**war-ftp1.6.5溢出实验**

**实验环境：**

VMware16；Windows XP；war-ftp1.6.5

x32dbg（尝试使用OllyDBG和WinDBG都出现了问题，于是选择了这个）

GIT链接：https://github.com/Fattysand/warftp\_overflow.git

**实验步骤：**

用x32dbg打开war-ftp1.6.5并运行，打开online模式

使用python作为输入脚本，查看EIP是否改变，如果改变，就证明溢出了

先用500个A尝试能否溢出，发现EIP已经被修改（ASCII码为41，是A）

图形用户界面, 文本, 应用程序

描述已自动生成

于是尝试定位是多少，发现是E（ASCII码为45），确定是401-500这一区间

图形用户界面, 文本, 应用程序

描述已自动生成

以此类推，不断尝试十位和个位，得到精确溢出数值，十位是J（ASCII码为4A），确定是481-490之间

图片包含 图形用户界面

描述已自动生成

个位是GHIJ（小端序的4A494847），G的ASCII码为47，所以填充字节应为485个，486-489为EIP的地址。

图形用户界面, 文本

中度可信度描述已自动生成

尝试构造输入使EIP成为我们想要的数据，成功取得11451419

图形用户界面, 文本

描述已自动生成

众所周知， EIP寄存器中的值为下一个要运行的指令，且ESP与EIP起始地址相隔8个字节。所以可以在在EIP中填入JMP ESP指令，并同时控制ESP的值（填入指令）就可以了。

于是再用4个字节来填充EIP和ESP中间相隔的字节。

验证思路，发现ESP果然被操控（被填入了4个C），EIP也是4个B。

图形用户界面, 文本, 应用程序

描述已自动生成

既然思路有了，就可以将B替换成JMP ESP指令，将C替换成shellcode执行。

图形用户界面, 应用程序, Word

描述已自动生成

随便找一个JMP ESP指令7635C3FB（找一个靠中间的也许靠谱一点，但其实都一样）

然后发现程序卡死，没能成功跳转

图形用户界面, 文本, 应用程序

描述已自动生成

去查看ESP寄存器信息，地址在00AEFD48，发现48后的值都不太正确

图片包含 表格

描述已自动生成

从00AEFD58开始，是buffer第二行第三个字节往后的31781503，所以前面还需填充16个字节，为了保证成功，额外再填充16个NOP指令（\x90），即使超出了也可以滑入shellcode保证执行

文本

描述已自动生成

再次开启warftpd，再次使用python脚本，成功召唤出计算器

电脑萤幕的截图

描述已自动生成

结果符合预期，实验成功。