Hochschule Augsburg
Fakultät für Informatik

Wintersemester 2017/2018

Prüfung

Prüfungsfach:	Systemnahe Programmierung
Datum/Uhrzeit:	1. Februar 2018 / 12:30 Uhr
Raum:	M1.01, J2.18
Prüfer:	Dr. Hubert Högl
Dauer:	**60** Minuten
Hilfsmittel:	keine

Hinweise:

1. Dieses Angabenblatt hat auch eine **Rückseite**! Bitte sofort überprüfen.
2. **Schreiben** Sie bitte **nicht** auf das **Angabenblatt**. Verwenden Sie für Ihre Antworten die separat ausgeteilten Bögen. **Die Angaben dürfen Sie behalten.**
3. Schreiben Sie **nicht mit Bleistift**.

Viel Glück!

Aufgabe 1 ("Warmup", 12 Punkte)

(a) Nennen Sie zwei relevante Anwendungsgebiete von systemnaher Programmierung in Assembler.

(b) Wie gross wird in etwa der Wert von `eax` sein, wenn Sie das folgende Programmstück im Debugger eine Sekunde laufen lassen und dann abbrechen? (2 Punkte)

```
    movl $0, %eax
L1:
    incl %eax
    jmp  L1
```

(c) Erklären Sie den Unterschied zwischen den beiden folgenden Zeilen (2 Punkte)

```
movl 10, %ebx
movl $10, %ebx
```

(d) Was bewirkt der folgende Befehl?

```
movb (%eax), (%ebx)
```

(e) Wie viele Bytes umfassen x86 Maschinenbefehle (CISC) im Vergleich zum ARM Prozessor (RISC)?

(f) Was passiert, wenn Sie bei einem Programm das finale `exit` weglassen?

Aufgabe 2 (8 Punkte)

In **welche Abschnitte** ist jedes Programm im Speicher gegliedert und wozu dienen diese Abschnitte? Zeichnen Sie ein **Bild des gesamten Linux Programmes** beim Start.

Aufgabe 3 (4 Punkte)

Erläutern Sie, warum **unterschiedliche Prozesse** auf Ihrem Rechner immer an den **gleichen Adressen** liegen. Wenn man sich die **physikalischen Adressen** ansieht, dann kann eine Speicherzelle im Speicher nur von einem Prozess genutzt werden, unterschiedliche Prozesse müssen also an verschiedenen physikalischen Adressen liegen. Wie funktioniert das?

Aufgabe 4 (4 Punkte)

Beantworten Sie bitte folgende Fragen zum Stack:

1. Wie heissen die Assemblerbefehle um auf den Stack zuzugreifen?
2. Wie implementieren Sie diese Befehle aus elementaren `mov`, `sub` und `add` Befehlen?

Aufgabe 5 (5 Punkte)

Der folgende Systemaufruf `ptrace` wird von Debuggern verwendet, um Prozesse zu debuggen. Er hat die Nummer 26. Schreiben Sie in Assembler einen beispielhaften Systemaufruf von `ptrace` hin. Die Schreibweise `void *vp` bezeichnet einen Zeiger `vp` auf einen beliebigen Typ. Das vierte Argument wird in `esi` übergeben.

```
long ptrace(int request, int pid, void *addr, void *data);
```

Aufgabe 6 (4 Punkte)

Schreiben Sie in Assembler eine Funktion „string copy“

```
int strcpy(char *s1, char *s2)
```

die einen Null-terminierten String `s1` an die Adresse `s2` kopiert. Die Schreibweise `char *s1` bedeutet, dass `s1` ein Zeiger auf Character ist. Schreiben Sie die Funktion vollständig mit Prolog und Epilog. Der Rückgabewert soll immer Null sein.

Aufgabe 7 (6 Punkte)

(a) Was macht die folgende C Funktion?

```
int pot(int x, int n)
{
    if (n == 1) {
        return x;
    }
    else {
        return x * pot(x, n-1);
    }
}
```

(b) (1 Punkt) Wie nennt man die verwendete Programmiertechnik?

(c) (5 Punkte) Zeichnen Sie für den Aufruf `pot(4, 2)` ein Stackdiagramm, das alle Stackframes enthält bis zur maximalen Aufruftiefe (denken Sie auch an die Verlinkung der Frames!).

Aufgabe 8 (13 Punkte)

Hier geht es um Exploits

Punkte: (a) 4, (b) 4, (c) 2, (d) 1, (e) 2

(a) Was bedeuten die Begriffe **Opfer**, **Einbrecher**, **Injektionscode** und **Shellcode**?

(b) Erläutern Sie die einzelnen nötigen Schritte, damit ein Einbrecher in dem Beispiel aus der Vorlesung/Übung zu einer Shell mit Admin-Rechten kommen kann.

(c) Warum ist der folgende naive Shellcode falsch?

```
naive_shellcode:
        mov  ebx, <filename_addr> ; filename
        mov  ecx, <argv_addr>     ; 0x804971c   ; argv
        mov  edx, 0x0             ; envp
        mov  eax, 0xb             ; code for execve system call
        int  0x80
```

(d) Wie welchem GDB-Kommando kann man eine Folge von Maschinencode-Bytes wieder in Assembleranweisungen verwandeln?

(e) Nennen Sie **zwei** Massnahmen, die diese Art Einbrüche heutzutage verhindern.

Ende der Prüfung