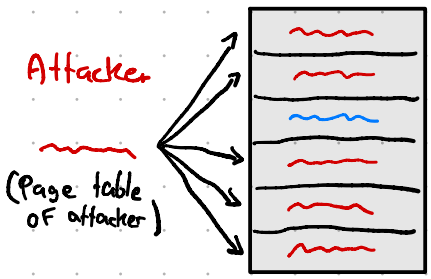
a) The rowhammer problem is based on bit flips occuring in rows of memory, when adjacent memory is repeatedly accessed.

For this to happen, one must repeatedly access and flush two different rows of the same bank of DRAM
(banks are groups of DRAM chips whose rows are activated in lockstep)

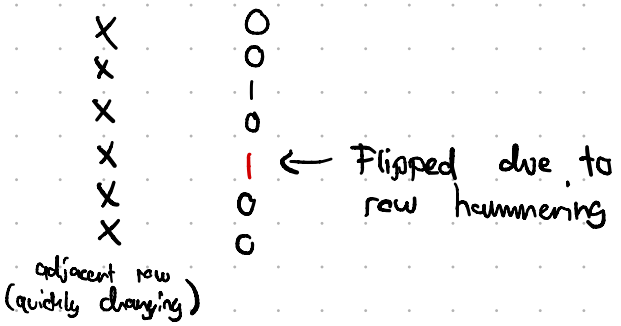
A targeted attack would focus on flipping the bit that is responsible for granting memory page permissions. This way a process could gain acces of the entire physical memory

b) Kernel privilege escalation is the process of changing the bit of a page-table entry, which causes it to point to a physical page containing the page table of the attacking process.

Step 1: Spray physical memory



Step 2: Flip Bit of page table entry



→ If row based on chance the Page table entry points to one of the attackers page tables (~~) the attacking process gains read- and write access to one of its own Page tables !!! Essentially being able to access all physical memory.

The process that is initially able to access the page table of the attacking process is the scheduler. Only the OS should have permission to these tables.

c) You only need to change the page that a PTE points to. this is stored in the page address. Flipping a bit in this address can vastly change where a PTE points to

This exploit requires to spray most of the system's available memory with page tables. This is both possible for normal and inverted page table.

In fact, it has been experimentally tested on linux systems, that use a 4-level page table approach

Aufgabe 3

The drives are connected to an Arduino, that directly uses the IDE pinout of the drives to access their mechanical properties. They use 3 pins to control the drive motor and the LEDs:

Pin 11: Enables + Disables Motor + LED
(Drive Select)
Pin 20: Moves the motor by one step
(Step)
Pin 18: Controls the direction of the motor
(Dir)

→ By applying read/write operations in a frequency that matches musical notes, and applying proper timings, Songs can be played on the drive motors.

Aufgabe 4

The model designates an owner and a group (all is always the same). These three "levels" are appointed the authority to read (r), write (w) and/or execute (x). This way all necessary permissions could be organized with as little as 9 bits (rwxrwxrwx).

Hardlinks and softlinks can both have different permissions than the affiliated file/directory, e.g. when a specific file needs to be viewed by all in an environment with stricter permissions.

Aufgabe 5

/usr/var/lehre/ibn/u607-ibn2022.pdf

↑ Root directory

usr → var → lehre → ibn → pdf

The I-Node for usr was already in memory from the beginning
Since each directory fits into one cell, we only need to read each inode once.
→ In total 4 Inodes have to be read

Aufgabe 6

There are several reasons for counting accesses of a file. Some of them could be

- Redundancy: If several processes access files at the same time, this can help avoid merge collisions.
- Efficiency: Knowing that a file is accessed much more frequently than others, making sure it can be loaded quickly can save valuable time. (For example by moving it into the primary block of an inode, reducing the number of reads required)

Aufgabe 8

a) Average rotation required $1/2\ r$

Rotation speed: 1500 rpm

$\rightarrow \overline{t_{rot}} = \frac{1/2\ r}{1500\,rpm} = 0.02\,s$

b) $\overline{t_{rot}} \approx 20ms$, $\overline{t_{seek}} = 85\,ms$ $\rightarrow \overline{t_{tot}} = 105\,ms$

chance for a cylinder change: 5/6

$10 \cdot (20ms + \frac{5}{6} 85ms) = 908\ ms$

c) $1/2\ r / 8{,}333ms \approx 3600\ rpm$

Aufgabe 9

$125 \xrightarrow{18} 143 \rightarrow$

86
1470
913
1774
948
1509
1022
1750
130

FCFS: 143 → 86 → 1470 → 913 → 1774 → 948 → 1509 → 1022 → 1750 → 130
18 + 57 1384 557 861 826 561 467 728 1620 = 7099

SSTF: 143 → 130 → 86 → 913 → 948 → 1022 → 1470 → 1509 → 1750 → 1774
18 + 13 44 827 35 74 48 39 241 24 = 1363

SCAN: 143 → 913 → 948 → 1022 → 1470 → 1509 → 1750 → 1774 → 130 → 86
18 730 35 74 48 39 241 24 1644 44 = 3025

C-SCAN: 143 → 913 → 948 → 1022 → 1470 → 1509 → 1750 → 1774 → 4999 → 0 → 86 → 130
18 730 35 74 48 39 241 24 3225 4999 86 44 = 9603

