|  |  |
| --- | --- |
| Názov tímu: | 42 |
| Členovia: | Nicolas Macák  Veronika Szabóová  Petra Kirschová |

Blok 1: Shellcode

# Level 1

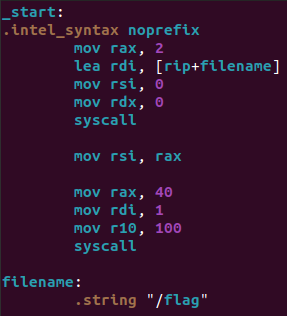
## Obmedzenia:

*Žiadne*

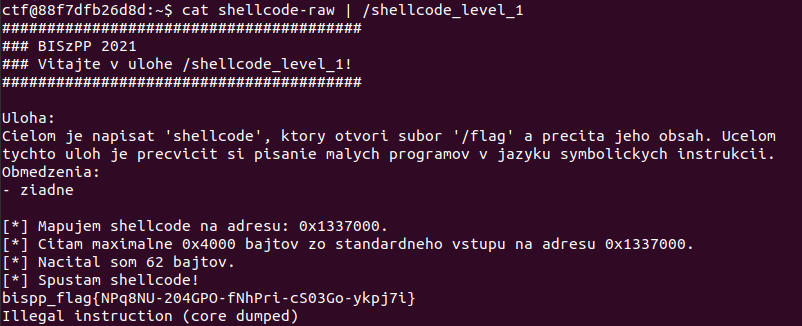
## Riešenie:

Pomocou systémového volania sys\_open sme otvorili súbor /flag a následne sme jeho obsah vypísali na štandardný výstup pomocou systémového volania sys\_sendfile.

***shellcode.s***



***./shellcode\_level\_1***



# Level 2

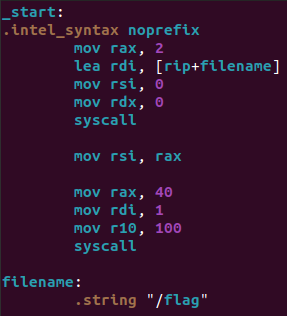
## Obmedzenia:

*Shellcode nesmie obsahovať žiadne fixné adresy*

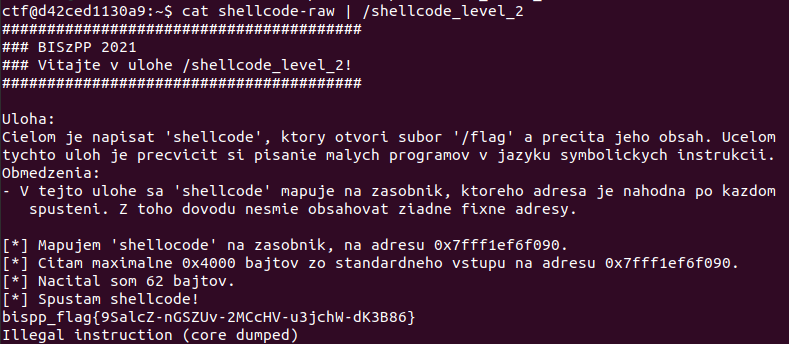
## Riešenie:

Použili sme rovnaký kód, ako v Leveli 1, nakoľko neobsahoval žiadne fixné adresy.

