SOUL-TLAMAQUI

Fighting Evil Since 2008

Entendiendo los BufferOverflow

Lo Necesario

- Procesador basado en la Arquitectura IA32 (x86)
- Compilador para el lenguaje de Programación C y ensamblador para la arquitectura. (gcc y gas)
- Sistema Operativo Basado en el Kernel de Linux.
- Conocimientos básicos de C y gdb.

Marco de Referencia

- En la Arquitectura IA32 (x86) tiene registros de propósito general de 32 bits (4 bytes) *ergo* el tamaño de palabra (“Word”) es de 32 bits.
- La memoria es Direccionable por byte.
- Un entero se almacena en un Word.
- Un *Char* se almacena en un byte.
- Los espacios en memoria se reservan en múltiplos de Word.
- El espacio de direcciones es administrado por el SO.

Endianness

- Se refiere a la forma en que se almacenan los datos multiByte en determinada arquitectura.
- Big-Endian: El byte mas significativo se encuentra mas a la izq.
- Little-Endian: El byte menos significativo se encuentra mas a la izq.

  Ejemplo: 0xFFEE2211

  Big-Endian: {FF,EE,22,11}

  Little-Endian:{11,22,EE,FF}

Buffer

Un buffer es simplemente un bloque contiguo de memoria que mantiene múltiples registros del mismo tipo. Los programadores en C lo asocian con cadenas o arrays.

¿Que es una pila?

La pila de manera abstracta es una estructura de datos que tiene la propiedad de que su primer elemento sera el primero en ser extraído.
Se definen dos operaciones básicas PUSH y POP.

Procesos

Un Proceso es un programa en ejecución
Usualmente esta en memoria principal y se le asigna un “Espacio de Direcciones”.

Se divide en 4 Segmentos Texto o código, Datos, Heap y Stack(pila).

Codigo/Texto

Región del espacio de direcciones que contiene las instrucciones que son ejecutadas por la CPU. Es de Solo Lectura.

Datos

Region del espacio de direcciones que contiene los datos del programa, es aquí donde se guardan los datos declarados dentro del cuerpo principal del programa.

Heap

Parte del espacio de direcciones usado para la asignación de memoria dinámica Usualmente manipulada por funciones de la familia *alloc.

A diferencia del Stack, esta memoria se libera mediante la función free(3).
En IA32 crece hacia las direcciones mas altas

Stack/Pila

Region del espacio de direcciones que permite el paso de parámetros entre funciones. Crece y decrece según los requerimientos del programa, además **guarda las direcciones de retorno a las funciones invocadoras**.
En IA32 crece hacia las direcciones mas bajas.

Desbordamiento

Overflow: Se refiere ala condición en que se exceden los limites de almacenamiento. Como excepción que suele terminar con la ejecución de un programa, aunque no siempre es así, por ejemplo si retorna a una dirección valida dentro del espacio de direcciones.

Heap OverFlow

Es una condición de Buffer OverFlow en la que se sobre-escribe la memoria asignada por unad e las funciones de la familia *alloc. Puede usarse para manipular datos y objetos (C++) dentro del heap.

Stack OverFlow

Es la condición del BufferOverFlow en donde se sobre escribe la memoria en la pila haciendo corrupción de datos y posiblemente de las direcciones de retorno.

Funciones No seguras y su Nueva implementación

1. gets() -> fgets()
2. strcpy() -> strncpy()
3. strcat() -> strncat()
4. sprintf() -> snprintf()

ShellCode

El ShellCode es un conjunto de generalmente programadas en ensamblador y trasladadas a su respectivo código de operación (OPCODE).

ShellCode (Cont.)

```
Char shellcode[]=
        "\xeb\x19\x31\xc0\x31\xdb\x31\xd2\x31\xc9"
        "\xb0\x04\xb3\x01\x59\xb2\x21\xcd\x80\x31"
        "\xc0\xb0\x01\x31\xdb\xcd\x80\xe8\xe2\xff"
        "\xff\xff\x76\x69\x73\x69\x74\x61\x20\x68"
        "\x74\x74\x70\x3a\x2f\x2f\x68\x65\x69\x6e"
        "\x7a\x2e\x68\x65\x72\x6c\x69\x74\x7a\x2e"
        "\x63\x6c\x20\x3d\x29";
```

No Null ShellCode

Cuando se trata de explotar una función de cadena, se tienen que eliminar los caracteres nulos, esto usualmente se hace mediante operaciones XOR y usando registros mas pequeños.