Klausur Betriebssysteme und Sicherheit, 01.03.2024

— Bearbeitungszeit: 90 Minuten — Prüfer: Prof. Dr. Schirmeier, Prof. Dr. Tschorsch —

1	2	3	4	5	6	$\sum$
15	15	16	15	15	14	90

Alle Aussagen sind so ausführlich wie nötig, aber so knapp wie möglich zu begründen.

Aufgabe 1 – Kryptografie

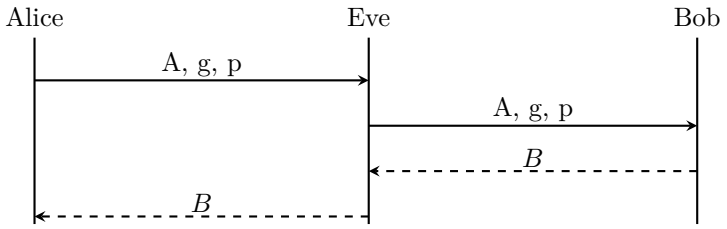
15 Punkte

In der folgenden Abbildung versucht Eve einen Monster-in-the-Middle-(MitM-)Angriff auf den Diffie-Hellmann-(DH-)Schlüsselaustausch von Alice und Bob durchzuführen.

Gehen Sie von der Notation aus der Vorlesung aus. Zur Erinnerung:

- a ist eine geheime Zufallszahl von Alice
- b ist eine geheime Zufallszahl von Bob
- g ist der sog. Primzahlgenerator
- p ist eine Primzahl
- $A = g^a \mod p$ und $B = g^b \mod p$

Hinweis: Sie dürfen Ihre Angaben auch direkt in der Abbildung ergänzen.



a) Was macht Eve falsch? Beschreiben und skizzieren Sie einen erfolgreichen MitM-Angriff. **4 P**

b) Wie kann man den DH-Schlüsselaustausch zuverlässig gegenüber Eve absichern? Nennen Sie hierzu die erforderlichen Schutzziele. **2 P**

c) Unter welchen Bedingungen ist ein ungesicherter DH-Schlüsselaustausch sicher? **2 P**

d) Welche datenschutzrelevanten Informationen kann Eve beobachten? Sie dürfen von einem sicheren DH-Schlüsselaustausch ausgehen. Begründen Sie Ihre Antwort. **2 P**

Gehen Sie von einer 2-Bit-Blockchiffre mit nebenstehend dargestellter Schlüsseltabelle aus.

In	Out
00	01
01	11
10	00
11	10

e) Verschlüsseln Sie den Klartext `01010000 01010011` mit der Betriebsart ECB (Electronic Codebook). **2 P**

f) Welches Problem ergibt sich in der Betriebsart ECB? Geben Sie basierend auf dem Klartext und Ihrem berechneten Geheimtext ein konkretes Beispiel für das Problem an. **2 P**

g) Wie könnte das Problem aus der vorherigen Teilaufgabe adressiert werden? **1 P**

Aufgabe 2 – Sicherheitslücken **15 Punkte**

Ein Unternehmen hat kürzlich eine kritische Sicherheitslücke in seiner Software identifiziert. Die Schwachstelle wurde von einem Geheimdienst am 01.01.2024 entdeckt. Das Softwareunternehmen hat am 06.01.2024 von der Sicherheitslücke erfahren und am 07.01.2024 sowohl Details zu der Sicherheitslücke als auch einen Patch veröffentlicht, welcher am 15.01.2024 eingespielt wurde.

a) Berechnen Sie das Fenster der Verwundbarkeit (*Window of Vulnerability*). **1 P**

b) Beschreiben Sie den Unterschied zwischen einem *Zero-Day Exploit* und einem *One-Day Exploit* in wenigen Worten. Konstruieren Sie ein Beispiel-Datum für den jeweiligen Exploit. **4 P**

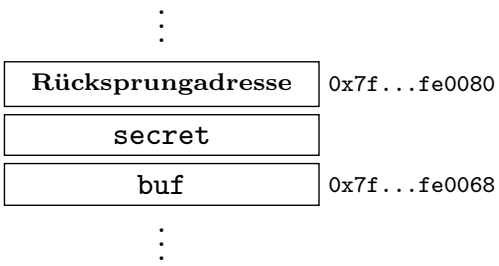
c) Beschreiben Sie die Konsequenzen in Bezug auf das *Window of Vulnerability* bei einer Offenlegung von Sicherheitslücken nach dem Prinzip der *Responsible Disclosure* im Vergleich zur *Full Disclosure*. **2 P**

Gegeben sei folgender C-Codeausschnitt.

