UAF漏洞

原创

Tinjou 于 2022-02-27 16:54:39 发布 61 收藏
文章标签： c语言 c++ 开发语言

本文链接：https://blog.csdn.net/m0_60584218/article/details/123163895

UAF漏洞

Use After Free

我们可以直接从字面上翻译它的意思：使用被释放的内存块。其实当一个内存块被释放之后重新使用有如下几种情况：

内存块被释放后，其对应的指针被设置为NULL，再次使用时程序会崩溃 内存块被释放后，其对应的指针没有被设置为NULL，在它下一次被使用之前，没有代码对这块内存块进行修改，那么程序有可能可以正常运转 内存块被释放后，其对应的指针没有被设置为NULL，但是在下一次使用之前，有代码对这块内存进行了修改，那么当程序再次使用这块内存时，就很有可能出现问题。

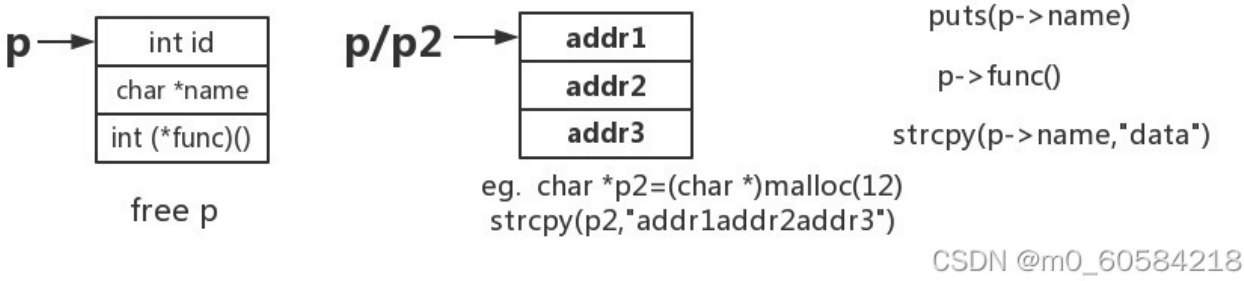
Dangling pointer即指向被释放的内存的指针，通常是由于释放内存后，未将指针置为NULL。

UAF原理：对Dangling pointer所指向内存进行use，如指针解引用等。 **利用思路**：将Dangling pointer所指向的内存重新分配回来，且尽可能使该内存中的内容可控（如重新分配为字符串）

举例

```
typedef struct
{
int id;
char *name;
int (*func)() //函数指针，可以理解为类里面的方法
};
```

假设有上述这样的一个结构体指针p：



在释放掉p之后，没有将p置NULL，所以p变成Dangling pointer，再通过重新分配，再次拿到p之前指向的这段地址空间。

之后，通过strcpy(p2,"addr")，或者其他方式，向这段地址空间写入新数据。

然后当我们通过其他函数，再次使用p指针，就会造成无法预料的后果，因为**此时p指针指向的内存包含的已经是完全不同的数据**

任意地址读：puts(p->name)---------->puts(char*(addr2))

任意地址写：strcpy(p->name,data);------>strcpy((char *)(addr2),data)

控制流劫持：p->func()---------->call addr3

参考

好好说话之Use After Free_hollk’s blog-CSDN博客

CTF Pwn中的 UAF 及 pwnable.kr UAF writeup - 知乎 (zhihu.com)