htw saar

Studiengang Kommunikationsinformatik (Bachelor)
Studiengang Praktische Informatik (Bachelor)
Prof. Dr.–Ing. Damian Weber
Dipl.-Phys. Michael Meßner

Security Engineering

5. Übung

Aufgabe 1 (Password-Cracking (Dictionary–Attack))

Schreiben Sie ein einfaches Password-Cracking-Tool.

Die Wörterbuch-Attacke (Dictionary–Attack) bildet den verschlüsselten (oder gehash-ten) `passwd`–Eintrag für alle Wörter des Wörterbuchs. Wird dabei eine Übereinstimmung gefunden, so ist das Klartextpassword bekannt.

Beispiel: bei einem bekannten `passwd`–Eintrag von

```
$1$ZYXWVUTS$xN7BRlVyRwgiKy.Gfnss0/
```

würde die Wörterbuch-Attacke bei Benutzung der Wörter aus `/usr/share/dict/words` diese testen, z.B.

```
A       $1$ZYXWVUTS$y6t.HbZrqRxPSI5PT1eoW1
...
secpar  $1$ZYXWVUTS$Px29OT6yKgy4P4C5l1yAd0
secque  $1$ZYXWVUTS$2h2opR3NHam9NUqEGmHbE/
secre   $1$ZYXWVUTS$3kghTnCpC904O5tH2RInH1
secrecy $1$ZYXWVUTS$oiE5.nXUXQ/sF.Y.M5PlI0
secret  $1$ZYXWVUTS$xN7BRlVyRwgiKy.Gfnss0/
```

und bei `secret` merken, dass der gesuchte Wert gefunden wurde, somit auf das Passwort `secret` schließen.

Schreiben Sie ein Shellskript, das ein Kommando der Form

```
openssl passwd -1 -salt ...
```

benutzt, um die Passwörter folgender Benutzer zu finden. Diese stammen alle aus einer UNIX `/usr/share/dict/words` Datei, siehe

`https://www-crypto.htwsaar.de/weber/download/words-dab00c99.txt`:

```
Steffi.Jones    $1$O7v0C21Z$2FH7ib2Dxtoq6B83qTgON1
Marco.Reus      $1$Jebn3vQ5$2k..iqxtXNwfsCFAamWCS0
Almuth.Schult   $1$0ngrMRa1$uXLzWhnrYzmiRM3fi8Nde1
Manuel.Neuer    $1$1aaPttrp$VoF2rkOyC/tE.DxzQuuIY1
Birgit.Prinz    $1$7ieEwjFr$T/jwatbzqhLZNVDEfymB41
```

Aufgabe 2 (ONE-TIME-Passwörter / TAN-Listen)

Erzeugen Sie für einen gegebenen Username u und einer Anzahl n mit einem Shell-Skript `generate_tan` eine TAN-Liste von n Nummern für u, die für ein Online-Banking-System benutzt werden könnten. Verwenden Sie hierbei das in der Vorlesung vorgestellte Verfahren (Abschnitt *Secure One–Time–Passwords and TANs*).
Simulieren Sie den Server durch ein Shell-Skript `bank_transaction`, das

- in einer Endlosschleife läuft,
- nicht mit [STRG+C] abgebrochen werden kann und
- von der Tastatur zwei Eingaben entgegennimmt
  - Username,
  - TAN

Pro User soll in einem Unterverzeichnis `TAN` eine Datei existieren, die die aktuelle TAN-Liste enthält. Überlegen Sie sich einen geeigneten Mechanismus, dass die aktuelle TAN geprüft wird.
Das Skript `bank_transaction` soll ausgeben

- Zugriff erlaubt, wenn die aktuelle TAN richtig war
- Zugriff verweigert, wenn
  - die aktuelle TAN falsch war oder
  - die TAN-Liste des Users aufgebraucht ist

Aufgabe 3 (Limits und Signale)

Sinn dieser Aufgabe ist die Beobachtung von Prozessen, die vorgegebene Limits überschreiten.
Setzen Sie vor Ausführung eines Prozesses folgende Limits

- cpu time
- stack size
- file size

und sorgen Sie dafür, dass der Prozess diese Limits überschreitet.
Statten Sie den Prozess mit einem Signalhandler aus, der das vom Betriebssystem generierte Signal abfängt und sich danach beendet.