***shellcode.s***



***./shellcode\_level\_2***



# Level 3

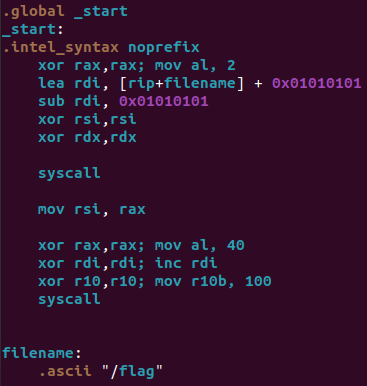
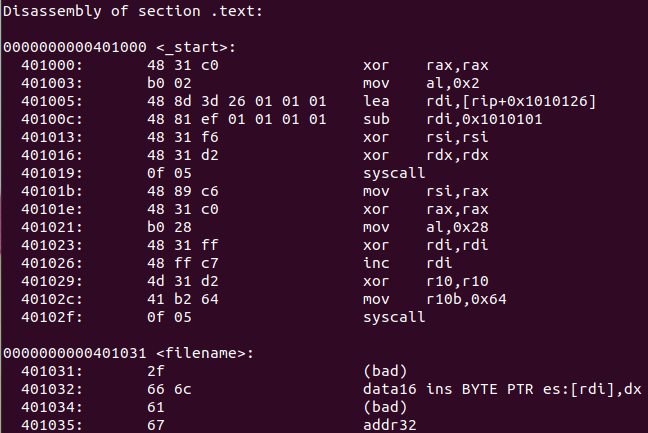
## Obmedzenia:

*Shellcode nesmie obsahovať nulové bajty.*

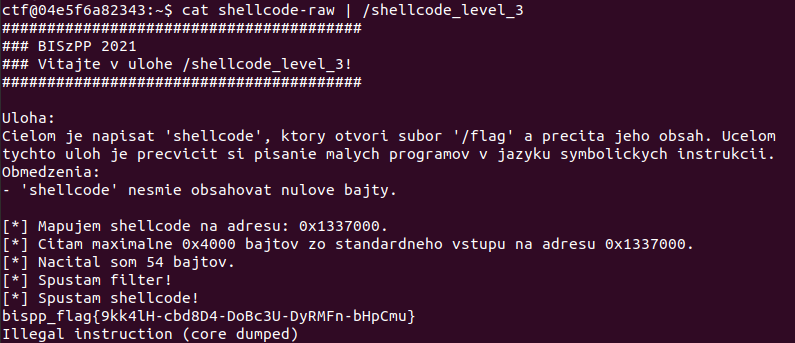
## Riešenie:

V pôvodnom kóde sme zmenili ukladanie nuly pomocou mov na inštrukciu xor, čím sa odstránili prebytočné nulové bajty. Keď sme potrebovali uložiť hodnotu vyššiu ako 0, použili sme registre menšej veľkosti – napr. namiesto použitia celého registra rax sme zaplnili iba jeho spodný bajt v registri al. Pri ukladaní adresy návestia pomocou lea rdi, [rip + filename] sme nulové bajty vyplnili pripočítaním a odpočítaním čísla 0x01010101.

***shellcode.s disassembly***

***./shellcode\_level\_3***



# Level 4

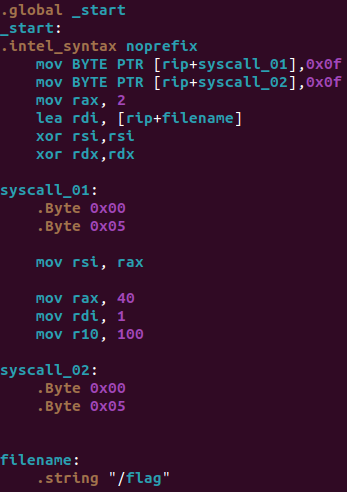
## Obmedzenia:

*Shellcode nesmie obsahovať nasledujúce sekvencie bajtov: 0x0f05 -> inštrukcia 'syscall', 0x0f34 -> inštrukcia 'sysenter', 0x80cd -> inštrukcia 'int'.*

