1. Warum können sog. *Off-by-One* Speicherüberläufe - oder Überläufe von weniger als 4 Byte - oft nicht in der Praxis ausgenutzt werden?

2. Ist Java - wie so oft behauptet - wirklich sicher? Testen Sie:

a) Buffer-Overflows

b) Integer-Overflows

c) Race-Conditions beim Arbeiten mit Dateien

d) Werden *Symbolic-Links* sicher gehandhabt?

3. Wie verhalten sich Gleitkommazahlen bei Wertebereichsüberschreitungen (in C)?

4. Programmieraufgabe:

a) Ein Programm mit zwei *char-Arrays* (A, B) soll Benutzereingaben von der Tastatur einlesen und in A speichern. Dabei wird die Größe von A missachtet (Speicherüberlauf). Nach dem Überlauf soll das Programm seinen *Stack* ausgeben.

b) Schreiben Sie ein Programm mit zwei Funktionen (A, B). Rufen Sie Funktion A über den Zeiger (der auf dem *Heap* liegt) ptrA auf. Irgendwo vor dem Funktionsaufruf muss ein Speicherüberlauf stattfinden, der es einem Angreifer erlaubt ptrA mit der Adresse von B zu überschreiben. Alle benötigten Informationen kann das Programm selbst ausgeben... oder arbeiten Sie doch einfach mit dem *gdb*. Viel Spass beim Spielen!

5. Löschen von sensiblen Daten im Speicher.

a) Wie kann man verhindern, dass der *gcc* Funktionen wie *memset(3),* aufgrund von übertriebener Optimierung, entfernt?

b) In Java ist es nicht möglich Speicherseiten mit *mlock(2)* zu schützen, und auch das Löschen von Objekten kann nur wenig beeinflusst werden (der *Garbage Collector* ist dafür verantwortlich). Stellen Sie sich vor, Sie haben eine Klasse, die Krypto-Schlüssel speichern muss und wollen verhindern, dass ein Angreifer den *Heap* der JVM analysiert, um Schlüssel-Bits zu eruieren. Wie können Sie dieses Dilemma lösen?