Die Funktion `password_check` vergleicht eine Eingabe (`buf`) mit dem Passwort vom System (`secret`). Hierbei lädt die Funktion `get_secret` das Passwort vom System. Die Funktion `strcmp` vergleicht zwei Strings und gibt `0` zurück, wenn die beiden Strings gleich sind.

Die Abbildung unten rechts zeigt den Stack unmittelbar vor dem `return` (Zeile 11) am Ende der Funktion `password_check`.

```
1  void get_secret(char*);
2  void unlock(void);
3
4  void password_check() {
5     char secret[8];
6     char buf[8];
7     get_secret(secret);
8     gets(buf);
9     if (strcmp(buf, secret) == 0)
10       unlock();
11    return;
12 }
```



d) Beschreiben Sie, warum die Verwendung von `gets` ein Sicherheitsrisiko darstellt. **2 P**

e) Können Sie eine Eingabe konstruieren, so dass `strcmp` den Wert 0 zurückliefert, ohne dass Sie das Passwort kennen? Falls ja, beschreiben Sie das allgemeine Vorgehen. Falls nein, begründen Sie Ihre Antwort.

4 P

f) Machen Sie einen Vorschlag, wie man die Funktion `password_check` modifizieren könnte, um die Schwachstelle zu beseitigen. Es ist ausreichend Ihren Ansatz kurz in Worten zu beschreiben. **2 P**

Aufgabe 3 – Ein- und Ausgabe **16 Punkte**

a) Festplatten und SSDs werden als blockorientierte Geräte bezeichnet. Benennen Sie eine andere Geräteklasse und geben Sie ein dafür beispielhaftes Gerät an! **2 P**

b) Erläutern Sie, auf welche Weise *Direct Memory Access* (DMA) die CPU bei der Ausführung von I/O-Operationen entlastet! **2 P**

Für den Rest dieser Aufgabe soll eine SSD mit einer Blockgröße von 4 KiB betrachtet werden. Auf der betrachteten SSD befindet sich das unten skizzierte Dateisystem. Das Dateisystem nutzt eine Blockgröße von 4 KiB. Die dargestellten Blöcke enthalten entweder Inodes (IB), Superblöcke (SB), Verzeichnisblöcke (VB) oder eine Allokations-Bitmap für Blöcke (BA) bzw. Inodes (IA). Weitere Blöcke (z.B. für Dateiinhalte) existieren, sind aber nicht dargestellt. Einträge in den Inodeblöcken haben dabei die Form:

`<Inode-Nummer>` → `<Liste der Datenblöcke> /<Dateigröße>`

Block-ID	0	1	2	3	4	5
Typ	SB	BA	IA	IB	VB	VB

Inhalt	free: 90
	root: `I0`
		0–9	0,1,
			2,3
				`I0` → `4 / 32`
				`I1` → `5 / 32`
				`I2` → `6,7,8,9 / 14000`
					`users` → `I1`	`anthony` → `I2`

c) Welche Blöcke des Dateisystems müssen in welcher Reihenfolge gelesen werden, wenn aus der Datei `/users/anthony` 8 KiB ab einem Offset von 4 KiB eingelesen werden sollen? Nehmen Sie dabei an, dass die Inhalte eines Blockes immer vollständig gelesen werden (also kein Block ein zweites Mal gelesen werden muss). **6 P**

d) Im Folgenden werden 3000 Bytes Daten an die Datei `/users/anthony` angehangen. Auf wie viele Datenblöcke erstreckt sich die vergrößerte Datei? Wie viele Bytes werden beim Speichern der neu angehängten Daten auf die SSD geschrieben (ohne Updates der Metadaten)? **2 P**

e) Zur Vermeidung von Inkonsistenzen nach einem Systemabsturz nutzt das Dateisystem das Metadata-Journaling-Verfahren. Welche Blöcke werden somit im Rahmen der in Teilaufgabe d) beschriebenen Append-Operation in das Journal des Dateisystems geschrieben? Gehen Sie dabei davon aus, dass zum Vornehmen der Änderungen keine Metadatenblöcke allokiert werden müssen. **4 P**

