软件漏洞分析与防范

大作业报告

（ 2020 / 2021学年 第1学期）

题 目 Microsoft Windows

Media Player winmm.dll MIDI

文件堆溢出漏洞调试分析

**专 业 信 息 安 全**

**学 生 姓 名 茹兴奥**

**班 级 学 号 信安8班 B19030824**

**指 导 教 师 沙 乐 天**

**日 期 2022/1/10**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **评分内容** | **具 体 要 求** | | | **总分** | **评分** | |
| **准备情况** | 能独立查阅文献资料及从事其他形式的调研，能较好地理解任务并提出实施方案，有分析整理各类信息并从中获取新知识的能力。 | | | 10 |  | |
| **功能完成情况** | 论证、分析、设计、实现、实验正确合理。 | | | 30 |  | |
| **报告撰写质量** | 结构严谨，文字通顺，用语符合技术规范，图表清楚，书写格式规范，符合规定字数要求。 | | | 30 |  | |
| **答辩** | 内容全面，紧扣课题，突出重点，语言简练准确，演示过程顺利。 | | | 30 |  | |
| **总评分** |  | | | | | |
| **简短评语** | xx同学在这次大作业中通过PoshC2渗透软件、利用powershell环境，完成针对Windows 10操作系统的访问控制机制的理解、越过以及渗透测试，选题与课程内容结合较为紧密。  该同学准备充分，功能完成情况优秀，报告内容充实、格式较规范；答辩时思路清晰，演示熟练。  综上所述，大作业成绩评定为xx分。  **教师签名：**  **2022年 1 月 10日** | | | | | |
| a.讨论课成绩(30%) | | b.大作业成绩（70%） | 期末成绩（百分制） | | |
|  | |  |  | | |

1. **背景介绍**

1. 项目简介

2012 年 2月，地下黑产中流行着一款国产名为蜘蛛漏洞的攻击包 —— “Zhi-Zhu Exploit Pack”，该工具包含 5 个漏洞，都是在当时比较流行的漏洞，涵盖了 Flash、IE 等产品，其中就包含 CVE-2012-0003，此国产漏洞利用包与国外有一定的差别；国外通常用于构建僵尸网络或者窃取重要的情报，而 “蜘蛛” 这款漏洞攻击包主要用于盗取游戏账号，由于该漏洞过于特殊，所以在分析和利用的时候难度是相当大的，本次实验就复现这个漏洞的发现，漏洞的利用过于复杂不做表达。

2.MIDI文件格式

想要理解漏洞的成因，首先需要了解什么是 MIDI 文件格式：MIDI 全称 Musical Instrument Digital Interface，是一种乐器数字接口，说白了就是一种音频格式（通过winmm.dll 链接库解析这种格式的文件之后可以播放出音乐）总体来说 MIDI 可分为 “头块” 和若干多的 “音轨块”

1. **实验工具**

虚拟机：Windows XP sp3

调试工具：Windbg（Windbg32，Windbg64）、OD、IDA、C32Asm

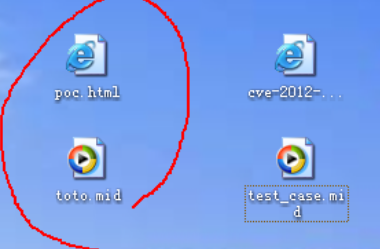
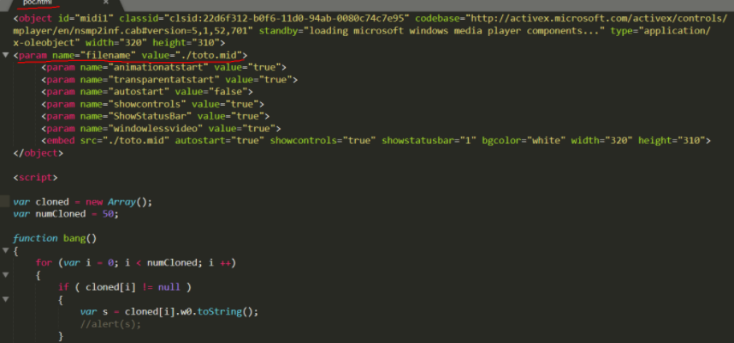
漏洞程序：IE6 浏览器

