

I-UPB Zadanie 6

V tomto zadaní sme mali za úlohu vytvoriť zraniteľné aplikácie v programovacom jazyku C/C++. Pre túto úlohu som sa rozhodol vytvoriť separátne zdrojové súbory.

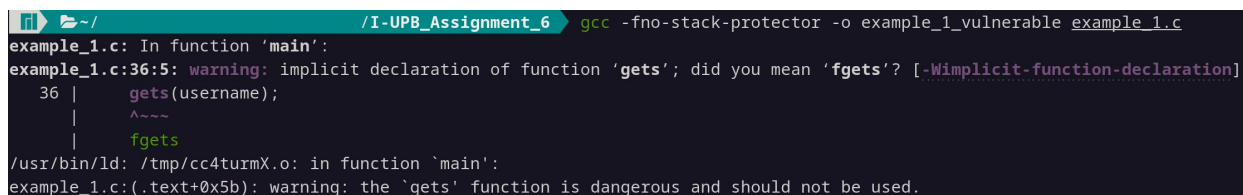
Súbor: example_1.c - gets() vs fgets()

Zraniteľná verzia

V prvom príklade som vytvoril jednoduchú aplikáciu, ktorá načíta používateľské meno a po overení si vykoná privilegované akcie. Pod overením sa rozumie, že či používateľské meno nie je NULL pointer, a privilegovaná akcia je zobrazenie jednoduchej správy. Pre zraniteľnosť som vytvoril príliš krátke (10 znakové) pole. Následne som používal už nepodporovanú funkciu `gets`¹, ktorá je nepodporovaná kvôli zraniteľnosti / nespoľahlivosti ². Ak zadáme menej ako 10 znakov program funguje poriadne, ale akonáhle prekročíme túto hranicu nastane neočakávané správanie (unexpected behaviour). Túto situáciu nazývame buffer overflow. Táto situácia je nebezpečná kvôli tomu, že môže zmeniť vykonanie daného programu. V tomto [článku](#) nájdete stručný popis o tejto zraniteľnosti.

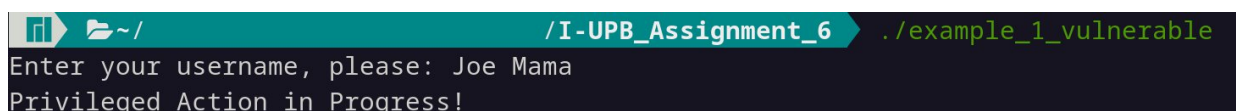
Kompilácia (Napriek tomu, že vyhodí upozornenie kompilácia je úspešná) :

```
gcc -fno-stack-protector -o example_1_vulnerable example_1.c
```



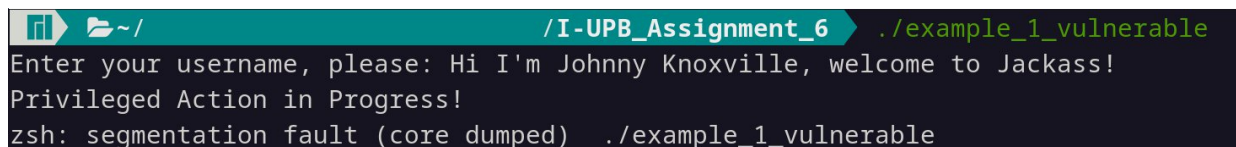
```
/I-UPB_Assignment_6 gcc -fno-stack-protector -o example_1_vulnerable example_1.c
example_1.c: In function 'main':
example_1.c:36:5: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [-Wimplicit-function-declaration]
   36 |     gets(username);
      |     ^~~~
      |     fgets
/usr/bin/ld: /tmp/cc4turmX.o: in function `main':
example_1.c:(.text+0x5b): warning: the `gets' function is dangerous and should not be used.
```

Úspešný beh programu:



```
/I-UPB_Assignment_6 ./example_1_vulnerable
Enter your username, please: Joe Mama
Privileged Action in Progress!
```

Neúspešný beh programu (overflow):



```
/I-UPB_Assignment_6 ./example_1_vulnerable
Enter your username, please: Hi I'm Johnny Knoxville, welcome to Jackass!
Privileged Action in Progress!
zsh: segmentation fault (core dumped) ./example_1_vulnerable
```

V prípade overflowu sme prepísali aj časť kódu v pamäti, a tým pádom, sme ani nemali čakať na validáciu aby sme mohli spustiť privilegovanú funkciu, lebo po buffer overflow už program mal k tomu prístup.

