Auf den Spuren einer Sicherheitslücke

Jonas Wagner
Dependable Systems Lab, EPFL

```c
// TecDay Login

#include <stdio.h>
#include <unistd.h>
#include <string.h>

const char *SECRET = "youllneverguess";

void read_password(char *password) {
    int c;
    int i = 0;

    c = getchar();
    while (c != '\n' && c != EOF) {
        password[i] = c;
        i = i + 1;
        c = getchar();
    }
    password[i] = 0;
}

int authorize() {
    char password[16];

    printf("Enter Password: ");
    read_password(password);

    return !strcmp(password, SECRET);
}

void start_shell() {
    printf("Have a lot of fun!\n");
    execl("/bin/sh", "sh", (char*)0);
    printf("After execl...\n");
}

int main() {
    printf("Welcome to TecDay Login\n");

    if (authorize()) {
        printf("Login successful\n");
        start_shell();
    } else {
        printf("Login failed\n");
        return 1;
    }

    return 0;
}
```

```
Dump of assembler code for function authorize:
   0x85a4 <+0>:  push   {lr}                    ; Rücksprungadresse sichern
   0x85a8 <+4>:  sub    sp, sp, 20              ; Platz für 20 Bytes reservieren
   0x85ac <+8>:  ldr    r0, [pc, 40]            ; Text "Enter Password" laden
   0x85b0 <+12>: bl     0x838c <printf>         ; Text ausgeben
   0x85b4 <+16>: mov    r0, sp                  ; Passwort-Speicheradresse laden
   0x85b8 <+20>: bl     0x8548 <read_password>  ; Funktion read_password aufrufen
   0x85bc <+24>: mov    r0, sp
   0x85c0 <+28>: ldr    r3, [pc, 24]            ; SECRET laden
   0x85c4 <+32>: ldr    r1, [r3]
   0x85c8 <+36>: bl     0x8380 <strcmp>         ; Funktion strcmp aufrufen
   0x85cc <+40>: rsbs   r0, r0, 1               ; Überprüfen, ob strcmp "wahr"...
   0x85d0 <+44>: movcc  r0, 0                   ; ... zurückgibt oder "falsch"
   0x85d4 <+48>: add    sp, sp, 20              ; 20 Bytes Speicherplatz freigeben
   0x85d8 <+52>: pop    {pc}                    ; ENDE
```

```
Dump of assembler code for function main:
   0x85e4 <+0>:  push   {r3, lr}                ; Register sichern
   0x85e8 <+4>:  bl     0x85a4 <authorize>      ; Funktion "authorize" aufrufen
   0x85ec <+8>:  cmp    r0, 0                   ; Falls Rückbabewert 0 ist, ...
   0x85f0 <+12>: beq    0x8608 <main+36>        ; ... nach unten springen
   0x85f4 <+16>: ldr    r0, [pc, 28]            ; Text "Login successful" laden
   0x85f8 <+20>: bl     0x83a4 <puts>           ; Text ausgeben
   0x85fc <+24>: bl     0x8514 <start_shell>    ; Funktion "start_shell" aufrufen
   0x8600 <+28>: mov    r0, 0
   0x8604 <+32>: pop    {r3, pc}                ; Register wiederherstellen, ENDE.
   0x8608 <+36>: ldr    r0, [pc, 12]            ; Text "Login failed" laden
   0x860c <+40>: bl     0x83a4 <puts>           ; Text ausgeben
   0x8610 <+44>: mov    r0, 1
   0x8614 <+48>: pop    {r3, pc}                ; Register wiederherstellen, ENDE
```

```c
// TecDay Login

#include <stdio.h>
#include <unistd.h>
#include <string.h>

const char *SECRET = "youllneverguess";

void read_password(char *password) {
    int c;
    int i = 0;

    c = getchar();
    while (c != '\n' && c != EOF) {
        password[i] = c;
        i = i + 1;
        c = getchar();
    }
    password[i] = 0;
}

int authorize() {
    char password[16];

    printf("Enter Password: ");
    read_password(password);

    return !strcmp(password, SECRET);
}

void start_shell() {
    printf("Have a lot of fun!\n");
    execl("/bin/sh", "sh", (char*)0);
    printf("After execl...\n");
}

int main() {
    printf("Welcome to TecDay Login\n");

    if (authorize()) {
        printf("Login successful\n");
        start_shell();
    } else {
        printf("Login failed\n");
        return 1;
    }

    return 0;
}
```

```
Dump of assembler code for function read_password:
   0x8548 <+0>:  push   {r4, r5, r6, lr}       ; Register sichern
   0x854c <+4>:  mov    r5, r0
   0x8550 <+8>:  ldr    r3, [pc, 72]
   0x8554 <+12>: ldr    r0, [r3]
   0x8558 <+16>: bl     0x8398 <_IO_getc>      ; ein Zeichen einlesen
   0x855c <+20>: cmp    r0, 10                 ; ist es 10 (Ende der Zeile)?
   0x8560 <+24>: cmnne  r0, 1                  ; ist es -1 (Ende der Eingabe)?
   0x8564 <+28>: beq    0x8590 <read_password+72>  ; falls ja, ans Ende springen
   0x8568 <+32>: mov    r4, 0
   0x856c <+36>: ldr    r6, [pc, 44]
   0x8570 <+40>: strb   r0, [r5, r4]           ; Zeichen an Position i (r4) speichern
   0x8574 <+44>: add    r4, r4, 1              ; Position i (r4) um 1 erhöhen
   0x8578 <+48>: ldr    r0, [r6]
   0x857c <+52>: bl     0x8398 <_IO_getc>      ; ein Zeichen einlesen
   0x8580 <+56>: cmp    r0, 10                 ; ist es 10 (Ende der Zeile)?
   0x8584 <+60>: cmnne  r0, 1                  ; ist es -1 (Ende der Eingabe)?
   0x8588 <+64>: bne    0x8570 <read_password+40>  ; falls nein, ab +40 wiederholen
   0x858c <+68>: b      0x8594 <read_password+76>  ; falls ja, weiter
   0x8590 <+72>: mov    r4, 0
   0x8594 <+76>: mov    r3, 0
   0x8598 <+80>: strb   r3, [r5, r4]           ; 0 (Ende des Passworts) speichern
   0x859c <+84>: pop    {r4, r5, r6, pc}       ; Register wiederherstellen, ENDE
```

Mehr Informationen

Der Quellcode und weitere Daten zu diesem Demo sind auf GitHub verfügbar:

https://github.com/Sjlver/tecday

Ihr dürft mir gerne schreiben:

jonas.wagner@epfl.ch