– Praktikumsaufgabe 9 –

Thema: *Authentifizierung*

Zielstellung: Implementierung einer verschlüsselten Authentifizierung, Kennenlernen einschlägiger Bibliotheksfunktionen, Diskussion potentieller Schwachstellen in der Implementierung

1. Implementieren Sie ein Werkzeug, um Zugänge zum System anzulegen analog dem `adduser`-Kommando. Dieses Werkzeug soll zu einem als Kommandozeilenparameter übergebenen Nutzerkennzeichen eine Passphrase abfragen, diese mittels `crypt()` DES-verschlüsseln und Nutzerkennzeichen und generierten Hash, getrennt durch einen Doppelpunkt, in einer Datei `mypasswords.txt` ablegen. Neue Accounts sollten an die Datei zeilenweise angehängt werden. Das Salz können Sie zunächst statisch festlegen. Die Länge der Passphrase sollte beschränkt sein. Das Löschen von Nutzern geschieht, indem mit einem Texteditor die entsprechende Zeile aus `mypasswords.txt` gelöscht wird.

   Hinweise: Machen Sie das Management robust. Verhindern Sie das doppelte Anlegen von Accounts und schließen Sie zu kurze Passphrasen aus. Ebenso sollten Sie sich gegen versehentlich falsch eingegebene Passphrasen sichern (wie?).

   Zum Einlesen der Passphrase können Sie entweder `getpass()`[1] nutzen oder dessen Funktionalität „per Hand“ nachprogrammieren.

   Um `crypt()` und `getpass()` zu nutzen, müssen Sie gegen die Bibliothek `crypt`[2] linken (Switch `-lcrypt`) und unbedingt (!) `crypt.h` inkludieren[3].

   Weitere Hinweise zur Nutzung beider Funktionen liefern die `man`-Pages.

2. Implementieren Sie nun ein Programm, das ein beliebiges Kommando als Parameter übernimmt und dieses Kommando (z. B. per `execl()`) ausführt. Unmittelbar vor der Ausführung soll sich der Nutzer jedoch unter Nutzung von `mypasswords.txt` authentifizieren.

   Hinweis: Es gibt 6 verschiedene `exec()-Varianten`, die sich nur in Details ihrer Parameter unterscheiden (z. B., ob der Pfad nach dem auszuführenden Binärimage durchsucht werden soll). Allen gemein ist:

   - sie überlagern das Abbild des rufenden Prozesses,
   - sie beginnen sofort, das neue Programm abzuarbeiten,

[1] Die Funktion getpass() ist als „obsolet“ gekennzeichnet, da sie nicht threadsicher und ihr Name irreführend ist. Außerdem soll ihre Funktionalität leicht implementierbar sein. Naja.

[2] Nicht mit der OpenSSL-Bibliothek `crypto` verwechseln!

[3] Vergessen Sie das `#include`, dann wird anstelle der Verschlüsselungsfunktionalität stillschweigend eine andere Funktion gelinkt, die zur Laufzeit schwer zu entdeckende Fehler liefert.

- sie kehren nur zurück, wenn der Aufruf fehlschlägt (d. h., es wird nicht in den Ursprungsprozess zurückgesprungen!),
- sie übernehmen als Parameter eine Pfadangabe zum auszuführenden Programm.

Am einfachsten ist die Nutzung von `execl()`, das einen Zeiger auf eine Zeichenkette übernimmt, die das auszuführende Programm enthält. Danach folgen weitere `char*`, auf dessen Argumente, die im einfachsten Falle NULL sind. Am elegantesten ist die Nutzung von `execvp()`.

3.* Eine potentielle Angriffsmöglichkeit auf einen Authentifizierungsmechanismus besteht darin, dass ein Angreifer nach dem Aufruf von `getpass()` durch einen autorisierten Nutzer einen zweiten Aufruf „hinterherschiebt“ [4] um eine sehr kurze Passphrase einzugeben. Finden Sie experimentell heraus, ob es möglich ist, durch eine solche Technik Teile der zuvor eingegebenen Passphrase zu restaurieren. Wie kann man sich ggf. dagegen schützen?

[4] Wie dies technisch geschieht, behandeln wir später; es ist u. U. auf alle Fälle möglich.