Aufgabe 4 – Virtueller Speicher **15 Punkte**

Wir betrachten ein 32-Bit-Betriebssystem, das zur Verwaltung des virtuellen Adressraums 2-stufige Seitentabellen nutzt. Ein Seite ist 4 KiB groß. Ein Seitentabelle umfasst 2^{10} Einträge, und jeder Eintrag benötigt 4 Bytes.

a) In diesem System soll ein schreibender Speicherzugriff auf die virtuelle Adresse `0x2703DEF` durchgeführt werden. Skizzieren Sie in den beiden oberen der vorgezeichneten Tabellen zwei unterschiedliche Szenarien, in denen die MMU bei diesem Zugriff einen Seitenfehler auslöst. Skizzieren Sie in der verbleibenden vorgezeichneten Tabelle einen Zustand, bei dem der Speicherzugriff erfolgreich ist und auf die physische Adresse `0xABCDEF` abbildet. Die Seitentabelle der ersten Stufe liegt an der Adresse `0xE85FD000`. Es sind jeweils **alle** grau hinterlegten Felder so auszufüllen, dass aus den skizzierten Tabellenausschnitten ersichtlich ist, dass die MMU ein Seitenfehler auslöst, bzw. der Zugriff erfolgreich ist!

Falls Sie ihre Antwort korrigieren möchten, können Sie die Ersatzdiagramme auf der letzten Seite nutzen. **Streichen Sie in diesem Fall ungültige Lösungen deutlich durch!** **8 P**

Zeichen	Bedeutung	Zulässige Werte
w	*Writable Bit*	`0` oder `1`
p	*Present Bit*	`0` oder `1`

Zugriffe mit Seitenfehler:

Adresse: `0xE85FD000`

Index	phys. Adresse	w	p
...			
...			

Adresse:

Index	phys. Adresse	w	p
...			
...			

Adresse: `0xE85FD000`

Index	phys. Adresse	w	p
...			
		`1`	`1`
...			

Adresse:

Index	phys. Adresse	w	p
...			
...			

Erfolgreicher Zugriff:

Adresse: `0xE85FD000`

Index	phys. Adresse	w	p
...			
...			

Adresse:

Index	phys. Adresse	w	p
...			
...			

b) Nun soll eine Seitentabelle für ein 64-Bit-Betriebssystem entwickelt werden. Erläutern Sie rechnerisch das Problem, dass in einer 2-stufigen Seitentabelle in Bezug auf den Speicherplatzbedarf einer Seitentabelle der ersten Stufe besteht, wenn eine Tabelle der zweiten Stufe weiterhin 2^{10} Einträge umfassen soll und die Seitengröße von 4 KiB beibehalten wird. Zudem soll das Betriebssystem auf einem Computer mit 32 GiB ($= 2^{35}$ Bytes) Hauptspeicher laufen. Beachten Sie, dass in einem 64-Bit-System ein Eintrag in der Seitentabelle 8 Bytes umfasst.

HINWEIS: 2er-Potenzen müssen **nicht** aufgelöst/ausmultipliziert werden.

