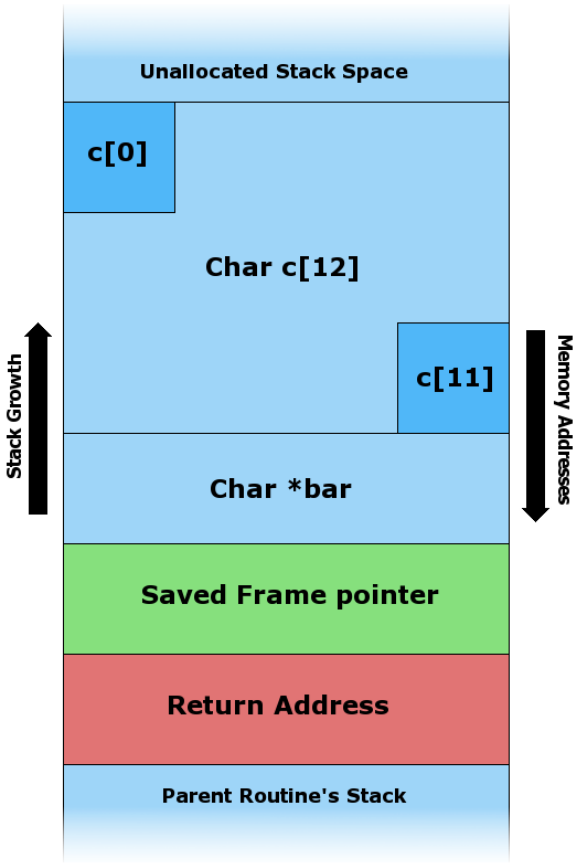
Stack buffer overflow en arquitecturas x86

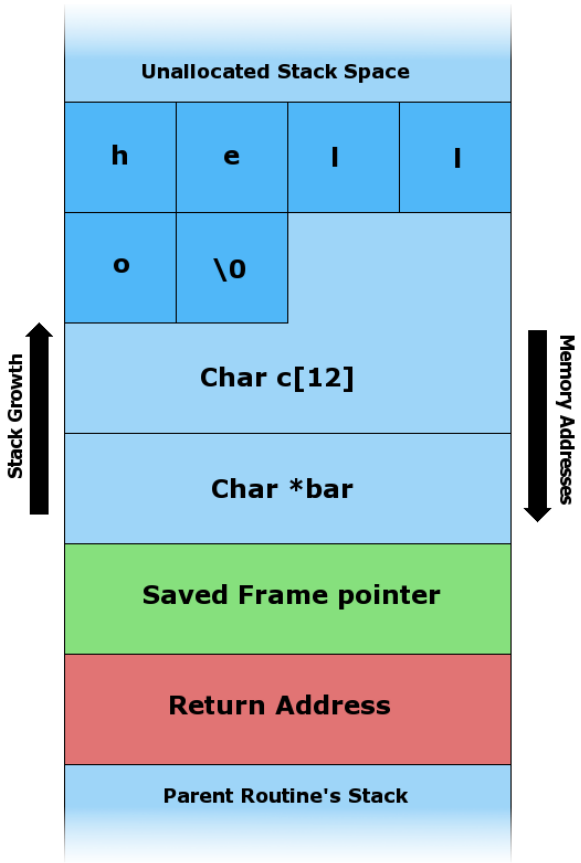
Juan Manuel Torres Palma

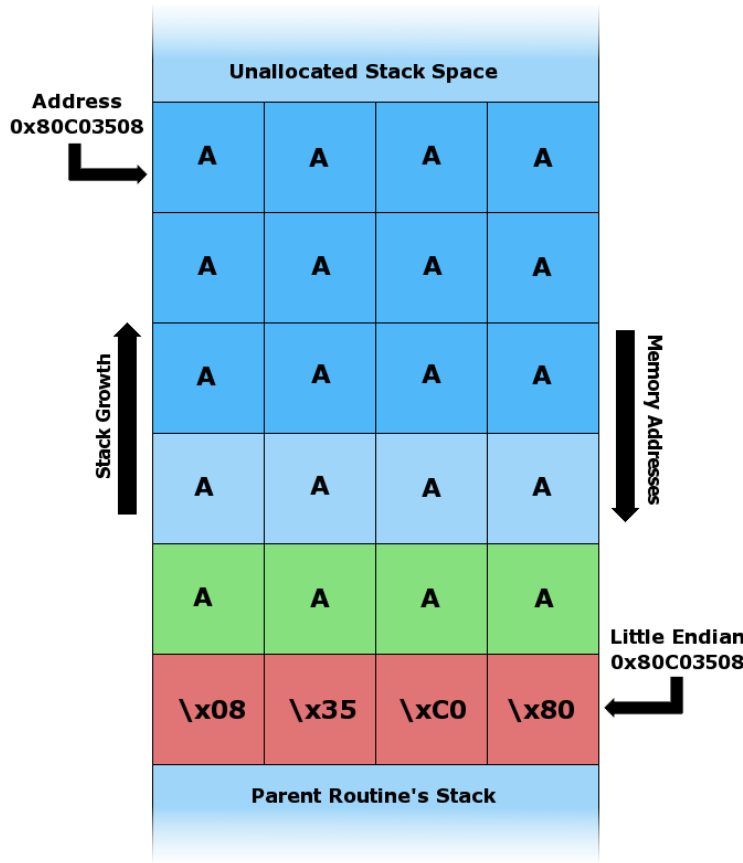
14 de abril de 2014

Índice de contenidos

Estructura de la pila







Mecanismos de defensa

1. Address Space Layer Randomization.
2. Data Execution Prevention.

Intentos de llevarlo a la práctica

1. Método teórico e intrepretación del código.
2. Depuración y fuerza bruta.
3. Combinación de los dos anteriores.

Desensamblado de sbo

sbo.att

```
Dump of assembler code for function main:
0x08048320 <+0>: lea 0x4(%esp),%ecx
0x08048324 <+4>: and $0xfffffff0,%esp
0x08048327 <+7>: pushl -0x4(%ecx)
0x0804832a <+10>: push%ebp
0x0804832b <+11>: mov%esp,%ebp
0x0804832d <+13>: push%ecx
0x0804832e <+14>: sub $0x4c,%esp
0x08048331 <+17>: mov 0x4(%ecx),%eax
0x08048334 <+20>: pushl 0x4(%eax)
```

Desensamblado de sbo(2)

sbo.att(2)

```
0x08048337 <+23>: lea -0x48(%ebp),%eax
0x0804833a <+26>: push%eax
0x0804833b <+27>: call 0x80482f0 <strcpy@plt>
0x08048340 <+32>: mov -0x4(%ebp),%ecx
0x08048343 <+35>: xor%eax,%eax
0x08048345 <+37>: leave
0x08048346 <+38>: lea -0x4(%ecx),%esp
0x08048349 <+41>: ret
End of assembler dump.
```

Obtener dirección del buffer

sbo_ebp.c

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]){
    char buffer[64];
    strcpy(buffer, argv[1]);

    void *p;
    asm("mov %%ebp,%0":"=a"(p));
    printf("ebp%p\n", p);
    return 0;
}
```

Hazlo tú también

1. Ve a mi GitHub y clona el repositorio.
2. Asegúrate de dar los permisos necesarios.
3. Asegúrate de tener instaladas las librerias de 32 bits de gcc, gdb y un intérprete de python 2.
4. Ejecuta make bof.
5. Si funciona no tiene mérito, porque te lo he hecho yo.
6. Si no funciona , ¡enhorabuena! Ahora intenta solucionarlo.