## **Modbus Slave缓冲区溢出漏洞CVE-2022-1068分析与复现**

**漏洞概述**

Modbus Slave是一个模拟工业领域通信协议Modbus从站的上位机软件，主要用于测试和调试Modbus从设备。该软件7.4.2以及以前的版本，在注册时，未对注册码长度进行安全检查，存在缓冲区溢出漏洞，导致软件崩溃。

**影响范围**

Modbus Slave 7.4.2以及以前的版本

**复现环境**

操作系统：Win7 sp1 32位操作系统。

分析工具：IDA、OD、Windbg、Procexp。

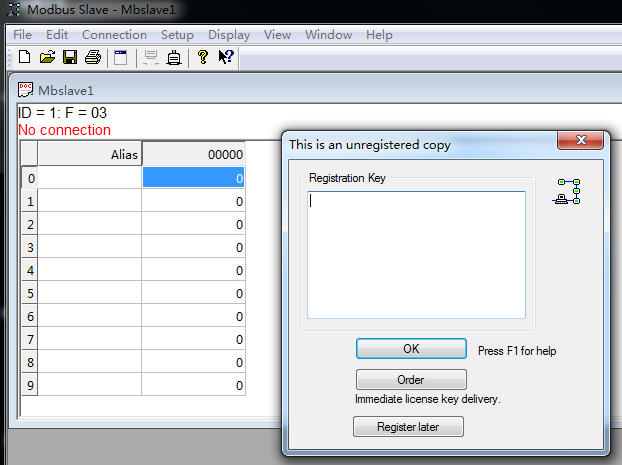
**漏洞分析**

分析时使用的版本为7.1.2.1357。首先安装modbusslavesetup32bit\_7.1.2.1357.exe程序，默认安装后主程序路径为C:\Program Files\Modbus Tools\Modbus Slave\mbslave.exe。

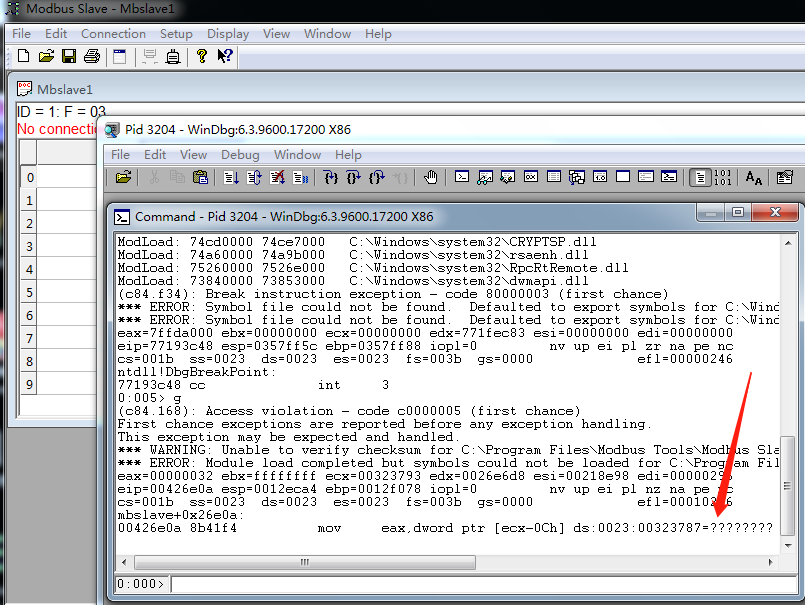
根据exploit-db上已公开的poc.py生成Modbus.txt。Poc比较简单，Modbus.txt的内容为736个字符“A”，如下图所示：



