Buffer Overflows

Valentin Brandl <mail@vbrandl.net>

Fakultät Informatik und Mathematik

20. September 2022

Inhalt

- Maschinennahe Programmiersprachen ohne Memorysafety (z.B. C, C++, Assembly, FORTRAN) erlauben es, Speicher beliebig zu beschreiben
- Bei fehlender Validierung kann ein Programm mehr Speicher schreiben, als eigentlich reserviert wurde und dabei andere Daten im RAM überschreiben
- Entsprechend präparierter Input kann dazu führen, dass ein Angreifer den Ablauf der Programmausführung übernehmen kann

```c
#include<stdio.h>
#include<string.h>
void foo(char *input) {
  char buf[50];
  strcpy(buf, input);
  puts(buf);
}
int main(int argc, char **argv) {
  foo(argv[1]);
}
```

Stack Layout

Execution Flow

- Shellcode ist der Maschinencode, der nach Übernahme des Ausführungsablauf ausgeführt werden soll
- Buffer kann klein sein $\Rightarrow$ Shellcode häufig auf Größe optimiert
- Häufig Strings als Eingabe
- In C terminiert mit `\0` $\Rightarrow$ Payload darf kein `0x00` enthalten, da alles danach abgeschnitten wird
- $\Rightarrow$ Selbst schreiben ist möglich, aber aufwändig und setzt Kenntnisse in Assembly und der anzugreifenden Architektur/OS/... voraus
- Verfügbare, öffentliche Sammlungen verwenden:
  - `https://shell-storm.org/shellcode/`
  - `https://www.exploit-db.com/shellcodes`

Ablauf

- Shellcode im Speicher plazieren
- Buffer überschreiben
- *IP* überschreiben
- *IP* auf Shellcode zeigen lassen

- Address Space Layout Randomization (ASLR)
- $w \hat{} x$ Memory
- Runtime Bounds Checks
- Typesystem basierte Lösungen [Con07]

[Con07] Condit, Jeremy and Harren, Matthew and Anderson, Zachary and Gay, David and Necula, George C. “Dependent Types for Low-Level Programming”. In: *Programming Languages and Systems*. 2007.