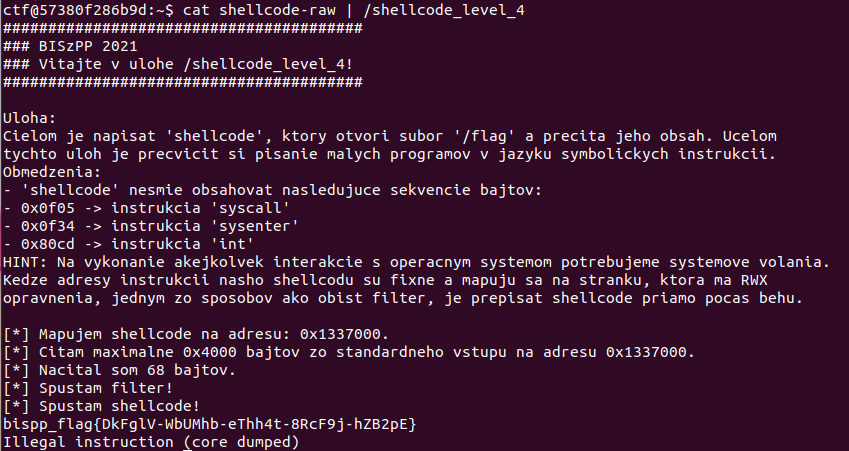
## Riešenie:

Syscall sme zavolali nepriamo pomocou návestí syscall\_01 a syscall\_02, kde boli najskôr uložené bajty 0x00 a 0x05. Bajt 0x00 bol počas behu programu prepísaný na 0x0f, čím sme dostali opcode inštrukcie syscall 0x0f05 a na mieste návestia bolo vykonané systémové volanie.

***shellcode.s***



***./shellcode\_level\_4***



# Level 5

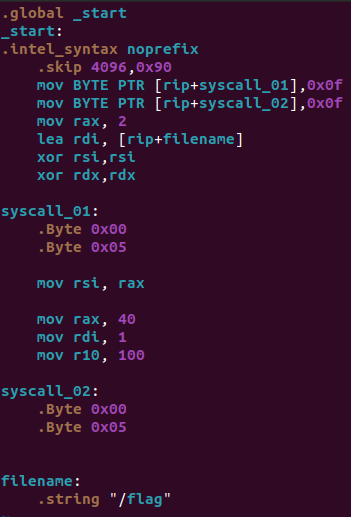
## Obmedzenia:

*Shellcode nesmie obsahovať nasledujúce sekvencie bajtov: 0x0f05 -> inštrukcia 'syscall', 0x0f34 -> inštrukcia 'sysenter', 0x80cd -> inštrukcia 'int'. Oprávnenia na zápis (W) sa odstránia z prvých 4096 bajtov shellcodu.*

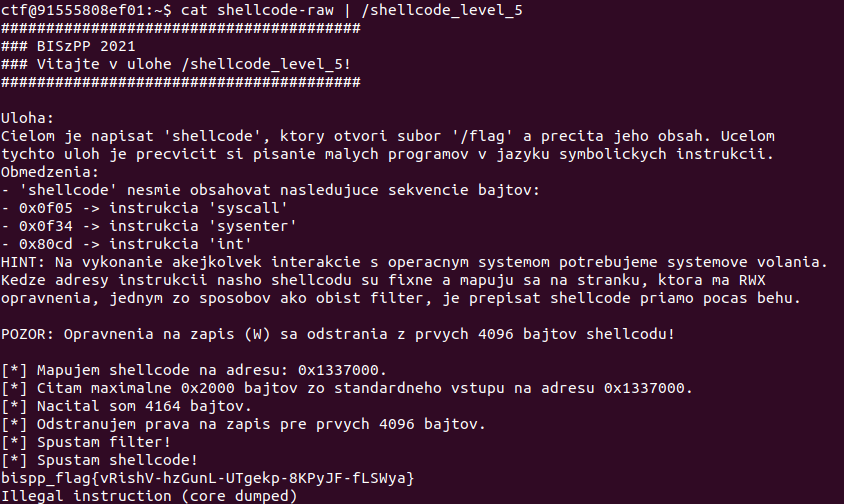
## Riešenie:

Použili sme rovnaký kód, ako v Leveli 4, do ktorého sme na začiatok pridali inštrukciu .skip 4096, 0x90, ktorou sme prvých 4096 bajtov vyplnili inštrukciou nop a tým pádom sa inštrukcie programu začali vykonávať až v oblasti pamäti, ktorá mala oprávnenia na zápis.

