信安21XX班

MTX

U2021XXXXX

逆向分析报告

**一．具体实验过程**

1.首先利用反汇编，研究输入部分的相关函数，发现使用了LOBYTE，存在整数溢出漏洞。

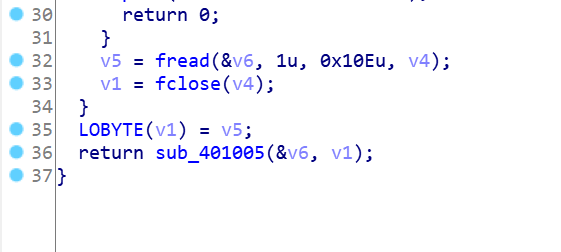


图1 整数溢出漏洞代码

2.跟踪使用了LOBYTE的变量传入的函数，发现使用了strcpy函数，存在栈溢出漏洞。

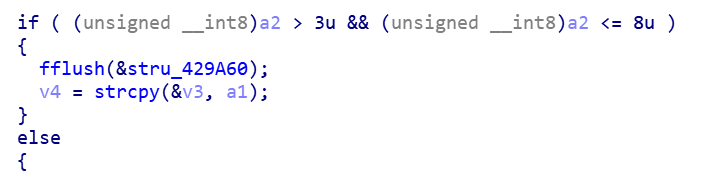


图2 栈溢出漏洞代码

3.通过查找String的方法找到我们需要的Success字符串，找到对应的代码。

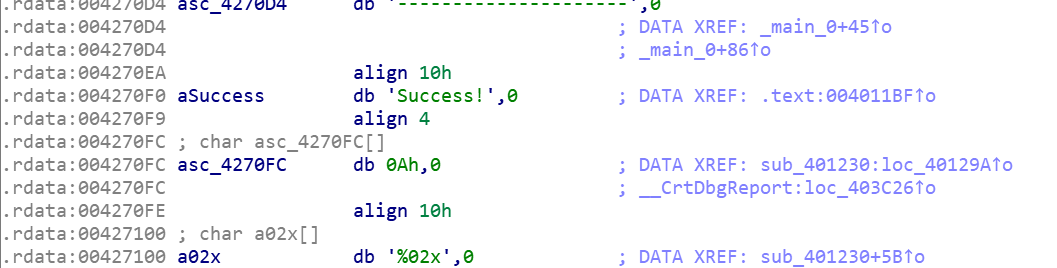


图3 Success字符串

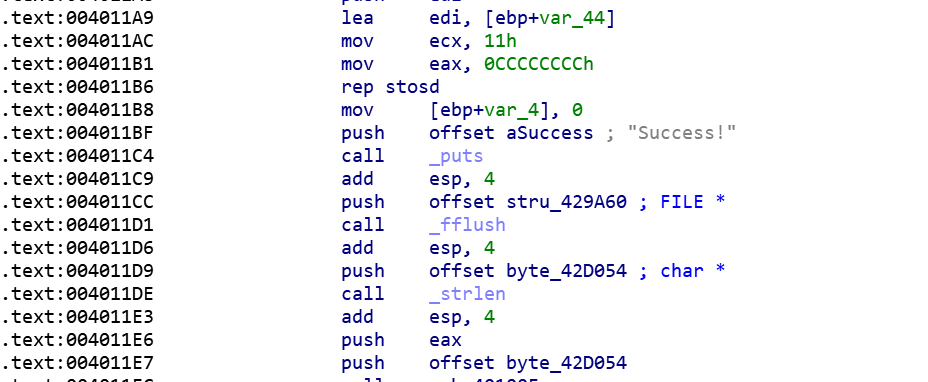


图4 输出Success字符串代码段

4．反汇编后确定该函数即为我们所需要的输出Success的函数，并且跟踪sub\_40100F函数可以发现其为生成Flag的函数，于是我们准备利用栈溢出漏洞来跳转到该函数。

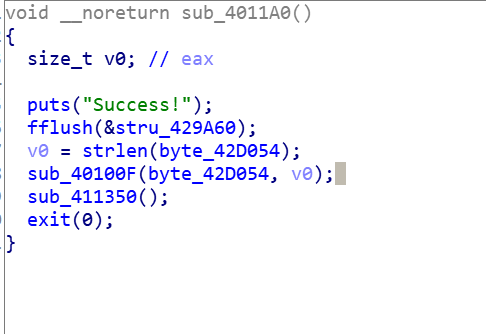


图5 反汇编后的输出Success函数

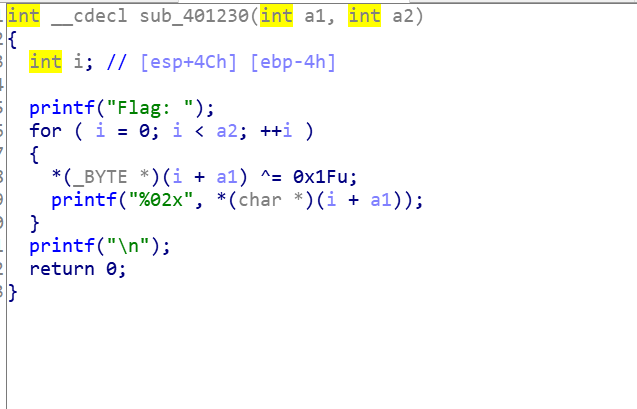


图6 输出Flag函数

5. 在调试模式下，我们利用之前发现的整数溢出漏洞，构造一个长度大于 256 + 3 但小于等于 256 + 8 的文件。在设置了 strcpy 前的断点后，我们检查了栈的情况。通过观察 EBP 的值，我们得知 0x19FC54 存放的是 EIP 返回地址。分析栈溢出的覆盖情况时，发现从 0x19FC44 开始进行了覆盖，或者通过查看汇编代码得知编译器为字符串预留的空间，确定了变量地址为 0x19FC44。随后，我们开始构造填充数字，前 12 字节用于填充字符串的空间，13-16 字节用于填充栈中 EBP 的位置的内容，17-20 字节用于填充栈中 EIP 的内容。

