Systemsicherheit - 2. Übung

Dennis Rotärmel, Niklas Entschladen, Tobias Ratajczyk, Gruppe Q

May 2, 2019

1 Aufgabe 1

a)

Folgender Assembly-code führt die Berechnung aus:

```
1         imul ecx, ecx ;ecx*ecx
2         sub ecx, ebx ;ecx*ecx-ebx
3         shl eax, 1 ;eax*2
4         add eax, ecx ;ecx*ecx-ebx+eax*2
5         add ebx, 0xaaaa ;ebx+0xaaaa
6         xor eax, ebx ;(ecx*ecx-ebx+eax*2)^(ebx+0xaaaa)
```

Die Ausgabe des Programms lautet: `12345`.

2 Aufgabe 2

a)

Bei der Ausführung des Programmes wird der jeweils i-te Buchstabe um den Wert i-1 erhöht (siehe ASCII-Tabelle). Danach wird der daraus entstehende String mit folgendem String vergleichen: "HPFRV". Daraus lässt sich schlussfolgern, dass das Schlüsselwort "HODOR" lautet. Die Eingabe des Wortes bestätigt dies. Bis auf den Befehl "break verif_key" und die dazugehörigen step- und continue-Anweisungen wurden keine weiteren Befehle benötigt.

b)

```
1  #include <stdio.h>
2
3  int verify_key(char *str){
4          char key[5] = "HPFRV";
5
6          for(int i=0; i<5; i++){
7                  if (str[i]!=key[i]){
8                          printf("Key is not valid :(\n");
9                          return 0;
10                 }
11         }
12         printf("Key is valid! Whoop whoop :)\n");
13         return 0;
14 }
15
16 int main(){
17         char str[5];
18         printf("Enter serial (5 capital letters): ");
19         scanf("%s", str);
20
21         for(int i=0; i<5; i++){
22                 str[i]=str[i]+i;
23         }
24
25         return verify_key(str);
26 }
```

3 Aufgabe 3

a)

- **Data Movement**:
- **Arithmetic and Logic**:
  - `xor eax, eax` $\hat{=}$ eax $\oplus$ eax $= 0$
  - `add eax, 1234h` $\hat{=}$ eax $+ 4660 = 0 + 4660 = 4660$
  - `ror eax, 16` $\hat{=} 0001001000110100_2 \rightarrow 0011010000010010_2 \hat{=} 13330_{10}$
  - `or eax, 55h` $\hat{=} 0011010000010010_2 \vee 0000000001010101_2 = 0011010001010111_2 \hat{=} 13399_{10}$
  - `inc eax` $\hat{=} =$ eax $+ 1 = 13400$
  - `shl ax, 8` $\hat{=} 0011010001011000_2 \rightarrow 0011010001011000_2$ (ax $= 0$, somit keine Änderung)
  - `mov al, 78h` $\hat{=} 0011010001011000_2 \rightarrow 0011010001110100_2 \hat{=} 13428_{10}$
  - Damit ist am Ende der Wert 13428 im `eax` Register
- **Control Flow**:
  - `mov eax, 1h` $\hat{=}$ eax $= 00\ldots0001$
  - `neg eax` $\hat{=}$ eax $= 11\ldots1111$
  - `mov ebx, FFFFFFF8h`
  - `cmp eax, ebx`
  - `jg true` $\rightarrow$ ebx ist größer als eax (da Vorzeichen beachtet)
  - `mov eax, 0` wird ausgeführt
  - damit ist im eax der Wert 0 am Ende.

b)

`ja` ist ein unsigned-Vergleich, damit wird beim vergleichen das Vorzeichen nicht beachtet. In diesem Fall ist der Wert im `eax` größer und es wird `mov eax, 1` ausgeführt.