3 P

c) Dieses Problem kann mit invertierten Seitentabellen umgangen werden. Nennen Sie einen Vor- und einen Nachteil von invertierten Seitentabellen gegenüber regulären Seitentabellen. **2 P**

d) In modernen Systemen sind 4- oder mehrstufige Seitentabellen zur Adressübersetzung üblich. Welche Technologie bzw. Hardware-Einheit wird in diesen Systemen genutzt, um implizite Speicherzugriffe bei der Adressübersetzung zu vermeiden? Erklären Sie auch kurz deren Funktionsweise. **2 P**

Aufgabe 5 – Unix

15 Punkte

a) Nennen Sie zwei Beispiele für Daten, die im Prozesskontrollblock gespeichert werden, und erklären Sie, wofür diese Daten benötigt werden. **2 P**

Gegeben sei nebenstehendes Programm. Gehen Sie im Folgenden von einer fehlerfreien Abarbeitung aus.

Bedeutung der genutzten Flags:

- `O_CREAT`: Wenn die angegebene Datei nicht existiert, wird eine neue Datei erzeugt.
- `O_WRONLY`: Die angegebene Datei wird ausschließlich zum Schreiben geöffnet.

```
1  int main() {
2      int fd1 = open("datei1.txt", O_CREAT|O_WRONLY);
3
4      int pid = fork();
5
6      int fd2 = open("datei2.txt", O_CREAT|O_WRONLY);
7
8      if (pid > 0) {
9          write(fd1, "c", 1);
10         wait(NULL);
11         write(fd2, "c", 1);
12     } else {
13         write(fd1, "a", 1);
14         write(fd2, "a", 1);
15     }
16
17     write(fd1, "b", 1);
18     write(fd2, "b", 1);
19
20     return 0;
21 }
```

b) Was steht nach Beendigung des Programms in der Datei `datei1.txt`? Sofern es mehrere Varianten geben kann, nennen Sie diese. **3 P**

c) Nennen und **begründen** Sie, was nach Beendigung des Programms in der Datei `datei2.txt` steht. **2 P**

Die nebenstehenden Shell-Befehle geben alle Einträge in einem Verzeichnis aus, die das Wort „klausur“ enthalten. Dabei wird die temporäre Datei `tmp` genutzt.

```
1 ls > tmp
2 grep "klausur" < tmp
```

d) Welche Voraussetzung muss in diesem Beispiel `ls` bzw. `grep` erfüllen, um die Datei `tmp` nutzen zu können? Geben Sie außerdem einen äquivalenten Shell-Befehl an, der keine temporäre Datei verwendet. **2 P**

In einem Unix-System existieren die folgenden Dateien:

Dateiname	Besitzer	Gruppe	Rechte
`simulator`	`sim`	`sim`	`r-s --- r-x`
`data`			

Die beiden Nutzer `armin` und `berta` sollen die Datei `data` bearbeiten, also lesen und schreiben können. Darüber hinaus soll keinerlei Zugriff auf die Datei möglich sein.

e) Welche Rechte müssen der Datei `data` zugewiesen werden? Füllen Sie die grau hinterlegten Felder in der Tabelle entsprechend aus. Sollten für die Umsetzung der beschriebenen Zugriffsrechte noch weitere Voraussetzungen nötig sein, beschreiben Sie diese kurz. **3 P**

f) Während seiner Ausführung liest das Programm `simulator` die Datei `data`. Nutzer `armin` versucht die Datei `simulator` nun auszuführen. Skizzieren Sie kurz, was geschieht. **3 P**

Aufgabe 6 – Synchronisation **14 Punkte**

a) Die drei Funktionen des folgenden Programms werden als Threads ausgeführt. Alle drei Threads sind zur selben Zeit laufbereit. Sorgen Sie durch geeignete Synchronisation der Abläufe dafür, dass das Programm die Zahlen von 1–6 in aufsteigender Reihenfolge (`123456`) ausgibt. Zu diesem Zweck stehen Ihnen drei Semaphore zur Verfügung. Diese gilt es geeignet zu initialisieren und anschließend an den grau hinterlegten Stellen im Programm die nötigen Semaphor-Operationen einzufügen.

HINWEIS: Es müssen **nicht** alle grauen Felder ausgefüllt werden.

