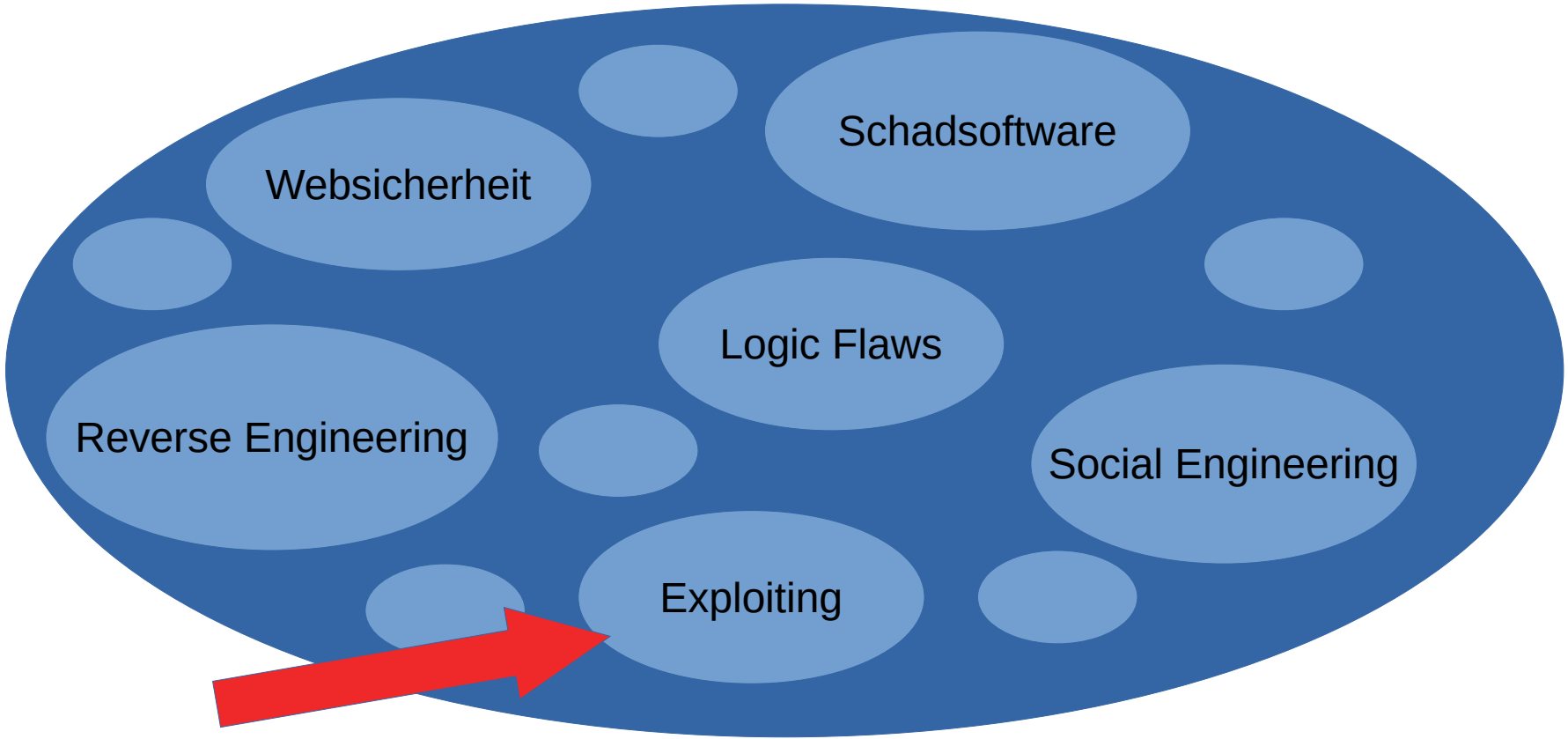
Informationssicherheit und IT-Forensik

– Übung –

UEB-03 – Buffer Overflow

Einordnung

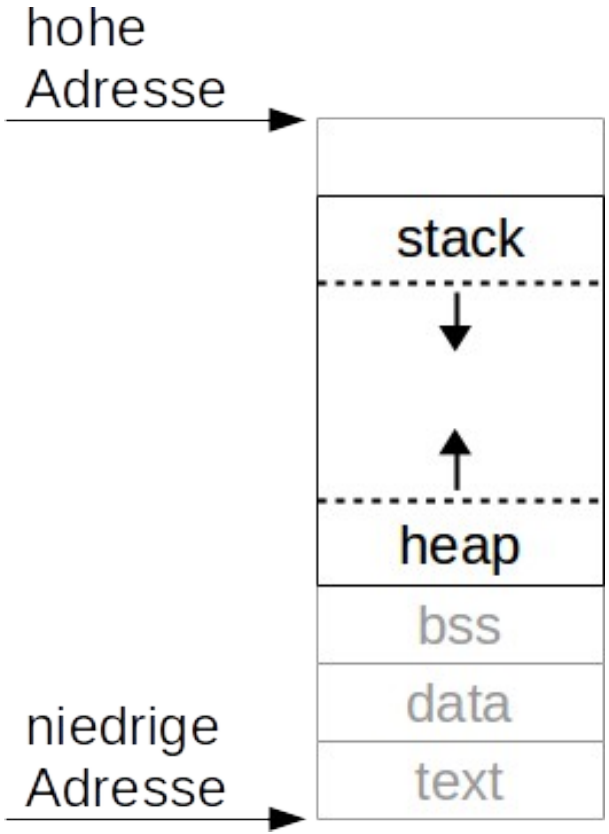

Websicherheit
Schadsoftware
Logic Flaws
Reverse Engineering
Social Engineering
Exploiting

Grundlagen

- Ein Programm nimmt Eingabedaten vom Benutzer entgegen, ohne die Länge korrekt zu prüfen
- Das Programm reserviert eine Variable (Buffer) mit einer Länge von **n** Zeichen, um die Nutzereingabe zu speichern
- Die Eingabe, die ein Angreifer tätigt, ist größer als **n** Zeichen
