

# Problémy na zásobníku, práce s GDB

PV173 Programování v C++11

Vladimír Štill, Jiří Weiser

Fakulta Informatiky, Masarykova Univerzita

22. září 2014

# Problémy na stacku

Aneb co nedělat, proč to nedělat a jak to jde zneužít.  
Postup platný pro x86 (32 bit) architekturu.

# Problémy na stacku

Zdrojový kód

Cíl: spustit funkci **goal**, aniž by byl nastavený parametr **admin**.

# Problémy na stacku – plán postupu

Krok za krokem

- Zjistit slabé místo programu.
- Zjistit jak a kde ovlivnit zásobník.
  - Zjistit rozložení zásobníku.
  - Získat assembler.
  - Potřeba osvojit si práci v GDB.
- Vytvořit postup útoku.
- Útok.

# Problémy na stacku

Zjistit slabé místo.

Je třeba se koukat a bádát :)

Soubor **main.c** a binárka **test**.

# Problémy na stacku

Zjistit jak a kde ovlivnit zásobník.

- Funkce **printf**.
- Formátovací značka **%n**.
  - Zapíše do proměnné, kolik znaků se zapsalo.
  - Použití **printf("blabla%n-blabla", &written);**
- Potřeba zneužít proměnnou **passed**.

# Problémy na stacku

Zjistit rozložení zásobníku.

- Zásobník začíná na vysoké adrese.
- Zaplňuje se směrem k nižším adresám.
- Parametry funkcí jsou vkládány na zásobník.
  - Poslední parametr (nejvyšší adresa)
  - ...
  - První parametr
  - Návratová adresa volající funkce – **eip** registr
  - Předchozí hodnota **ebp** registru
  - Lokální proměnné ... (nejnižší adresa - **esp** registr)
- Parametry po předchozích voláních zůstávají na zásobníku.

# Problémy na stacku

Offset na zásobníku.

- **esp** registr obsahuje aktuální nejnižší adresu na zásobníku.
  - Vrchol zásobníku.
- **ebp** registr obsahuje adresu předchozí **ebp** hodnotu.
  - Označuje horní hranici rámce na zásobníku pro danou funkci.



# Problémy na stacku

Získat assembler.

```
$ objdump -M intel -d test > dump.txt
```

Uloží assembler programu **test** ve formátu Intel (další je AT&T) do souboru dump.txt.

# Problémy na stacku

Potřeba osvojit si práci v GDB.

**\$ gdb -args ./test hodnota**

Pozor – ačkoliv máte k dispozici zdrojový kód, program je přeložen **bez** ladících informací.

# Problémy na stacku

Potřeba osvojit si práci v GDB.

- run – spustit debugovaný program
- quit – ukončit GDB včetně laděného programu
- break {kde} – nastaví breakpoint
  - *jméno funkce* – když jsou k dispozici názvy funkcí
  - *jméno souboru:číslo řádku* – když jsou k dispozici ladící informace
  - \*0x08048450 – když je k dispozici pouze assembler

# Problémy na stacku

Potřeba osvojit si práci v GDB.

- `print /x {co}` – vypíše hodnotu
  - *jméno proměnné*
  - adresu – nutné otypovat, stejná syntaxe jako C
- `examine /8x {kde}` – vypsát paměť
- `next` – další příkaz
- `step` – zanořit se do volání funkce
- `next instruction` – další instrukce, přeskakuje **call**
- `step instruction` – další instrukce

# Problémy na stacku

Potřeba osvojit si práci v GDB.

- finish – vyskočí z funkce
- continue – pokračovat v běhu
- info registers – zobrazí aktuální hodnoty v registrech
- CTRL + C – okamžitě vyvolá breakpoint za běhu

# Problémy na stacku

Vytvořit postup útoku

Vytvořit takový vstup, který donutí funkci **printf** zápis do svého (imaginárního) 3. parametru.

# Problémy na stacku

Útok

```
$ ./test %x%n
```