Le *shellcoding*

Le plan de la scéance

- Une courte définition du *shellcoding*
- L'assembleur x86 pour débutant
- Les exercices (45 minutes)

Le *shellcoding*

Un *shellcode* est un programme (un code) au format binaire qui, lorsqu'il est exécuté par un autre programme, lance un *shell* (ou effectue tout autre action intéressante pour l'auteur).

Le *shellcoding* est l'acte de développer un *shellcode*.

Ça sert à quoi

Le *shellcoding* est une compétence élémentaire de l'exploitation binaire moderne. Elle ouvre les portes à, notamment:

- l'exploitation des débordements de tableaux;
- la programmation orientée-retour (l'écriture de *ROP chain*);
- l'injection de processus;
- le développement de maliciels avancés.

Une familiarité avec l'assembleur, l'écriture d'exploits et autres maliciels facilite également les tâches d'analyse de binaires, de rétro-ingénierie.

Les types d'exercices

Les exercices de *shellcoding* vous demandent de fournir un *shellcode* qui effectue une action particulière. Généralement, le *shellcode* doit respecter certaines contraintes; e.g. récupérer le contenu d'un fichier ou exécuter une commande, avec un *shellcode* d'une taille inférieure à une certaine limite et/ou aucun octet de valeur `0x2f`.

Les contraintes obligent à devenir créatif et trouver des manières alternatives de coder le programme.

Les particularités du *shellcoding*

Il existe des différences majeures entre un code “normal” et un *shellcode*.

- Un *shellcode* est généralement écris pour s'exécuter de n'importe où en mémoire. Les branchements (`JMP`, `CALL`, etc.) sont relatifs.
- N'assumer **rien** sur la valeur, *a priori*, des registres.
- Il n'y a que le code du programme. Il n'y a pas de section pour les données (de variables globales) et la pile est votre amie, prenez en soin.

De Pep/8 à x86

	Pep/8	x86
registres	4 (`A`, `X`, `SP` et `PC`)	plus de 18
taille du bus	16 bits	16, 32, 64
appels systèmes	`CHARI` et `CHARO`	selon le kernel

La base est sensiblement la même, mais les détails varient.

Quelques ressources

- `site:cs.brown.edu x64 cheat sheet`
- https://x64.syscall.sh/

Les appels systèmes

```
# ...

# write(1, rsp, 13);
mov rax, 1        # sys_write
mov rdi, 1        # 1
mov rsi, rsp      # rsi <- rsp
mov rdx, 13       # 13
syscall           # appel
```

L'utilisation de la pile

Attention au boutisme, les processeurs Intel sont little-endian Et la pile va de haut en bas.

```
# rld\n
    mov     rax, 0x000a646c72
    push    rax
# Hello wo
    mov     rax, 0x6f77206f6c6c6568
    push    rax

# "Hello world\n" est au sommet de la pile...
# ... et est disposée, en mémoire, de bas en haut
# (i.e. correctement)
```

Hello world

Demo

Les outils de *shellcoding*

- l'assembleur (GNU as, NASM, yasm, etc.)
- le désassembleur (GNU objdump, Ghidra, r2, etc.)
- le déboggeur (GNU gdb, LLVM ldb, etc.)
- le *loader* (généralement écris à la main)

L'assemblage et l'extraction du code

```
# assemblage
as -o sc.o sc.s
ld -o sc.elf sc.o

# extraction du code
objcopy -O binary --only-section=.text sc.elf sc.bin
```

Exercice 1 - I'm a shellcoder

> *Fournir un shellcode qui écris "I know how to shellcode" dans le descripteur 4 et retourne. Attention à l'état de la pile au retour.*

https://github.com/antifob/atelier-pwn/blob/main/01-shellcoding/tools/tmpl.gas

```
nc pwn01.f0b.org 9001
```

Exercice 1 - Une solution

Demo

Exercice 2 - Aucun filtre

Ouvrir un fichier, et lire et afficher son contenu.

```
nc pwn01.f0b.org 9002
```

Exercice 2 - Une solution

Demo

Exercice 3 - Les caractères interdits

Ouvrir un fichier, et lire et afficher son contenu. Certains octets sont interdits.

```
nc pwn01.f0b.org 9003
```

Exercice 3 - Une solution

On peut utiliser l'instruction XOR pour altérer les valeurs.

Demo

Exercice 4 - Les caractères interdits et la taille restreinte

Ouvrir un fichier, et lire et afficher son contenu. Certains octets sont interdits et le shellcode a une taille maximale.

```
nc pwn01.f0b.org 9004
```

Exercice 4 - Une solution

On peut utiliser des instructions alternatives, plus petites, pour des effets équivalents et aller charger du nouveau shellcode.

Demo

Conclusion

Le *shellcoding* est une activité créative en raison des multiples contraintes imposées. Elle permet de se familiariser avec la programmation, l'assembleur et l'architecture des ordinateurs. C'est une excellente manière de se préparer à la rétro-ingénierie et et l'exploitation de binaires de toute sorte.

Des cours comme INF217x, INF317x et INF600C vous permettront de pratiquer ces compétences dans un cadre académique. Les CTFs vous permettront d'aller plus loin.

Aller plus loin

- RingZer0 CTF, https://ringzer0team.com/