Hochschule Augsburg
Fakultät für Informatik

Wintersemester 2015/2016

Prüfung

Prüfungsfach:	Systemnahe Programmierung
Datum/Uhrzeit:	28. Januar 2016 / 8:30 Uhr
Raum:	wie angekündigt
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 (6 Punkte)

Betrachten Sie folgenden Assembler-Befehl:

```
movl    data_items(, %edi, 4), %eax
```

(a) Nennen Sie die einzelnen Bestandteile der verwendeten Adressierungsart.

(b) Welche Adresse wird mit welcher Breite angesprochen, wenn man `data_items` mit `0xa3cd` und `edi` mit 4 annimmt?

(c) Um welchen weiteren Freiheitsgrad könnte man diese Adressierungsart erweitern (Tipp: vor dem Komma)?

Punkte: (a) 3, (b) 2, (c) 1

Aufgabe 2 (4 Punkte)

Fragen zur Endianness

(a) Was bedeutet die „Endianness“ eines Rechners?

(b) Wie kann man herausfinden, welche Endianness ein Rechner hat? Beschreiben Sie das verwendete Prinzip und geben Sie auch die nötigen Assembler-Befehle an.

Punkte: (a) 2, (b) 2

Aufgabe 3 (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 4 (4 Punkte)

Was machen folgende GDB Kommandos?

(a) `(gdb) list main`

(b) `(gdb) br 15`

(c) `(gdb) p/x $edi`

(d) `(gdb) x/4xw $esp+8`

Aufgabe 5 (5 Punkte)

Das Manual zum `execve()` Systemaufruf (`man 2 execve`) beschreibt den Aufruf wie folgt:

```
int execve(char *filename, char *argv[], char *envp[]);
```

Der Aufruf hat die Nummer 11.

(a) Der Systemaufruf wird in C als Funktion aufgerufen, in Wirklichkeit ist es aber ein Software Interrupt. Wie passt das zusammen?

(b) Beschreiben Sie den tatsächlichen Systemaufruf in Assembler. Die Schreibweise `char *ar[]` bezeichnet ein Array `ar` im Speicher, das als Elemente Zeiger auf Strings beinhaltet. Der erste Parameter `filename` ist ein Zeiger auf einen String.

Punkte: (a) 2, (b) 3

Aufgabe 6 (13 Punkte)

Sehen Sie sich folgenden Assembler-Quelltext an. Die drei Punkte ... stehen für beliebigen Code, der uns nicht interessiert. Beantworten Sie bitte folgende Fragen:

(a) Beschreiben Sie, was an den Stellen (1) bis (7) gemacht wird. Bitte keine trivialen Bemerkungen, sondern die übergeordnete Absicht des Codes deutlich machen.

(b) Zeichnen Sie den Stack direkt nach der Ausführung von Zeile 5. Zeichnen Sie auch Framepointer und Stackpointer ein.

(c) Wie greift man innerhalb der Funktion `tuwas()` auf die lokalen Daten zu? Nehmen Sie an, dass die 8 Byte aus zwei Integer (long) Werten bestehen. Schreiben Sie die Framepointer-relative Adressierung für den Integer mit der kleineren Adresse hin.

```
    pushl $2              # (1)
    pushl $4              # (1)
    call tuwas            # (2)
    addl $8, %esp         # (3)
    ...

tuwas:
    pushl %ebp            # (4)
    movl %esp, %ebp       # (4)
    subl %esp, 8          # (5)
    ...
    movl %ebp, %esp       # (6)
    popl %ebp             # (6)
    ret                   # (7)
```

Punkte: (a) 7, (b) 4, (c) 2

Aufgabe 7 (5 Punkte)

Diese Frage dreht sich um das dynamische Linken von Objektdateien. Ihr Modul `main.o` soll aus der C Standardbibliothek die Funktion `printf()` zur Laufzeit verwenden.

(a) Welche Komponente muss die ausführbare Datei enthalten, damit das dynamische Linken möglich wird?

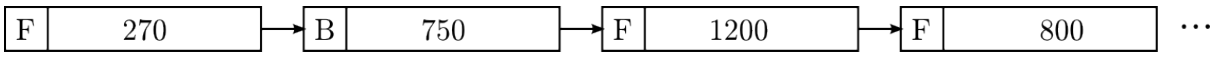
(b) Beschreiben Sie die einzelnen Schritte die beim dynamischen Linken gemacht werden. Verwenden Sie eine Skizze, so wie wir das in der Vorlesung gemacht haben.

(c) Schreiben Sie die notwendigen Kommandozeilen für den GNU C Compiler hin.

Punkte: (a) 1, (b) 2, (c) 2

Aufgabe 8 (10 Punkte)

Die folgende Abbildung zeigt eine **Kette aus freien (F) und belegten (B) Blöcken auf dem Heap**. Die Zahl gibt die Grösse des Blockes an.



a) **Wo liegt der Heap-Speicher** im Speicher eines Prozesses?

b) **Welcher Block** wird bei einem `allocate(600)` Aufruf nach dem Algorithmus aus dem Buch von Bartlett belegt?

c) **Welche Nachteile** hat dieser einfache Algorithmus?

d) Wie kann der einfache Algorithmus **verbessert** werden?

e) Wie kann der gesamte, dem Heap zur Verfügung stehende Speicher **vergrössert** werden?

Aufgabe 9 (4 Punkte)

Zeitmessung in Assembler

(a) Mit welchem Maschinenbefehl kann man bei der x86 CPU sehr genau Zeiten bestimmen? (Wie heisst er und wie genau misst er?)

(b) Warum muss man die Messung mehrfach ausführen und das Minimum der einzelnen gemessenen Zeiten bestimmen?

Aufgabe 10 (13 Punkte)

Exploits

(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.

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