***shellcode.s***

******

***./ shellcode\_level\_5***

******

# Level 6

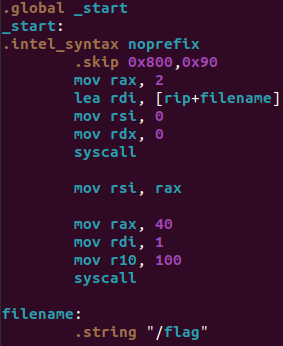
## Obmedzenia:

*Shellcode sa nezačne vykonávať priamo od začiatku, ale skočí na náhodný offset v rozsahu 0x000 až 0x800 bajtov.*

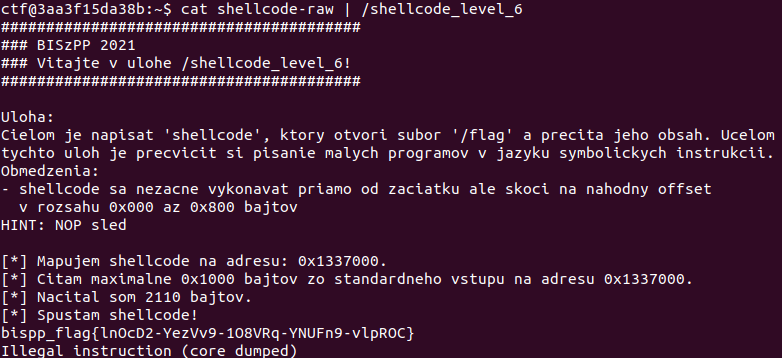
## Riešenie:

Postupovali sme podobne, ako pri Leveli 5. Pomocou .skip 0x800,0x90 sme preskočili prvých 0x800 bajtov a tieto bajty sme vyplnili inštrukciou nop.

**shellcode.s**

****

**./ shellcode\_level\_6**

****

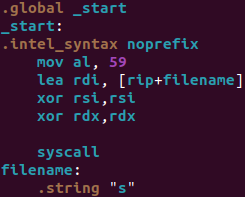
# Level 7

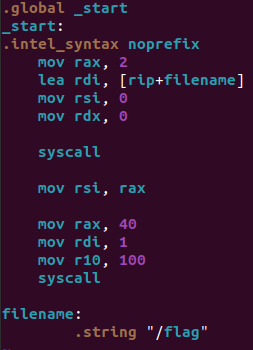
## Obmedzenia:

*Maximálna dĺžka shellcodu je 18 bajtov.*

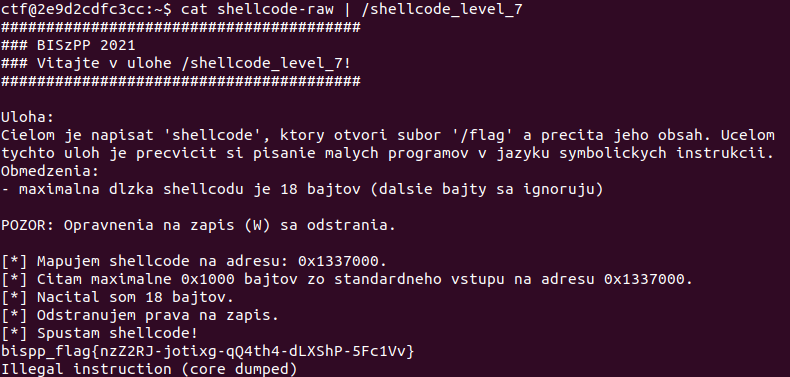
## Riešenie:

Keďže bolo potrebné vytvoriť čo najmenší súbor, shellcode, ktorý sa použije ako vstup do programu, obsahuje iba systémové volanie execve, pomocou ktorého sme spustili shellcode z druhého súboru s.s. Súbor s.s obsahoval rovnaký kód, ako v Leveli 1 a jeho názov sme tiež redukovali, aby nezapĺňal prebytočné bajty.

***shellcode.s s.s***

******

***./ shellcode\_level\_7***

******