

**Security**  
**SoSe 23**  
**LV 4120, 7240**  
**Übungsblatt 11**

**Aufgabe 11.1 (Schwachstellen):**

- a) Wie kann Malware, welche eine Schwachstelle ausnutzt, grundsätzlich auf ein System einwirken?
- b) Zählen Sie auf, welche Bewertungskriterien der Basic Score des CVSS nutzt, um die Ausnutzbarkeit einer Schwachstelle zu ermitteln.
- c) Nennen Sie, welche Analysemethoden es gibt, um Schwachstellen in einem Code zu finden. Erläutern Sie die Unterschiede zwischen den Testmethoden und die Vor- und Nachteile.
- d) Erläutern Sie, wie die Gegenmaßnahme Stack Canary funktioniert.

**Aufgabe 11.2 (Quellcode-Review um Schwachstellen zu finden):**

Analysieren Sie die folgenden Quellcode-Ausschnitte auf Schwachstellen und beschreiben Sie:

- A In welcher Zeile die Schwachstelle liegt und wodurch sie ermöglicht wird.
  - B Wie ein Exploit aussehen könnte, um die Schwachstelle auszunutzen.
  - C Was für einen Effekt die Ausnutzung der Schwachstelle hätte.
  - D Wie die Schwachstelle behoben werden könnte.
- a) Pizzabestellung: Empfange die Pizzabestellungen und gebe die Kosten zurück. *Hinweis: Die Funktion `receive(n)` gibt  $n$  Bytes zurück.*

```
1  short berechne_pizza_kosten() {  
2      short preis = 8;  
3      short pizzen = receive(2);  
4      return preis * pizzen;  
5  }
```

- b) Heartbeat: Empfange einen Request und sende die gleiche Nachricht als Lebenszeichen zurück

```

1 //Empfange Daten bis ein newline Character gesendet wird
2 char* heartbeat_request = receive_byte_array_newline_end();
3 short len = receive(2);
4
5 char* heartbeat_reply = (char*) malloc(len);
6 memcpy(heartbeat_reply, heartbeat_request, len);
7
8 //Sende Byte Array der Länge len
9 send(heartbeat_reply, len);

```

- c) Passwortabgleich: Empfange das Passwort Byte-weise und gleiche das empfangene mit dem gespeicherten Passwort ab

```

1 bool pruefe passwort(char* passwort, unsigned int laenge) {
2     bool gleichheit = false;
3     char empfangenes_passwort[laenge];
4     //Empfange Passwort Byte-weise
5     for(unsigned int i = 0; i <= laenge; i++) {
6         empfangenes_passwort[i] = receive(1);
7     }
8     //Vergleiche Passwörter
9     if(memcmp(passwort, empfangenes_passwort, laenge)) {
10         gleichheit = true;
11     }
12     return gleichheit;
13 }

```

### Aufgabe 11.3 (Ungewollte Hilfsfunktionen):

Gegeben sei folgendes C-Programm:

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4
5  #define MAX_CMD 1000
6
7  int main(int argc, char** argv) {
8      char cat[] = "cat ";
9      char command[MAX_CMD];
10     size_t commandLength;
11
12     commandLength = strlen(cat) + strlen(argv[1]) + 1;
13
14     strncpy(command, cat, commandLength);
15     strncat(command, argv[1], (commandLength - strlen(cat)));
16
17     system(command);
18 }
```

- a) Analysieren Sie den gegebenen Code und versuchen Sie rauszufinden, was dieser Code bewirkt.
- b) Kompilieren Sie das Programm und probieren Sie es mit der Datei "story.txt", welche der Übung beigelegt ist, aus.
- c) Was für eine Schwachstelle ist in dem Programmcode enthalten und wie kann man diese Ausnutzen. Geben Sie ein Beispiel dafür.
- d) Geben Sie ein Beispiel, wie dieses Programm als Malware eingesetzt werden kann. Schlagen Sie zusätzlich eine Lösung, vor um diesen Angriff mit der Malware zu detektieren oder zu umgehen.