- Beim Versuch, die Eingabe in der Variable abzuspeichern, überschreibt das Programm versehentlich Daten, die hinter der Variable im Speicher liegen, der Buffer läuft also über

Speicherlayout eines Programms

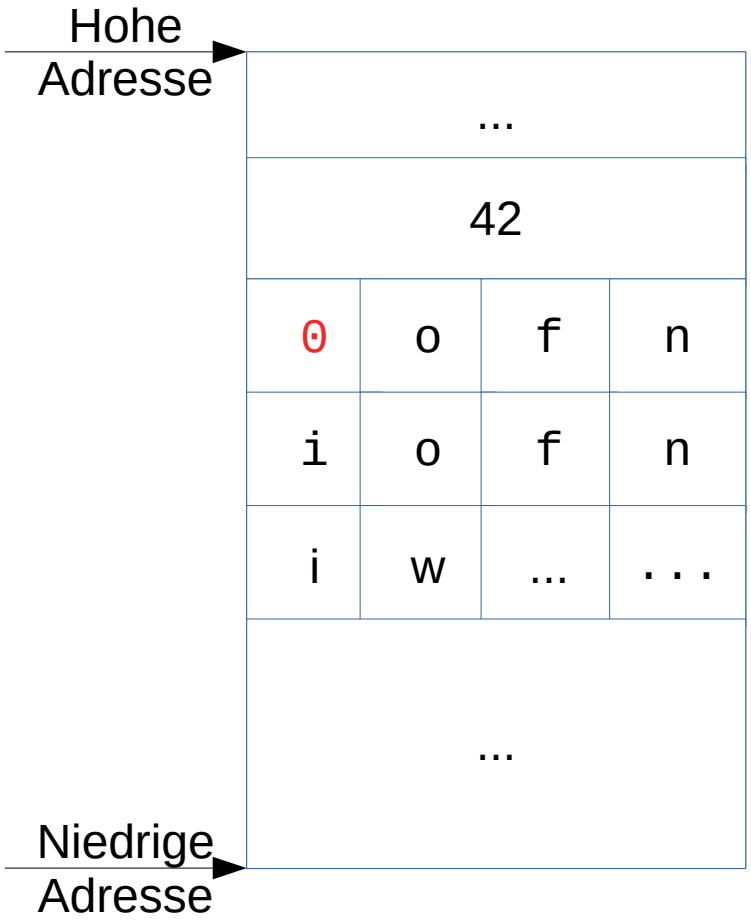
- Wichtig ist:
  - Der Stack ist der Bereich, in dem ein Programm Platz für Variablen reserviert
  - Speicheradressen auf dem Stack beginnen bei hohen Werten, später reservierte Variablen haben kleinere Adressen
  - Eine Variable selbst wird dennoch von kleinen zu großen Werten hin befüllt



Speicherlayout eines Programms

Beispiel:

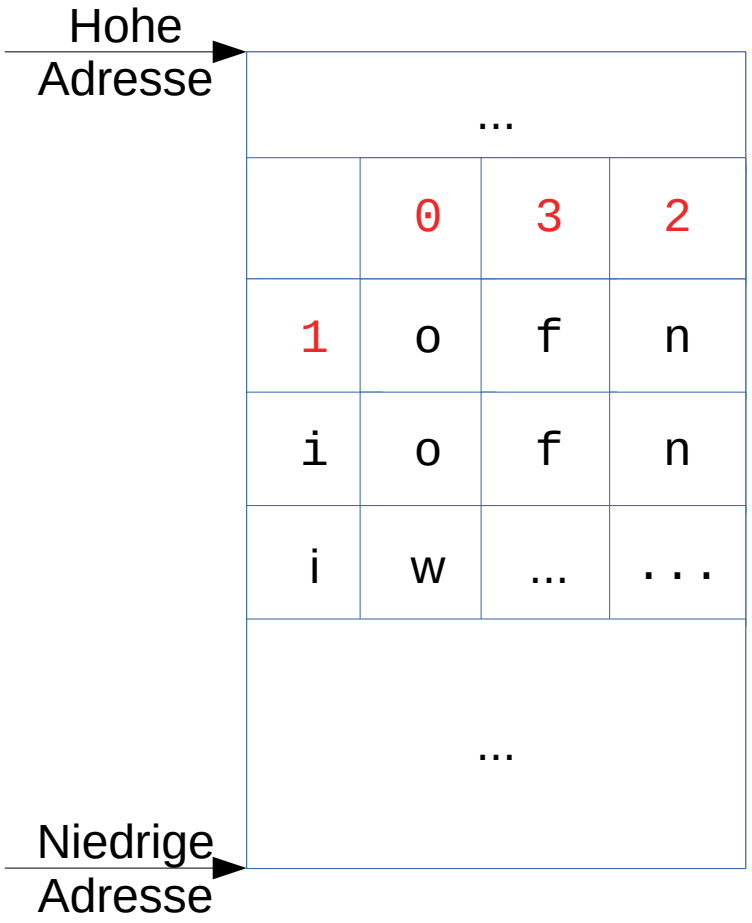
```
int main() {
  int zahl = 42;
  char str[10];
  str = “winfoinfo“;
}
```



Speicherüberlauf...

Beispiel:

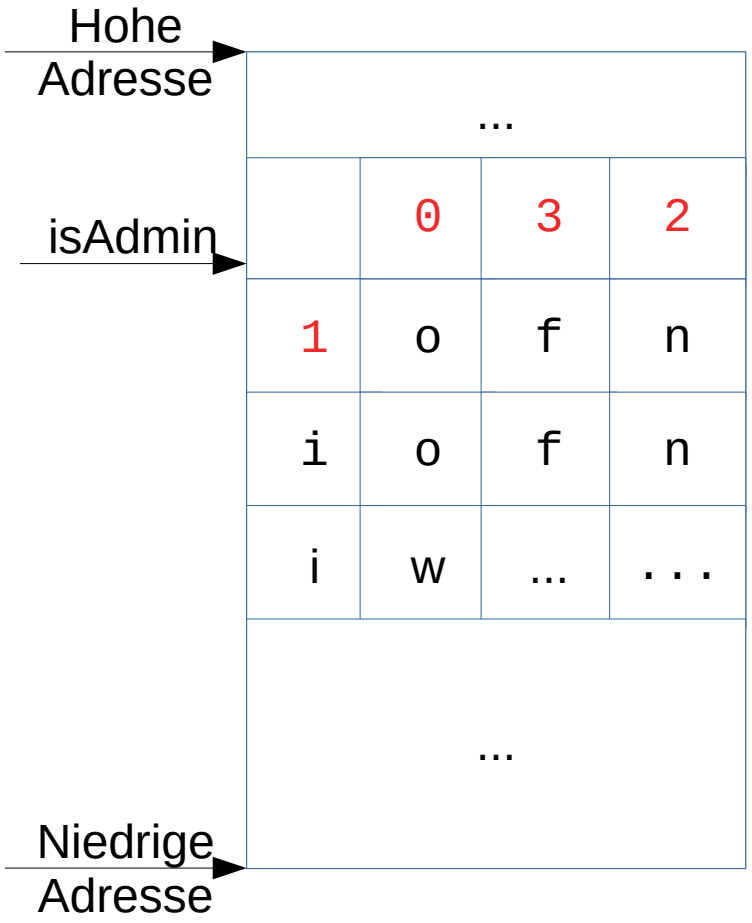
```
int main() {
  int zahl = 42;
  char str[10];
  str = “winfoinfo123“;
}
```



