Hochschule Augsburg
Fakultät für Informatik

Wintersemester 2012/2013

Klausur

Prüfungsfach:	Systemnahe Programmierung
Datum/Uhrzeit:	4. Februar 2013 / 12:30 Uhr
Raum:	M1.02, W3.02
Prüfer:	Dr. Hubert Högl
Dauer:	**60** Minuten
Hilfsmittel:	keine

Hinweise:

1. Dieses Klausurangabenblatt 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 (2 Punkte)

Was versteht man unter robusten Programmen, so wie es im Kapitel 7 im Buch von Bartlett steht?

Aufgabe 2 (4 Punkte)

Wenn man ein Programm aufruft, dann wird es zunächst in den Speicher geladen. In welche Abschnitte ist das Programm im Speicher gegliedert und wozu dienen diese Abschnitte? Zeichnen Sie auch ein Bild des gesamten Linux Programmes beim Start (inklusive Programmname, Argumenten, Umgebungsvariablen und Heap).

Aufgabe 3 (4 Punkte)

Schreiben Sie den Assembler-Code hin, um mit der Funktion `write()` aus der C Standardbibliothek einen String Ihrer Wahl auf die Standardausgabe auszugeben. Hier sind die Aufrufparameter von `write`:

```
write(int fd, char *buf, int count);
```

Aufgabe 4 (5 Punkte)

Punkte: a) 2, b) 3

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.

Aufgabe 5 (12 Punkte)

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

Der folgende Auszug aus dem Hauptspeicher beginnt bei der konstanten Adresse `mem`. Nach oben hin steigen die Adressen an. In jeder Zeile sind 8 Byte enthalten. Rechts neben der Tabelle steht die ASCII Darstellung der Speichertabelle.

```
mem+32:   00 00 00 00 00 00 00 00
mem+24:   00 00 00 00 00 00 00 00
mem+16:   00 00 00 00 00 00 00 00
mem+8:    00 00 00 00 00 00 00 00
mem:      41 42 43 00 97 a0 bf 19  ABC.....
          -----------------------
          0  1  2  3  4  5  6  7
```

a) Führen Sie den folgenden Maschinenbefehl aus. Die Register sind wie folgt gesetzt: `eax = 32`, `ebx = 1`.

```
movb    $0x55,  mem(%eax, %ebx, 4)
```

b) Ab Adresse `mem` findet man den String `ABC`. Welche Bytes gehören zu diesem String?

c) Der String `ABC` ab Adresse `mem` soll an die Stelle `mem+8` kopiert werden. Schreiben Sie in Assembler eine Funktion `scopy(a1, a2)` mit zwei Parametern, die byteweise einen String von Adresse `a1` nach Adresse `a2` kopiert.

d) In welcher Weise könnte man den String noch im Speicher ablegen? Tragen Sie Ihren Vorschlag ab Adresse `mem+8` in den Speicher ein.

e) Ihr Programm schreibt mit einer `movl` Operation an die Adresse `mem+24` die Zahl `0x12345678` in den Speicher. Schreiben Sie die Zahl in die Speichertabelle.

f) Wie können Sie mit dem `gdb` Debugger die Speichertabelle ausgeben? Schreiben Sie das dazu nötige Kommando hin.

Aufgabe 6 (6 Punkte)

Sie erinnern sich noch an den Kurztest in diesem Semester. Füllen Sie die Funktion `timespow2()` aus.

```
#   Aufgabe: timespow(3, 2) + timespow(2, 3)

    .section .data
    .section .text
    .globl _start

_start:
    pushl $2        # b
    pushl $3        # x
    call  timespow2
    addl  $8, %esp
    pushl %eax
    pushl $3        # b
    pushl $2        # x
    call  timespow2
    addl  $8, %esp
    popl  %ebx
    addl %eax, %ebx
    movl $1, %eax
    int $0x80

# timespow2(x, b)
#   return x * 2^b
#   Trick: x * 2^b = shift argument x left by b bits
#                    shll %cl, %ebx  (shift ebx left by cl bits)
.type timespow2, @function
timespow2:
    ..................... # 1 Prolog
    .....................# 2 Prolog
    ..................... # 3 Argument holen
    ..................... # 4 Argument holen
    ..................... # 5 Schieben
    ..................... # 6 Ergebnis ablegen
    ..................... # 7 Epilog
    ..................... # 8 Epilog
    ..................... # 9 Zurueckkehren
```

Aufgabe 7 (4 Punkte)

Wie können Sie ein Laufzeitprofil eines Programmes erstellen? Beschreiben Sie kurz die einzelnen dazu notwendigen Schritte.

Aufgabe 8 (8 Punkte)

Punkte: a) 2, b) 2, c) 2, d) 2

Im Buch von Bartlett wird im Kapitel 9 die Funktionsweise des Heap erklärt. Beantworten Sie dazu folgende Fragen:

a) Wie heissen die wesentlichen Funktionsaufrufe zur Verwendung des Heap?

b) Was verstehen Sie unter *Unmapped Memory*?

c) Wie kann der Heap-Speicherbereich vergrössert werden?

d) In welchen Datenstrukturen werden die vom Heap angeforderten Speicherblöcke verwaltet? Zeichnen Sie ein Diagramm zur Erläuterung.

Aufgabe 9 (6 Punkte)

Hier geht es um Einbruchtechniken.

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

a) Beschreiben Sie in verständlichen Sätzen und Skizzen, wie **Opfer**, **Einbrecher**, **Injektionscode** bzw. **Shellcode** zusammenhängen.

b) Warum ist der folgende naive Shellcode falsch? Warum ist es schwierig, Shellcode zu schreiben?

```
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
```

c) Was kann man gegen diese Art des Einbruchs wirkungsvoll unternehmen?

Ende der Klausur