Computer Engineering – Submission due 25.10.2020

16. Oktober 2025

1 Data Encodings

1.1 Remote Access

Tasks

1. Run the three commands `whoami`, `hostname`, and `lscpu` on one of the systems.

   whoami - Benutzerinformationen

   Person 1:

   ```
   ce42@elaine00:~$ whoami
   ce42
   ```

   Person 2:

   ```
   ce43@elaine00:~$ whoami
   ce43
   ```

   hostname - Rechnername

   ```
   ce42@elaine00:~$ hostname
   elaine00
   ```

   lscpu - CPU-Informationen

   ```
   ce42@elaine00:~$ (lscpu | head -n 5; echo "---"; lscpu | tail -n 5)
   Architecture:                         x86_64
   CPU op-mode(s):                       32-bit, 64-bit
   Address sizes:                        48 bits physical, 48 bits virtual
   Byte Order:                           Little Endian
   CPU(s):                               24
   ---
   Vulnerability Spec store bypass:      Mitigation; Speculative Store Bypass disabled via prctl
   Vulnerability Spectre v1:             Mitigation; usercopy/swapgs barriers and __user pointer
   Vulnerability Spectre v2:             Mitigation; Enhanced / Automatic IBRS; IBPB conditional
   Vulnerability Srbds:                  Not affected
   Vulnerability Tsx async abort:        Not affected
   ```

2. Print the contents of the file `welcome.txt` in your home directory.

   Inhalt von welcome.txt:

   ```
   Welcome to the Computer Engineering class! My waveform told a long story and
   ended with 'It works.'
   ```

1.2 C/C++ Data Types

Tasks

1. Write C/C++ code to print the binary representations of the following variables.

   `(view ./src)`

2. For each of the variables, briefly explain why you obtained the resulting bits! (Siehe Erklärungen oben)

   Binäre Repräsentationen:

   1: 1: `00000001` – Dezimalwert 1.

   2: 255: `11111111` – Maximalwert für 8-bit unsigned char ($2^8 - 1$).

   3: 255 + 1: `00000000` – Integer Overflow: 9. Bit wird abgeschnitten.

   4: 0xA1: `10100001` – Hexadezimal: A = 1010, 1 = 0001.

   5: 0b1001011: `01001011` – Binärliteral auf 8 Bits aufgefüllt.

   6: 'H': `01001000` – ASCII-Wert 72.

   7: -4: `11111100` – Zweierkomplement: +4 invertieren und +1.

   8: 1u « 11: `00000000000000000000100000000000` – Linksshift der 1 (unsigned integer) um 11 Positionen (2^{11}).

   9: l_data8 « 21: `00000000000000000000000000000000` – Overflow: Bit auf Position 32.

   10: 0xFFFFFFFF » 5: `00000111111111111111111111111111` – Hexadezimalzahl - alle Bits sind 1. Rechtsshift um 5, linke Bits werden 0.

   11: 0b1001 ˆ 0b01111: `00000000000000000000000000000110` – XOR: 01001 ˆ 01111 = 00110.

   12: ˜0b1001: `11111111111111111111111111110110` – NOT invertiert alle 32 Bits.

   13: 0xF0 & 0b1010101: `00000000000000000000000001010000` – AND: 11110000 & 01010101 = 01010000.

   14: 0b001 | 0b101: `00000000000000000000000000000101` – OR: 001 | 101 = 101.

   15: 7743: `00000000000000000001111000111111` – Dezimal zu Binär.

   16: -7743: `11111111111111111110000111000001` – Zweierkomplement: invertieren und +1.