C-Look: 143 → 913 → 948 → 1022 → 1470 → 1509 → 1750 → 1774 → 86 → 130
18 730 35 74 48 39 241 24 1688 44 = 2981

Aufgabe 10

(i) Poisson: $p(k) = \frac{\lambda^k}{k!} e^{-\lambda}$

expected value $\lambda = n \cdot b$

$\rightarrow p(k, n, b) = \frac{1}{k!} n^k b^k e^{-nb}$

(ii) $p(1, n, b) + p(2, n, b) + \dots p(n, n, b)$

$= \sum_{i=0}^{n} p(i, n, b) - p(0, n, b)$

$= 1 - p(0, n, b)$

$p(0, n, b)$ is the probability that all objects experience the good outcome. The probability for each object is $(1-b) = g$

$\rightarrow p(0, n, b) = g^n$

$\rightarrow p(1, n, b) + p(2, n, b) + \dots p(n, n, b) = 1 - g^n$

Aufgabe 11

Designating disc 1-4 and probability of failure $p_1 = p_2 = p_3 = p_4 = 0.04$

There are 6 possible combinations for two failures. With all probabilities equal, the possibility for two failures is

$p_{total} = p_{12} + p_{23} + p_{34} + p_{13} + p_{24} + p_{14} = 6 \cdot p_{ab}$

$= 6 \cdot p_a \cdot p_b = 6 \cdot 0.04 \cdot 0.04 = 0.0096 \approx 0.1\%$

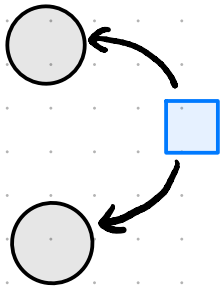
Aufgabe 12

RAID 1 saves the complete files in both available disks

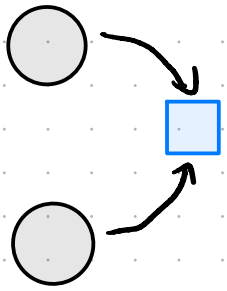
RAID 5 uses three of the four available discs to save parts of each incoming file in a way that all information is saved. The fourth disc is used to save a new file with the information necessary to retrieve the full file. The designated fourth disc changes with each incoming file.

Aufgabe 13

"Illegal" graphs are those, where a single piece of a mutually exclusive resource is attached to multiple processes

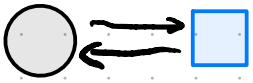


✗ Not allowed



✓ Allowed

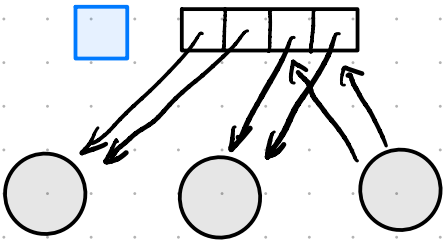
Also, a process cannot be waiting for a resource, while at the same time, that resource is attached to the process



✗ Not allowed

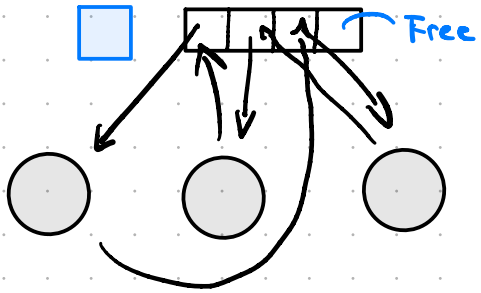
Aufgabe 14

a)



Case 1: Resource is attached to two processes

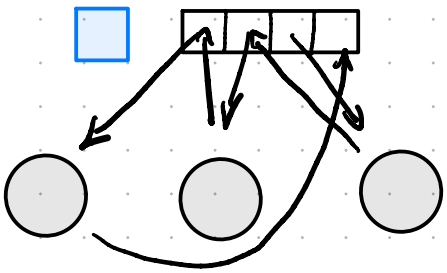
→ Only need to wait for (one of) the processes to finish



Case 2: Chain of Resource requests

→ There will always be one free instance of the resource left.

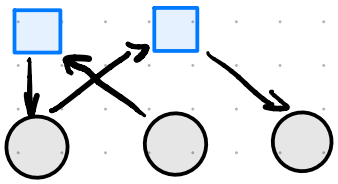
No possibility for this graph to happen



Case 3: Chain of requests with multiple requests per process

→ Once a process requests a free resource, that resource immediately becomes attached. Resulting back in Case 2.

b) 3 processes, 2 resources



→ A deadlock only occurs when a cycle is formed.