

UPB – Zadanie 6

Zraniteľnosti v C kóde

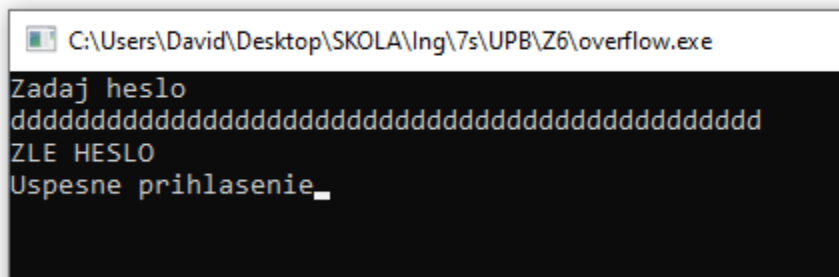
Úlohou je vytvoriť ukážkový príklad zraniteľnej aplikácie, minimálne pre **dve rôzne funkcie** (alebo dva rôzne typy útokov).

Zdroj:

<https://security.web.cern.ch/recommendations/en/codetools/c.shtml?fbclid=IwAR3u2TGpITU5VNV7dOS7ZxrYZzb-4OXRcfx0ARkMu6VfaKtxymurP9WE7gA>

Funkcia : gets()

Daný program simuluje scenár prihlasovania sa používateľa, kde sa po správnom zadaní hesla používateľ prihlási. Funkcia gets() nekontroluje hranice poľa a reťazce väčšej dĺžky zapíše do medzi pamäte, v ktorej je reťazec zapísaný.



```
C:\Users\David\Desktop\SKOLA\Ing\7s\UPB\Z6\overflow.exe
Zadaj heslo
dddddddddddddddddddddddddddddddddddddddddddddddddd
ZLE HESLO
Uspesne prihlasenie.
```

V prípade že zadáme teda dlhší reťazec ako 10 znakov (nastavené) tak dôjde k BUFFER OVERFLOW.

Riešenie?

Funkciu gets() nahradíme za bezpečnejšiu fgets(), kde je vstupom taktiež maximálna dĺžka reťazca.

Funkcia : sprintf()

Funkcia sprintf() taktiež nekontroluje hranice buffra a nie je chránená voči BUFFER OVERFLOW.

```
int main() {
    char buffer[10];
    int check = 0;

    sprintf(buffer, "%s", "This string is too long!");

    printf("check: %d", check); /* Malo vypisat 0, nespravi tak*/

    return EXIT_SUCCESS;
}
```

```
check: 1936269415
[Done] exited with code=0 in 0.658 seconds
```

Riešenie?

Funkciu sprintf() nahradíme za bezpečnejšiu snprintf(), ktorá má dve výhody. Zabraňuje BUFFER OVERFLOW a vracia minimálnu veľkosť buffera potrebnú pre celý reťazec.

```
enum { BUFFER_SIZE = 10 };

int main() {
    char buffer[BUFFER_SIZE];

    int length = snprintf(buffer, BUFFER_SIZE, "%s%s", "long-name", "suffix");

    if (length >= BUFFER_SIZE) {
        printf("Uspesne ochranene");
    }

    return EXIT_SUCCESS;
}
```

```
Uspesne ochranene
[Done] exited with code=0 in 0.654 seconds
```