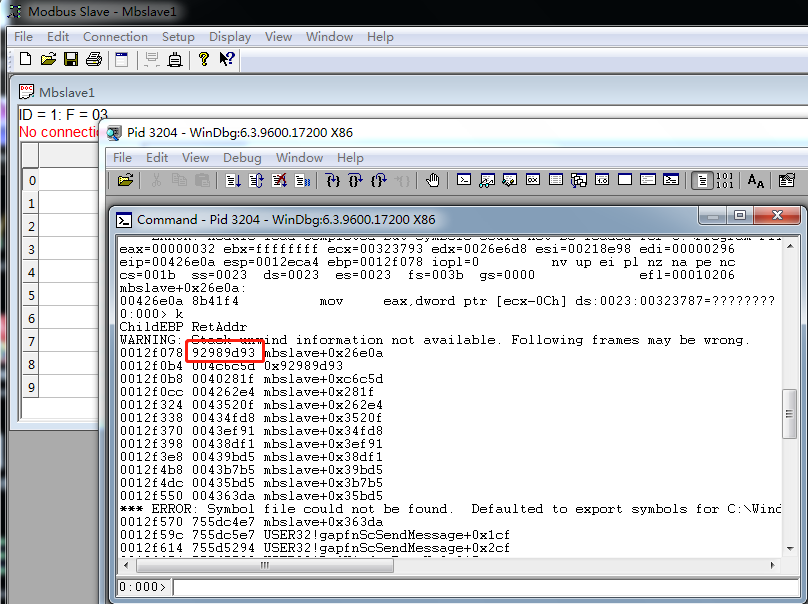
使用Windbg调试器调试主程序mbslave.exe，然后点击mbslave.exe菜单栏的“Connection”按钮，在弹出的菜单中选择“Connect”，此时将弹出注册对话框，如下图所示：



复制poc文件Modbus.txt中的字符串内容，粘贴到注册对话框中，最后点击“OK”按钮，验证是否触发漏洞。如果poc验证成功，Windbg将捕获到异常，如下图所示：

