GSS-Übungsblatt 3

Chamier, Eickhoff, Gäde, Hölzen, Jarsembinski · SoSe 2016

1 Rechnersicherheit

1.1 Zugangs- und Zugriffskontrolle

a) Die **Zugangskontrolle** dient der Identifikation beim Zugang auf Betriebsmittel und verbietet dabei unbefugten Zugang. Es wird nur mit berechtigten Partnern kommuniziert.
Die **Zugriffskontrolle** hingegen gibt einem identifizierten Kommunikationspartner nur den Zugriff auf Objekte, für die er Zugriffsrechte hat.

b) Eine Zugangskontrolle ohne interne Zugriffkontrolle ist durchaus sinvoll, da zwar keine Unterscheidung im Zugriff auf Objekte gemacht wird, sehr wohl jedoch der Zugang zum System insgesamt reguliert wird.

c) Für eine Zugriffskontrolle muss der Nutzer des Systems identifiziert sein, um ihm Zugriffsrechte zuordnen zu können.

1.4 Realisierung eines Online-Ticket-Verkaufs

a) **Rolle**: Außenstehender (Kunde)
Verbreitung: Onlinetransaktion(1), Ticketübermittlung(2), andere Kunden(3), Alte Codes von Tickets auswerten(4)
Verhalten: (1,2) passiv beobachtend, (3,4) aktiv beobachtend
Rechenkapazität: beschränkt.

b) Die Barcodes werden auf einem System im Kino erzeugt und auf dieses System kann man nur lokal zugreifen.
Die Barcodes werden pseudozufällig erzeugt, beliebig häufiges hashen, codes sind Unikate.

Kauft nun also jemand ein Ticket online, so wird auf besagtem System ein Barcode erzeugt und an diesen per Mail versandt. (Davon ausgehend, dass die Mail des Kunden sicher ist).

Kommt also jetzt besagter Kunde zum Kino, so wird der Barcode vom lokalen Server abgeglichen.

GSS-Übungsblatt 3

Chamier, Eickhoff, Gäde, Hölzen, Jarsembinski · SoSe 2016

2 Rechnersicherheit

1.

Listing 1: Timer.java

```
1
2  public class Timer {
3    public static void main(String[] args) {
4      char[] a = {'a','b','c','d'};
5      char[] b = {'a','b','c','d'};
6      char[] c = {'a','b','f','e'};
7      Timer t = new Timer();
8      System.out.println(t.timeAttack(a,b));
9      System.out.println(t.timeAttack(a,c));
10   }
11
12   long timeAttack(char[] a, char[] b) {
13     long startTime = 0;
14     long endTime = 0;
15     long deltaTime = 0;
16
17     startTime = System.nanoTime();
18     passwordCompare(a,b);
19     endTime = System.nanoTime();
20     deltaTime = endTime - startTime;
21     return deltaTime;
22   }
23
24   boolean passwordCompare(char[] a,char[] b) {
25     int i;
26     if (a.length != b.length) return false;
27     for
28     (i=0; i<a.length && a[i]==b[i]; i++);
29     return i == a.length;
30   }
31 }
```

3.
Zuerst testet der Angreifer auf Länge, d.h. wenn das Passwort prüfen länger dauert, ist die Länge erreicht.
Danach testet man Zeichen für Zeichen durch, von links nach rechts, jedes mal wenn ein weiteres Zeichen richtig ist, braucht das Programm länger.

3 Real-World-Brute-Force Angriff

Der Zugangscode für das svs-Submission-Tool besteht aus einem festen Header (GSS16), den wir als bekannt voraussetzen, einer Zahl von 0-9 und vier fünfstelligen Zeichenketten aus Großbuchstaben und Zahlen (36 Zeichen).
Die Wahrscheinlichkeit einen der 100 vergebenen Codes zu treffen ist also:

$$P = \frac{100}{10 \cdot 36^{20}} \approx 0 \tag{1}$$