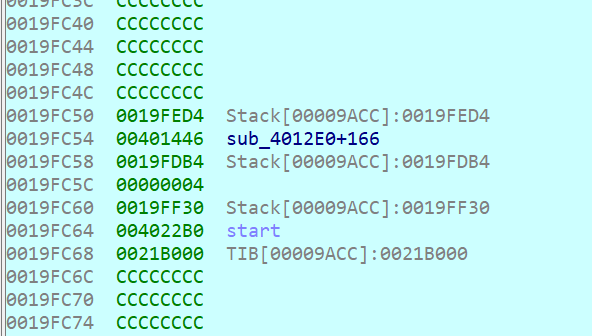


图7 执行Strcpy前栈中内容

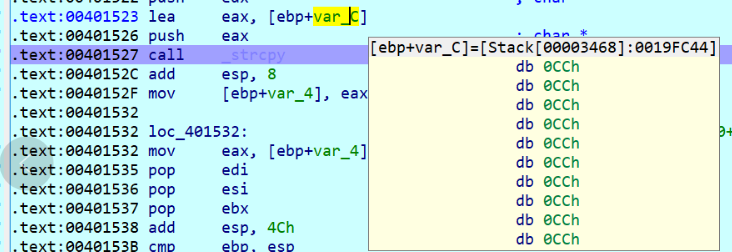


图8 字符串在栈中地址

6. 我们需要调整的函数地址为 0x004011A0，因此我们构造了一个填充内容为 260 字节全为字符 '1' 的文件。为了符合小端模式，其中 17-20 字节填充的值为 0xA0， 0x11， 0x40， 0x00。在调试中观察，执行完 strcpy 后，原来存放 EIP 返回地址的位置已经变为 0x004011A0。接下来，继续执行程序，就可以成功跳转至能输出 "Success" 的函数。

图9 输出Success函数地址

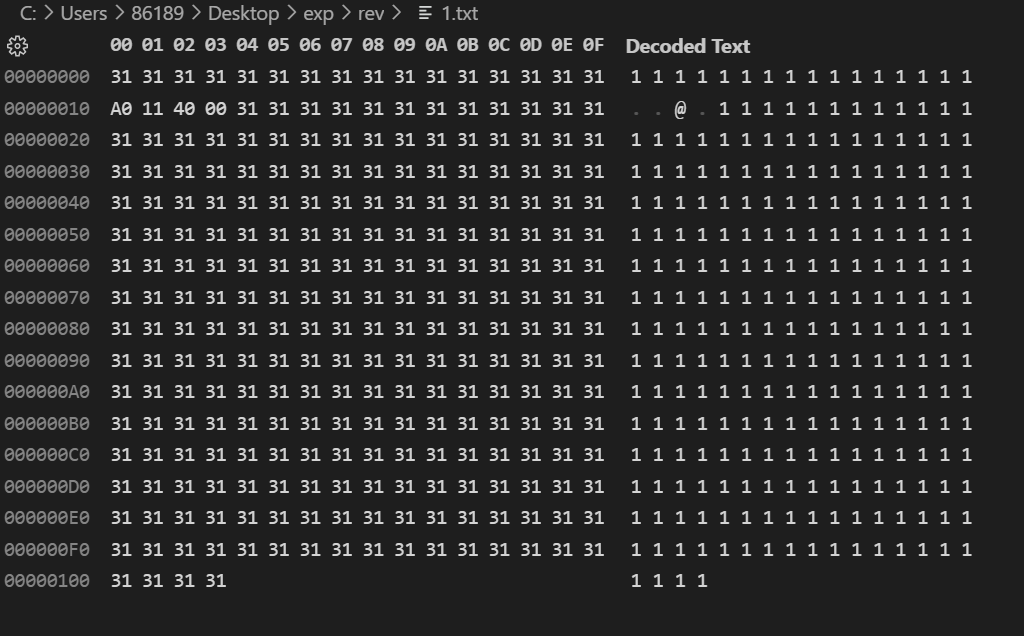


图10 构造文件填充内容

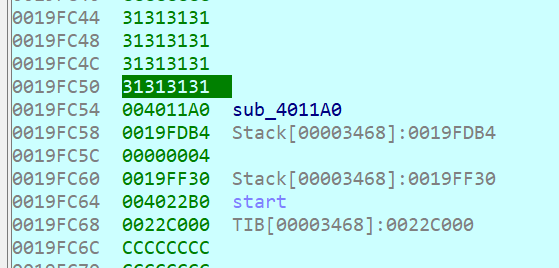


图11 使用构造文件利用栈溢出后栈中内容

7.执行效果及Flag如下

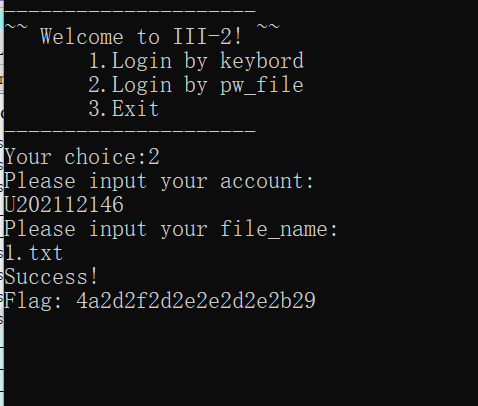


图12 利用漏洞获得flag