

Les Registres 86_64

Les registres sont des espaces mémoire situés dans le processeur, dont le rôle est de permettre un accès rapide aux données. Dans un ordinateur, ils constituent la mémoire la plus petite, mais aussi la plus rapide.

Lorsque notre ordinateur effectue des calculs, les données sont stockées dans les registres du processeur. Par exemple, lorsqu'on effectue une addition, le processeur utilise des registres pour stocker temporairement les valeurs des nombres à additionner.

```
ma_fonction addition(a:entier , b:entier){  
    retourner a+b  
}
```

dans l'exemple de la fonction ci-dessus, le processeur stocke la valeur de a dans un registre supposons Z et pour faire l'addition il ajoute b à Z qui contient déjà la valeur de a donc $Z = Z + b$

Types de registres :

Ils existe plusieurs types de registres en x64. Nous avons :

Les registres généraux : globalement se sont des registres qui sont utiliser pour contenir des données (variables, adresses mémoire, ...) :

.RAX (accumulateur) : Principalement utilisé pour effectuer des calculs arithmétiques et logiques. Il est également utilisé pour stocker la valeur de retour d'une fonction.

.RBX (base register) : N'a pas de rôle spécifique, mais peut être utilisé pour stocker des valeurs temporaires ou manipuler des indices de tableaux.

.RCX (compteur) : Généralement utilisé comme compteur de boucle ou pour certaines instructions répétitives (rep, loop).

.RDX (data register) : Utilisé pour stocker la partie haute du résultat lors d'opérations nécessitant plus de 64 bits (par exemple, une multiplication). Il est souvent combiné avec RAX dans ce cas.

.R8 à R15 : Registres généralistes comme RBX, utilisés pour stocker des valeurs temporaires et pour passer des arguments aux fonctions en x86_64 (R8 et R9 étant les 5^e et 6^e arguments).

.RSI (Source Index) & RDI (Destination Index) : Utilisés pour le passage des arguments aux fonctions (RSI pour le deuxième argument, RDI pour le premier argument). Peuvent être utilisés aussi comme RBX

.RFLAGS : est un registre qui contient des valeurs indiquant l'état du processeur lors de l'exécution d'un programme.

Pour le Pwn voici les registres qui sont importants :

Les registres de Pile:

RBP (base pointer)

Le registre RBP est utilisé pour sauvegarder l'adresse du cadre de pile(stack frame) dans une fonction. Le cadre de pile représente l'état de la pile au moment de l'appel d'une fonction.

Grâce à RBP, il est possible de localiser les variables locales et d'y accéder facilement, car il définit la base de la pile pour cette fonction particulière. Il se situe juste avant les variables locales d'une fonction et après le RIP (instruction pointeur)

RSP (stack pointer)

Le registre RSP pointe vers le sommet de la pile, c'est-à-dire l'endroit où la prochaine valeur sera empilée ou dépilée. Lors de l'exécution d'une fonction, il change dynamiquement, car chaque nouvelle valeur ajoutée à la pile fait que RSP diminue (puisque la pile croît vers les adresses basses).

“ Le registre RIP contient l'adresse de la prochaine instruction à exécuter. Il pointe vers l'emplacement mémoire où le processeur doit se rendre pour récupérer la prochaine instruction à exécuter dans le programme.

C'est un registre clé dans le contrôle du flux d'exécution, car il permet au processeur de savoir où il en est dans l'exécution du programme et quelle sera l'instruction suivante à exécuter.

RIP et attaques :

Dans le contexte des attaques comme le buffer overflow, le RIP joue un rôle crucial. Lorsqu'un payload malveillant est écrit dans la pile, il peut modifier le RIP pour rediriger l'exécution du programme vers un code malveillant. Cela se fait souvent en écrivant l'adresse qui contient le début de notre payload dans le RIP, amenant le processeur à exécuter ce code malveillant. Le but de ces attaques est souvent de faire en sorte que le RIP pointe vers une adresse sur laquelle on a le contrôle, afin d'exécuter une série d'instructions bien précises (selon notre objectif et les

morceaux de code disponibles dans le binaire) par le processeur.

Revision #2

Created 5 October 2024 01:01:18 by 0_c@m\$

Updated 20 February 2025 00:52:25 by 0_c@m\$