ulm university universität
uulm

Institut für Numerische Mathematik

Dr. Andreas F. Borchert
Fabian Berstecher

20. April 2017
Blatt 1

Systemnahe Software II (SS 2017)

Abgabe bis zum 27. April 2017, 16:00 Uhr

Lernziele:

- Sicherung von Speicherbereichen mit sensiblen Daten

Hinweis:

Melden Sie sich für diese Veranstaltung in SLC an. Ohne diese Anmeldung können Sie Ihre Lösung nicht mit *submit* einreichen.

Übungsbetrieb

Die Übungen finden jeweils am Donnerstag von 16 bis 18 Uhr in der Helmholtzstraße 22 im Raum E.03 statt. Die Anwesenheit in der Übungsstunde ist freiwillig. Da es keine Vorleistung mehr für Systemnahe Software II gibt, werden die eingereichten Lösungen nicht mehr bewertet.

Aufgabe 1: Speicherbelegung für sensible Daten

Ihnen ist sicherlich allen der Heartbleed-Bug zumindest namentlich bekannt. Dabei handelt es sich um einen Fehler in der OpenSSL-Bibliothek, die es einem durch einen lesenden Buffer-Overrun ermöglicht, andere Daten (unter Umständen auch Passwörter und private Schlüssel) im Speicher auszulesen. Eine sehr gute Veranschaulichung des dadurch entstandenen Problems finden Sie hier: *http://xkcd.com/1354/*

Das Ziel diese Aufgabe ist es, eine eine Methode zur Speicherreservierung zu implementieren, bei dem die kritischen Daten immun gegen das Auslesen per Buffer-Overrun sind. Dazu sollen Sie die folgenden zwei Funktionen implementieren:

```
void* secure_malloc(size_t nbytes);
void secure_free(void* ptr);
```

Hierbei wird der gewünschte Speicherbereich aufgerundet auf ein Vielfaches der Kachelgröße des Systems (siehe *getpagesize()*) und zusätzlich je vor und hinter den benötigten Kacheln je eine weitere Kachel belegt, die gegen das Auslesen und Schreiben geschützt ist. Dieser gesamte Bereich lässt sich mit *mmap* belegen. Danach können die Zugriffsrechte auch einzelner Teilbereiche mit Hilfe von *mprotect* so verändert werden, dass die Kachel vor und hinter den sensiblen Daten keine Lese- oder Schreibzugriffe zulassen. Sollte es danach zu einem Versuch mittels eines Buffer-Overruns kommen, bis hin zu den sensiblen Daten zu lesen, würde der Weg zunächst durch die gesperrten Bereiche führen. Ein Zugriff auf die gesperrten Bereiche hätte jedoch den Abbruch des Programms zur Folge (*segmentation fault*), so dass das Auslesen der sensiblen Daten unterbunden wird.

Bei *secure_free* ist der gesamte Bereich mit Hilfe von *munmap* freizugeben. Ähnlich wie bei *free* wird die Größe des Speicherbereichs nicht übergeben, so dass diese auf geeignete Weise wieder ermittelt werden muss.

Um zu überprüfen, ob Ihr Programm auch korrekt funktioniert, sollen Sie noch ein kleines Testprogramm schreiben, das als Parameter eine Anzahl von Bytes einliest, die dann mit *secure_malloc* reserviert werden. Es sollen die Anzahl der reservierten Bytes und die Speicheradresse ausgegeben werden. Außerdem können Sie in der Shell mit dem Kommando *pmap* überprüfen, wie viel Speicher reserviert wurde.

Anschließend soll Ihr Testprogramm es ermöglichen, Speicheradressen einzugeben und versuchen, von dieser aus den Speicher auszulesen.

Reichen Sie bitte Ihre Lösung auf dem Rechner Thales mit dem Kommando

```
submit ss2 1 secure_malloc.c secure_malloc.h testit.c
```

ein.

Viel Erfolg!