he machine advance

1. the memory layout

Stack

Runtime stack (8MB limit)
E. g., local variables

Heap

Dynamically allocated as needed
When call malloc(), calloc(), new()

Shared libraries /text

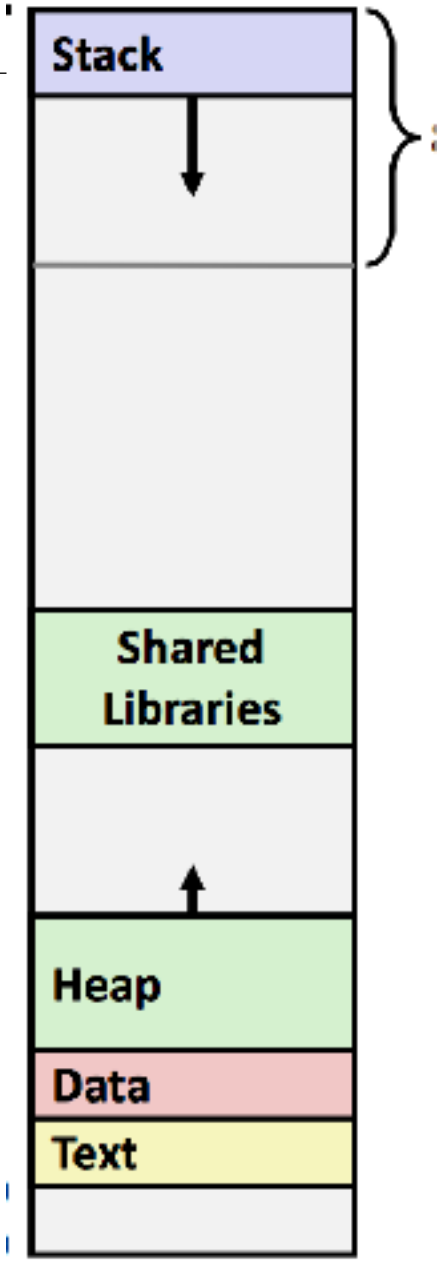
Statically allocated data
E.g., global vars, static vars, string constants

Data

Executable machine instructions
Read-only

相比32位 64 位存在两个heap

高的存高地址的数组，低的存后声明的数组



buffer overflow (缓冲区溢出)

example:

gets function
puts function （对引用数组的边界不进行检查等函数）

reason ：

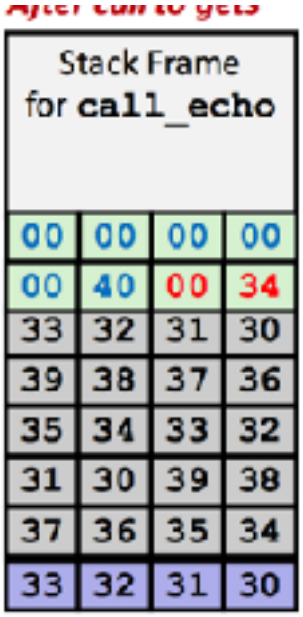
buffer overflow (缓冲区溢出)

example:

gets function
puts function （对引用数组的边界不进行检查等函数）

reason ：
1.c语言没有数组应用没有任何检查

2.虽然定义数组时会开辟多余的空间
（比如说char buff[4]实际会开辟24字节的空间）但是当输入的数据稍微超过边界时，此时的数开始污染寄存器，当前一步汇编代码执行完后，寄存器里储存额的数据将不可信，同理如果数继续增长，则栈中的返回地址也将被污染 。



3. 实例
Code Injection Attacks
通过栈的溢出污染，修改返回地址， 进而使程序跳转到攻击者想要执行的函数 。

4.how to avoid buffer overflow

应用功能相近但是安全性较好的函数 fgets instead of gets
strncpy instead of strcpy
Don't use scanf with %s conversion specification
Use fgets to read the string
Or use %ns where n is a suitable integer

5.System-Level Protections

1.Randomized stack offsets

开始分配一部分空间，运用的时候并不全部占有，每次运行时修改栈的地址

2.Setting Up Canary

在栈中设定一个值， 每次运行的时候检查值是否修改过 ， 来判断下一步是否运行。

chapter 3 machine level

1.
数据传送指令 目标和源值都不能是储存器
2.