… mit Folgen

Beispiel:

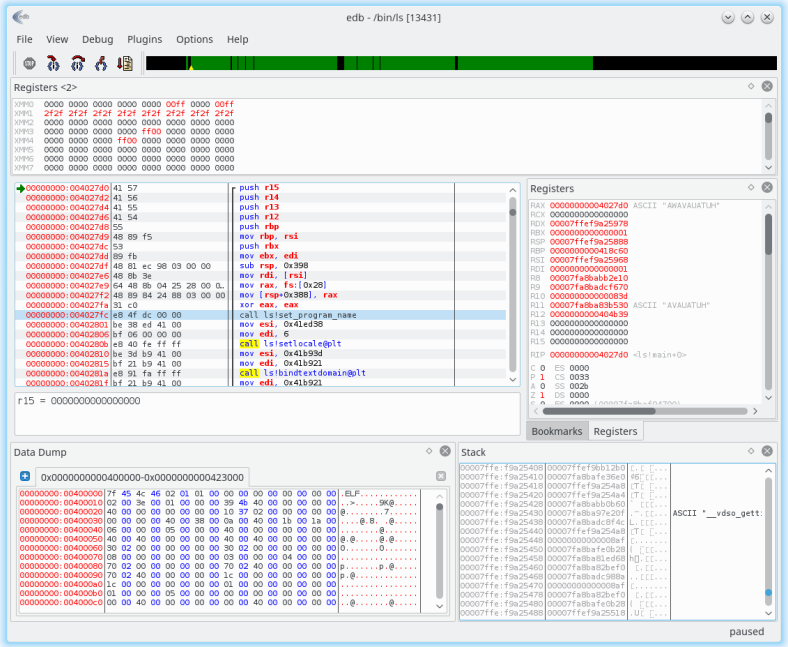
```
int main() {
  int isadmin = 0;
  char str[10];
  str = “winfoinfo123“;
  …
  if(isAdmin != 0){
    //Code, den nur der Admin ausführen darf
}
```



Speicherüberlauf: Praxisbeispiel

- Auf dem Desktop der Kali-VM befindet sich der Ordner `aufgaben/02_buffer_overflow`
- Öffnen Sie ein Terminal in diesem Ordner
- Einmalig: Deaktivieren von ASLR mit `./disableSecurity.sh`
- Kompilieren des C-Programms mit `./compile.sh`
- Ausführen mit `./programm <nutzereingabe> debug`
- `debug` ist optional, aber gibt interessante Informationen über das Speicherlayout des Programms aus
- Versuchen Sie, das Geheimnis ausgeben zu lassen
- Umprogrammieren (Reihenfolge der Variablen tauschen, neu kompilieren)
- Erneut exploiten (?)

Speicherüberlauf: Praxisbeispiel



- Wie sieht das ganze auf dem Stack in der Praxis aus?

- Debugging von Programmen würde hier den Rahmen sprengen. Für Interessierte:
  - `$ sudo apt install edb-debugger`
  - `$ edb --run programm <nutzereingabe>`

Exkurs: Reverse Engineering

- Kompilieren eines Programms erzeugt prozessorlesbaren Binärcode
- Wenn man diese Binärdatei nimmt und versucht, daraus wieder Informationen zu gewinnen, spricht man von *Reverse Engineering*
- Oft sehr komplex, da Hersteller versuchen, die Binärdaten zu verschleiern
- Manchmal aber auch sehr simpel!

Reverse Engineering: ~~Praxisbeispiel~~

- Paralenz – Eine Unterwasserkamera für Taucher
- ~~Download der aktuellsten Firmware-Version für die *DiveCamera*+ von https://www.paralenz.com/downloads (in Kali)~~
- Auflisten von in der Firmware enthaltenen Zeichenketten mit dem Kommandozeilentool strings:
  - `strings firmware.bin` oder: `strings -n MINDESTLÄNGE firmware.bin`
- Filtern der Ausgabe mit dem Kommandozeilentool grep:
  - `strings firmware.bin | grep -i password`
  - `Strings firmware.bin | grep -i :7:::` (findet Einträge in der zentralen /etc/shadow-Datei)