¹ https://www.tutorialspoint.com/c_standard_library/c_function_gets.htm

² <https://stackoverflow.com/questions/1694036/why-is-the-gets-function-so-dangerous-that-it-should-not-be-used>


Bezpečná verzia

V tomto prípade situáciu som riešil rozumnejšie z programátorského hľadiska. Pamäť som alokoval dynamicky a na načítanie používateľského mena som použil už bezpečnejšiu alternatívu `fgets`³. Táto funkcia načítava hodnoty zo tzv. „stream“-ov pričom môžeme pevne definovať maximálne koľko znakov by mala zobrať. Funkcia po načítaní (veľkosť - 1). znaku už ďalej nenačíta. Tiež zastaví aj pri znaku nového riadku. V tomto príklade som stream nastavil na štandardný vstup, a maximálnu veľkosť som nastavil na 10 podobne ako v zraniteľnej verzii.

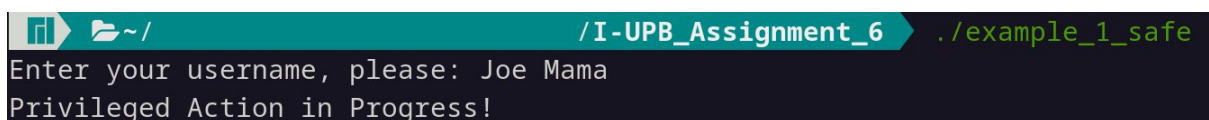
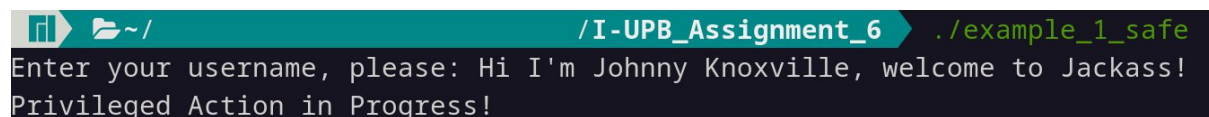
Kompilácia:

Po zakomentovaní zraniteľnej verzie skompilujeme program.

```
gcc -o example_1_safe example_1.c
```

A terminal window with a teal header bar showing the file path ~/I-UPB_Assignment_6. The command `gcc -o example_1_safe example_1.c` has been executed.

Spúšťanie:

A terminal window with a teal header bar showing the file path ~/I-UPB_Assignment_6. The command `./example_1_safe` has been executed. The output shows: `Enter your username, please: Joe Mama` followed by `Privileged Action in Progress!`A terminal window with a teal header bar showing the file path ~/I-UPB_Assignment_6. The command `./example_1_safe` has been executed. The output shows: `Enter your username, please: Hi I'm Johnny Knoxville, welcome to Jackass!` followed by `Privileged Action in Progress!`

V tomto druhom príklade už nenastal buffer overflow. Znak, ktoré sme tam navyše napísali program ani nezobral, zobral len `LENGTH - 1` znakov.

Súbor: example_2.c - printf formátovanie + memory leak

V tomto príklade som vyskúšal aké možné nebezpečenstvá v sebe zahŕňa ak nedefinujeme presne formátový reťazec. Funkcia `printf` ako prvý argument si zoberie formátový reťazec. Keď to nedefinujeme a rovno tam dáme vstup od používateľa, tak v prípade ak používateľ tam pridá formát špecifikátory ako napríklad `%x` alebo `%p`, tak bude už vedieť ktorými pamäťovými miestami pracuje program. Toto nazývame memory leak zraniteľnosť.

Kompilácia :

```
gcc -o example_2_vulnerable example_2.c
```

³ https://www.tutorialspoint.com/c_standard_library/c_function_fgets.htm

Spúšťanie:

[illegible]

Ale prečo je to nebezpečné? Existujú rôzne scenáre. Útočník môže zmeniť hodnotu globálnej / lokálnej premennej, ktorá umožní útočníkovi vykonať chránené metódy, ktoré sú spustiteľné vtedy, ak pred chvíľkou spomínaná premenná má špecifickú hodnotu. Podobne môže zmeniť aj beh programu, s tým, že nastaví vo funkcii printf pointer na danú funkciu. V tomto príklade som sa snažil modifikovať hodnotu globálnej premennej, ale z dôvodu nedostatočných vedomostí som nebol schopný vykonať tento útok. Pre istotu pridám linky na návody, ktoré sú celkom zaujímavé v tomto príklade.

<https://www.youtube.com/watch?v=2HxyGWD1htg>

<https://www.youtube.com/watch?v=0WvrSfcdq1I>

<https://www.youtube.com/watch?v=t1LH9D5cuK4>

Túto zraniteľnosť môžeme celkom jednoducho opraviť. Treba len jednoducho špecifikovať formát vstupu. Akonáhle sme zakomentovali zraniteľnú funkciu a pridali sme novú funkciu, v ktorom sme už špecifikovali, že so vstupom bude pracovať ako textom, tak už memory leak nenastane a už nie sme schopní vytáhať pamäťové miesta z aplikácie.

Kompilácia:

```
gcc -o example_2_safe example_2.c
```

Spúšťanie:

[illegible]

Zdroje:

<https://security.web.cern.ch/recommendations/en/codetools/c.shtml>