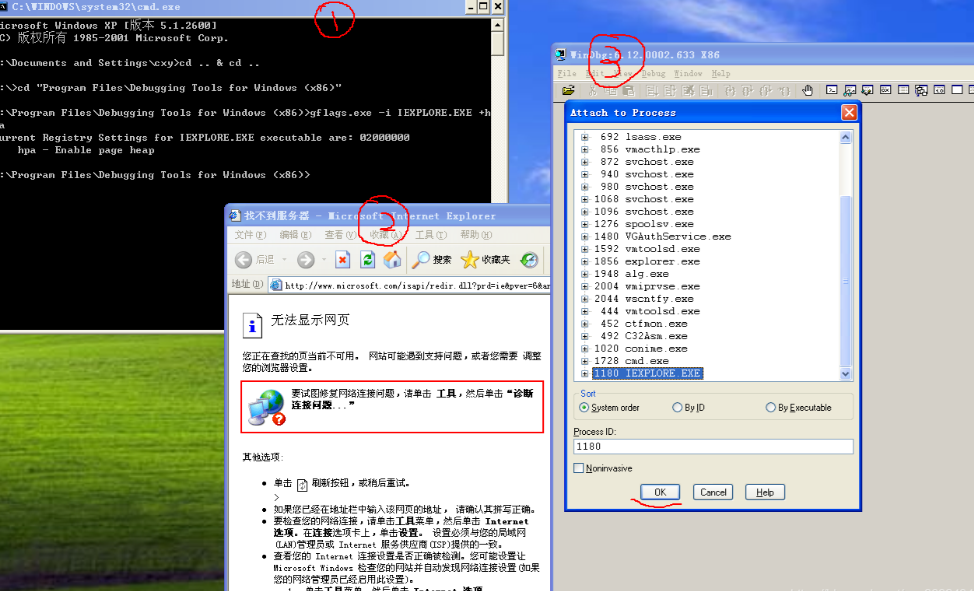
漏洞DLL：有漏洞的 winmm.dll 动态链接库文件

漏洞样本：能够触发漏洞 POC 样本（包含 poc.html 和 toto.mid 两个文件）

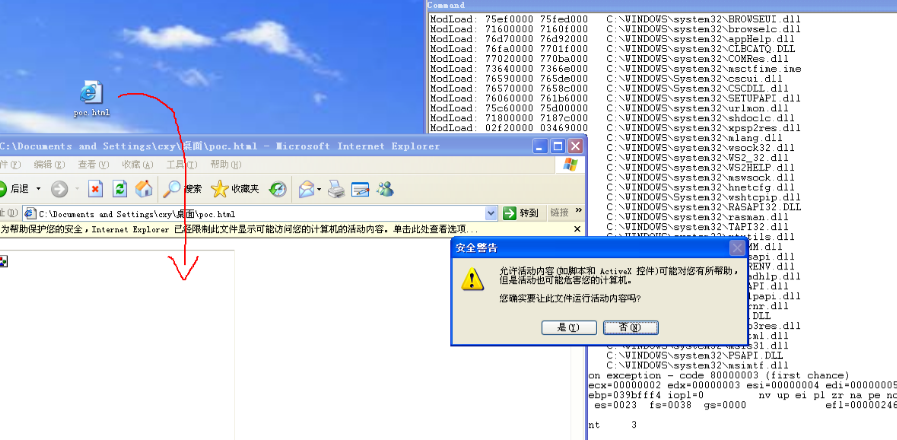
1. **实验步骤**

我们选择通过用 IE 加载 winmm.dll 的方式触发漏洞，为了让 IE 浏览器能够触发漏洞，所以在 poc.html 样本中嵌入了一个音频播放器，由音频播放器来调用 toto.mid 文件播放音频从而触发漏洞

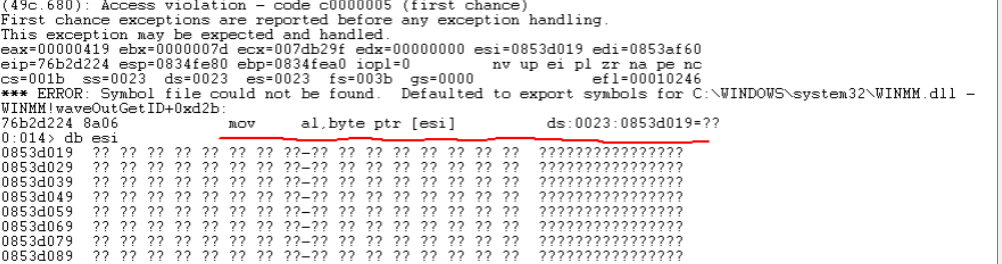


第一步设置堆页保护，第二部打开 IE，第三部使用 Windbg 附加进程

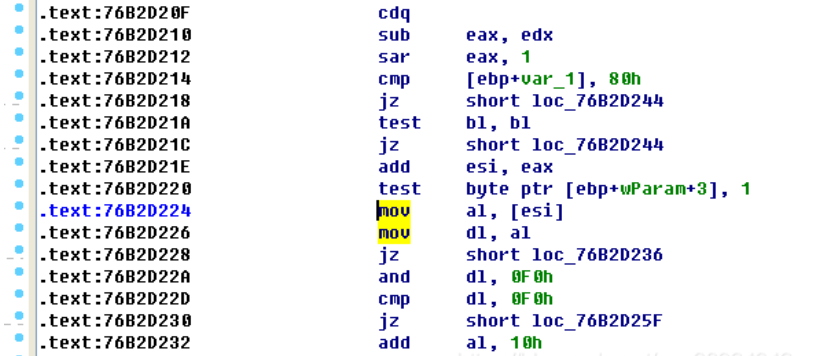
F5 运行之后将 poc.html 文件拖入 IE，并且允许运行 ActiveX 控件