Falls Sie ihre Antwort korrigieren möchten, können Sie die Ersatzdiagramme auf der letzten Seite nutzen. **Streichen Sie in diesem Fall ungültige Lösungen deutlich durch!** **5 P**

```
Initialwerte der Semaphore: S1:      S2:      S3:
```

```
f1() {
    puts("2");
    puts("3");
}
```

```
f2() {
    puts("1");
    puts("6");
}
```

```
f3() {
    puts("4");
    puts("5");
}
```

b) In dem unten abgebildeten Szenario ist eine Implementierung des Erzeuger-Verbraucher-Problems in C-ähnlichem Pseudocode dargestellt. Beschreiben Sie, welches Synchronisationsproblem bei der hier gezeigten Implementierung vorliegt, und in welchem Fall es eintreten kann. Korrigieren Sie das Programm, so dass das Problem nicht mehr auftreten kann, und erläutern Sie, warum der Fehler dadurch vermieden wird.

HINWEIS: Nehmen Sie zur Vereinfachung für dieses Szenario an, dass die Queue unendlich viele Elemente aufnehmen kann.

Falls Sie ihre Antwort korrigieren möchten, können Sie die Ersatzdiagramme auf der letzten Seite nutzen. **Streichen Sie in diesem Fall ungültige Lösungen deutlich durch!** **5 P**

```
1  // gemeinsamer Speicher
2  Semaphore mutex = 1;
3  Semaphore not_empty = 0;
4  struct list *queue;
5  struct element e;
```

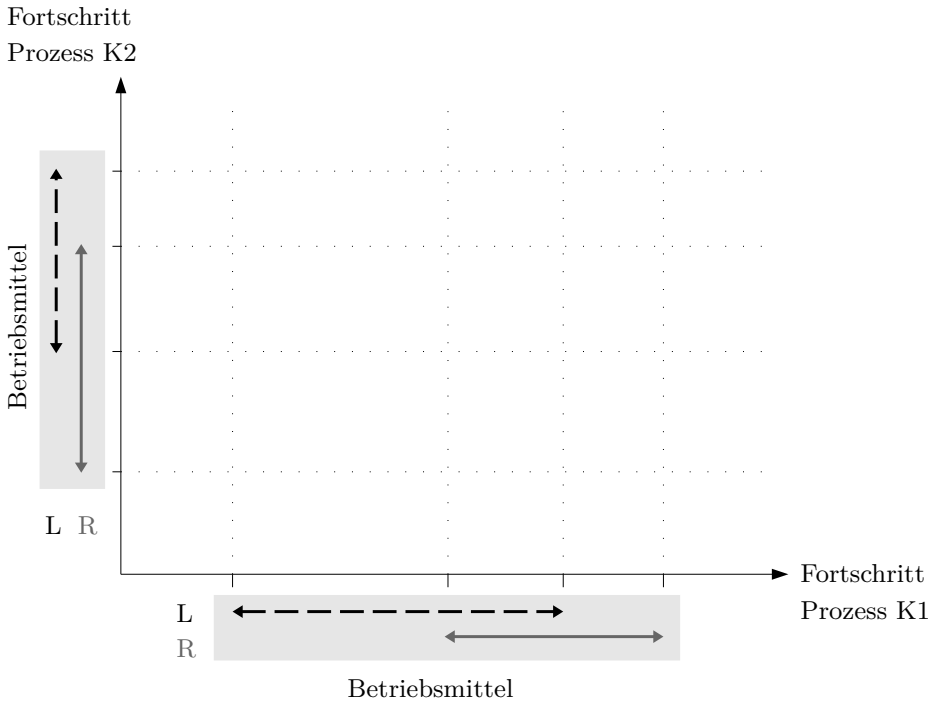
```
1  // Erzeuger
2  while(true) {
3      wait(&mutex);
4
5      enqueue(queue, e);
6
7      signal(&not_empty);
8
9      signal(&mutex);
10 }
```

```
1  // Verbraucher
2  while(true) {
3      wait(&mutex);
4
5      wait(&not_empty);
6
7      e = dequeue(queue);
8
9      signal(&mutex);
10 }
```