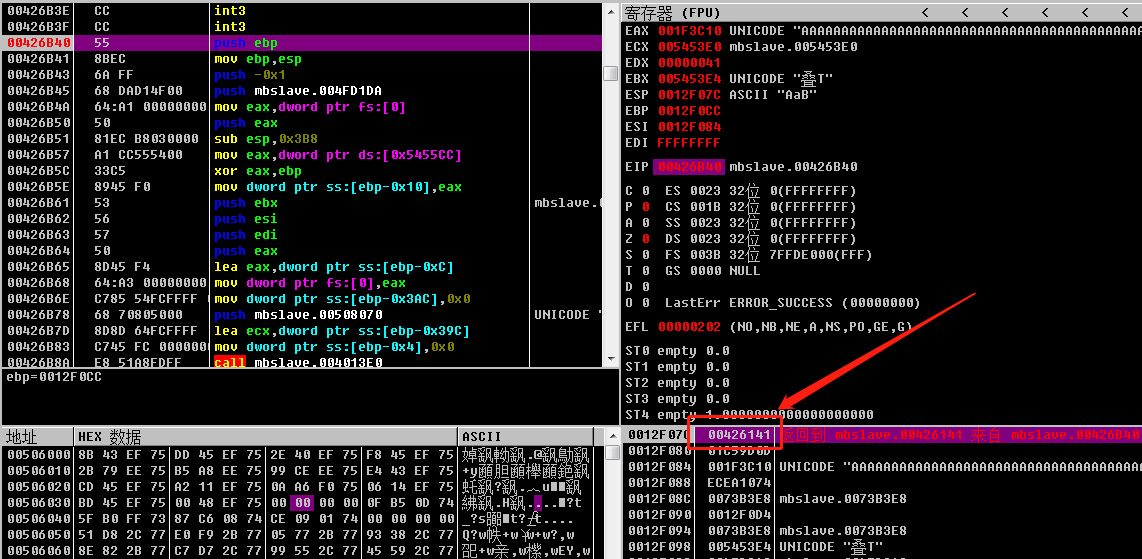


在Windbg中使用k命令查看调用函数调用栈，发现栈中函数的返回地址已被覆盖为一个非常规值0x92989d93，说明发生了栈溢出。如下图所示：

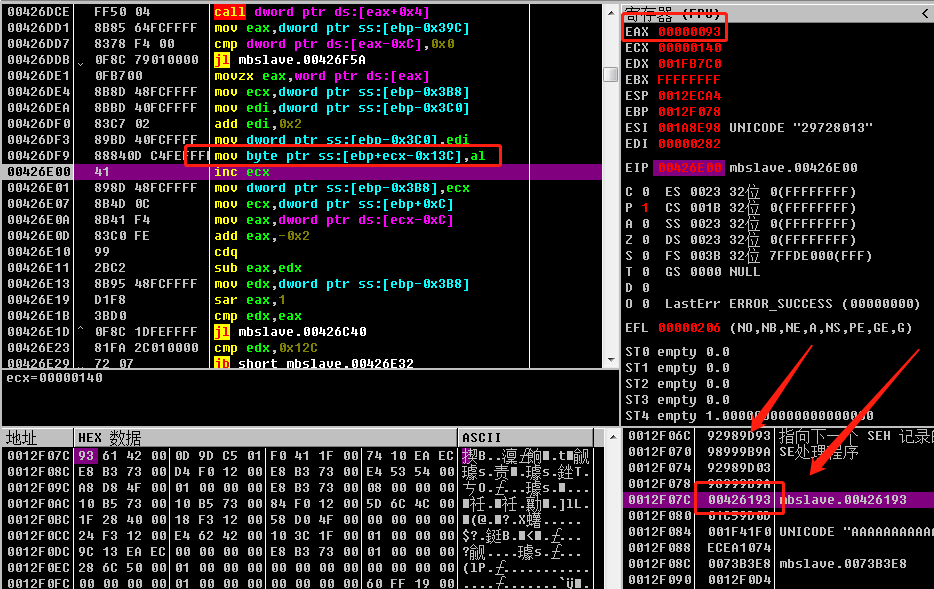


接下来查找栈溢出的原因。

为更方便和直观，使用调试器OD重新调试程序mbslave.exe，在异常发生地址所在函数0x426B40下断点。再次将poc内容复制到注册对话框中，触发漏洞，调试OD将停止在断点处。此时，栈中的函数返回地址为正常值0x426141，如下图所示：

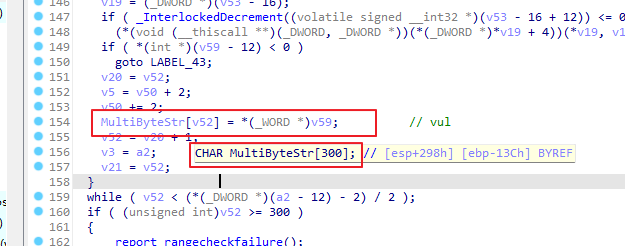


为了定位栈中函数返回地址被覆盖的代码，需要对保存函数返回地址的栈地址0x12f07c下硬件写断点，然后继续运行。程序将停止在0x426e00代码处，此时，栈中的返回地址已被覆盖为0x426193，如下图所示：



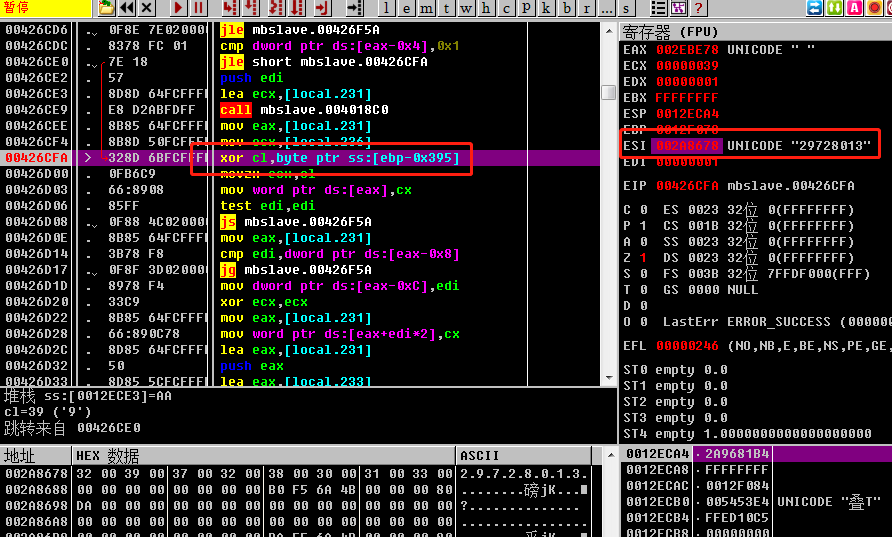
根据上图，可以看出，正是地址0x426DF9处的汇编代码，将EAX最低字节AL的值0x93覆盖栈中的函数返回地址的最低字节，说明导致栈溢出的代码就是该条汇编代码。当前除了覆盖栈中返回地址，栈溢出还覆盖了栈中结构化异常处理地址（SEH）等内容。

在反编译工具IDA中可以看到漏洞产生处的伪指令为 MultiByteStr[v52] = \*(\_WORD \*)v59，此时缓冲区的大小为300个字节，长度太小导致了溢出，覆盖了栈中函数返回地址和正常变量值，导致后续指令执行错误，使程序崩溃。如下图所示：



