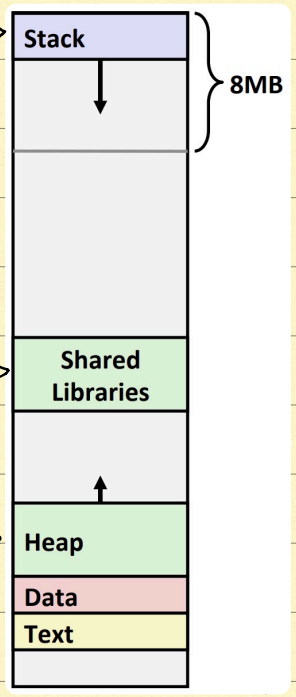
Advanced Topics. (进阶主题).

x86-64 内存布局:

内存结构:

目前只有47位可用, 128TB. 0x7FFFFFFFFFFFFFFF



栈:

运行时栈 (8MB 限制).

本地变量

堆:

动态分配的内存.

文本/共享库

可执行指令.

只读

数据:

静态分配数据

全局、静态变量, 文本常量.

程序分配实例:

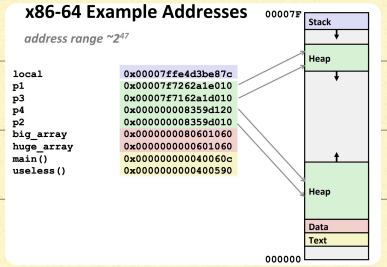
```
char big_array[1L<<24];  /* 16 MB */
char huge_array[1L<<31]; /*  2 GB */

int global = 0;

int useless() { return 0; }

int main ()
{
    void *p1, *p2, *p3, *p4;
    int local = 0;
    p1 = malloc(1L << 28); /* 256 MB */
    p2 = malloc(1L << 8);  /* 256  B */
    p3 = malloc(1L << 32); /*   4 GB */
    p4 = malloc(1L << 8);  /* 256  B */
 /* Some print statements ... */
}
```

注: 分配在堆中的数据, 较大的被分配到了接近栈的位置, 小的被分配到了接近全局变量的位置.



注: 堆会从低端向中间分配.

缓冲溢出问题: 容易造成极大的安全隐患.

- 代码注入攻击.

使用想要的字符填充缓冲区, 这些字符可能会变为可执行指令.

缓冲溢出相关讲解.

缓冲溢出的BUG可以导致受害机执行攻击者的目标代码

蠕虫:
- 可独立运行
- 可以自动机械地繁殖至整个电脑上

病毒
- 将自己植入其他程序
- 不可独立运行.

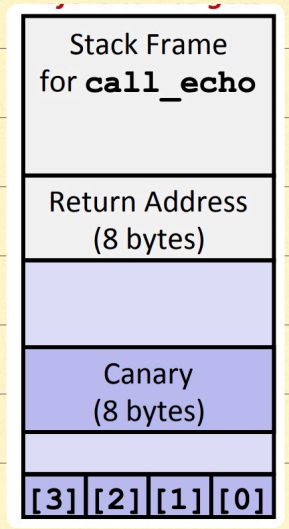
防止攻击的手段

①使用fgets替代gets.(指定长度 ②系统级防护(1:栈随机化ASLR):使每次地址无法预测

(2)标记栈是否可执行.

(3 Canary(金丝雀)防护:在栈上设置标记位,离开函数时进行检测,若被污染则报警

示例:



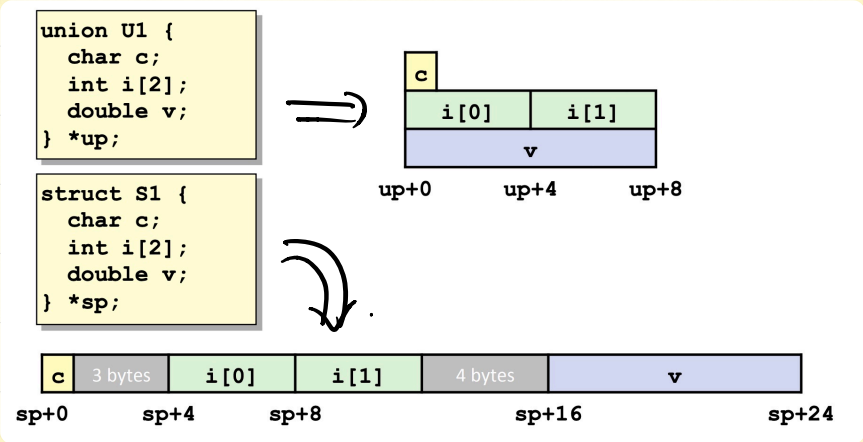
基于返回的程序攻击.(ROP: Return-Oriented Program attack)

核心思想:使用现有的代码,并且调用程序片段来实现想要的结果。

* 无法从完全意义上防护.

联合体(union):

根据最长的域名的内存(只有一个被使用)


union U1 {
char c;
int i[2];
double v;
} *up;
c
i[0]
i[1]
v
up+0
up+4
up+8
struct S1 {
char c;
int i[2];
double v;
} *sp;
c
3 bytes
i[0]
i[1]
4 bytes
v
sp+0
sp+4
sp+8
sp+16
sp+24