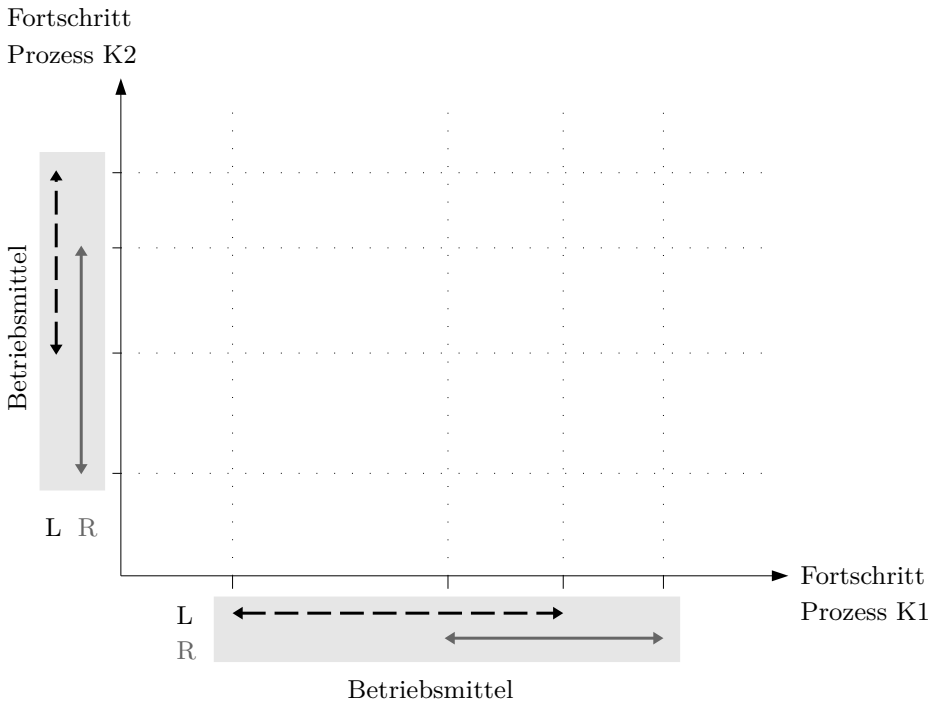
c) Das unten abgebildete Diagramm soll den Fortschritt der Prozesse K1 und K2 darstellen, die beide im Laufe ihrer Ausführung die unteilbaren Ressourcen L und R belegen und wieder freigeben.

- Zeichnen Sie den Bereich (A) ein, nach dessen Betreten eine Verklemmung unvermeidbar wird, und kennzeichnen Sie den Moment (X), in dem diese dann auch eintritt.
- Markieren Sie den Bereich (B), der von den Prozessen nicht betreten werden kann.
- Zeichnen Sie **einen** möglichen Ablauf (M) ein, in dem keine Verklemmung auftritt, und beide Prozesse zu Ende laufen.

4 P



<u>Ersatzdiagramm</u> (Streichen Sie ungültige Lösungen deutlich durch):



Ersatzdiagramme

Für Aufgabe 4 a)

Zugriffe mit Seitenfehler/Erfolgreicher Zugriff (nicht Zutreffendes durchstreichen!)

Adresse: `0xE85FD000`

Index	phys. Adresse	w	p
...			
...			

Adresse:

Index	phys. Adresse	w	p
...			
...			

Für Aufgabe 6 a)

`Initialwerte der Semaphore: S1:` `S2:` `S3:`

```
f1() {
    
    puts("2");
    
    puts("3");
    
}
```

```
f2() {
    
    puts("1");
    
    puts("6");
    
}
```

```
f3() {
    
    puts("4");
    
    puts("5");
    
}
```

Für Aufgabe 6 b)

```
1  // gemeinsamer Speicher
2  Semaphore mutex = 1;
3  Semaphore not_empty = 0;
4  struct list *queue;
5  struct element e;
```

```
1  // Erzeuger
2  while(true) {
3      wait(&mutex);
4
5      enqueue(queue, e);
6
7      signal(&not_empty);
8
9      signal(&mutex);
10 }
```

```
1  // Verbraucher
2  while(true) {
3      wait(&mutex);
4
5      wait(&not_empty);
6
7      e = dequeue(queue);
8
9      signal(&mutex);
10 }
```