从上述的分析可知，溢出产生时，覆盖的数据并不是poc中的”A”字符。那么覆盖的数据是否来源于poc中，经过运算后的数据呢？

在IDA中查找数据来源，结合OD中动态调试分析，发现覆盖的数据是地址0x00426CFA处的代码运算后的结果。这是一条异或运算的汇编代码，按字节异或，如下图所示：

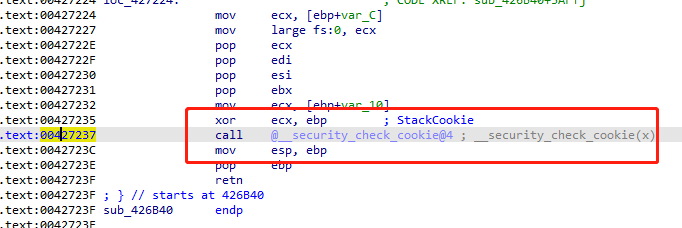


异或的源数据是poc中字符串按两位一组转化为十六进制的值，异或的KEY为“97280132”。比如第一轮循环时，将poc中的前两位字符串“AA”转换为0xAA，然后与KEY的第一位字符“9”，十六进制为0x39进行异或，最后将异或的结果放到栈中缓冲区MultiByteStr[v52]进行保存。如果poc中的字符串个数大于600个字符（600=300\*2，poc中为736个字符“A”），将发生栈溢出，可以覆盖栈中SEH地址，函数返回地址和局部变量等，导致程序崩溃。

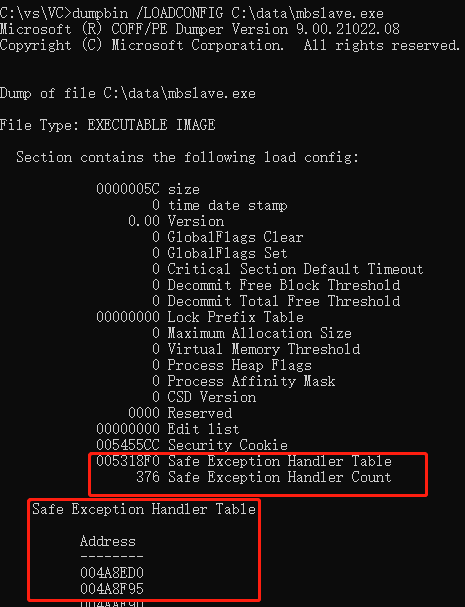
既然发生了栈溢出，并且数据可控，那么是否能执行任意shellcode呢？

**漏洞利用**

常见的栈溢出利用可以利用被覆盖的函数返回地址执行任意shellcode，即漏洞发生后，覆盖了栈中函数返回地址，当当前函数执行完毕后，将跳转到函数返回地址执行代码。但是该程序中开启了Security Cookie栈保护安全机制，位于0x00427237地址处，在函数返回前，Cookie检验失败，无法利用该方法。



除此之外，溢出还覆盖了栈中结构化溢出处理（SEH）函数地址，当异常发生时，将跳转到异常处理函数执行代码。但是该程序中开启了Safe SEH安全保护机制，简单来讲，不是预定义的异常处理函数地址，不予执行。使用 dumpbin /loadconfig mbslave.exe命令可以查看Safe SEH 表，如下图所示：



既然开启了Safe SEH，是否有办法绕过呢？常见的绕过方法如下:

1.攻击返回地址绕过

该程序开启了Security Cookie，无法绕过。

2.虚函数绕过

溢出覆盖范围有限，无法覆盖虚函数。

3.从堆中绕过，将shellcode布置在堆中 ，SEH处理函数指向这个地址即可

该程序开启数据执行保护（DEP），无法将数据作为代码执行。

4.利用未启用Safe SEH模块绕过，可以把这个模块的指令作为跳板，去执行shellcode

该程序没有未启用Safe SEH的模块。

5.加载模块之外的地址绕过，内存中有一些Map类型的映射文件，在这些文件中找到跳板指令覆盖SEH处理函数地址即可绕过。

该程序开启数据执行保护（DEP），无法将数据作为代码执行。

总的来说，该程序在测试系统win7 sp1上，无法绕过Safe SEH执行shellcode。如果使用低版本的系统，比如Windows XP低版本系统，可以实现执行任意shellcode。因为Safe SEH是Windows XP sp2以及之后的版本中加入的安全特性。

**参考链接**

https://www.exploit-db.com/exploits/50536

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-1068