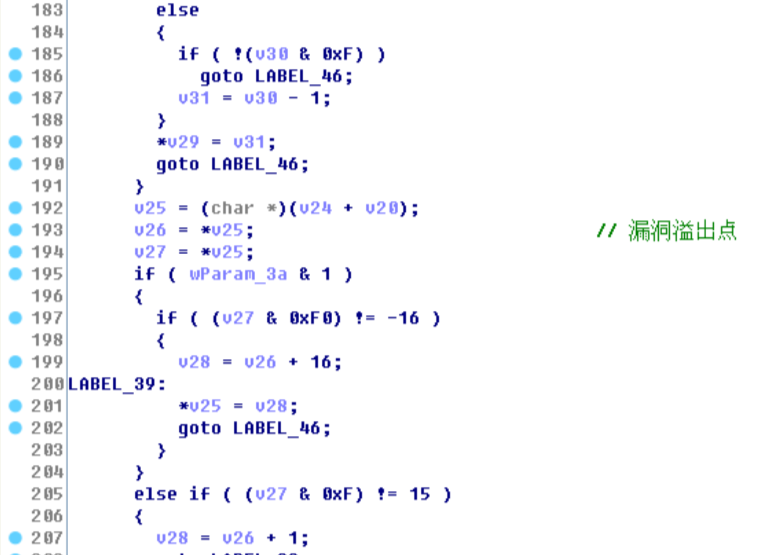
成功断下异常，可以看出 esi 指向的地址超出了堆空间，之后被堆页保护捕捉到，可以看出是 winmm.dll 中出了问题



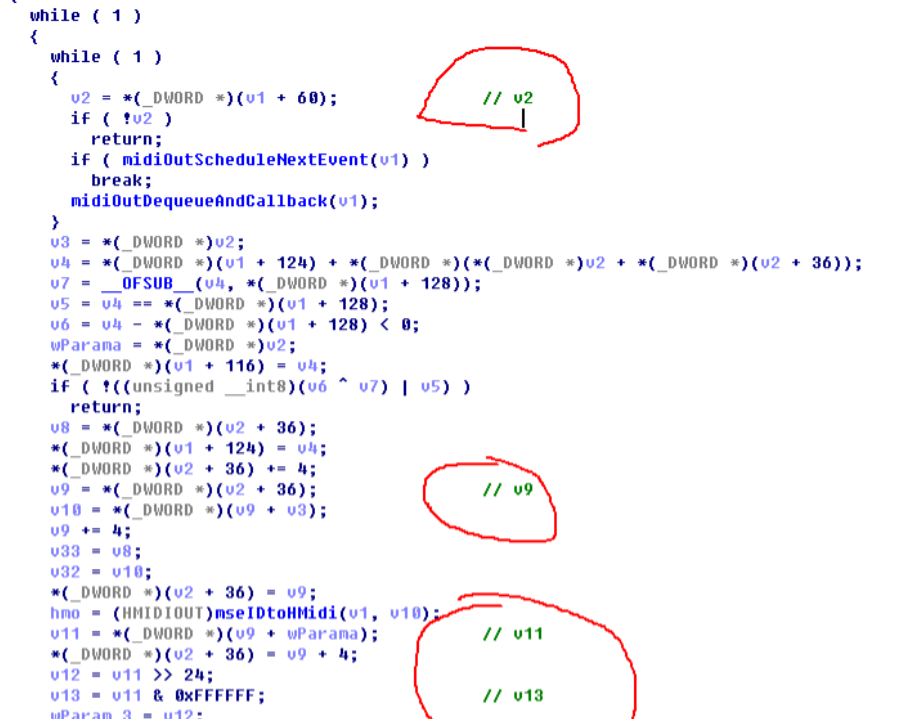
IDA 载入这个 winmm.dll，该动态库在 system32 下

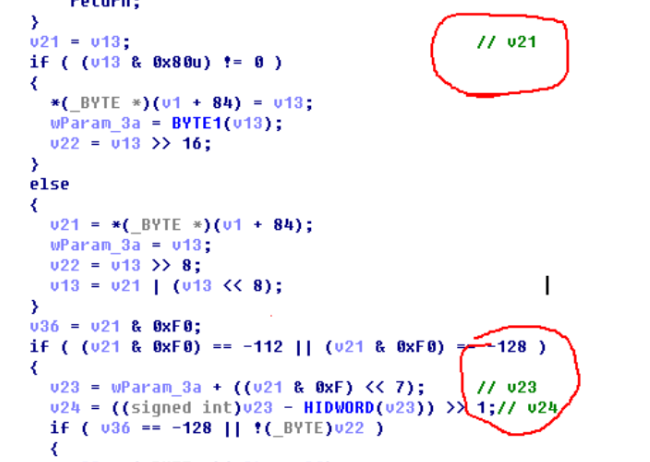


下图可以看出是 v25 这个指针越界访问了堆上的数据

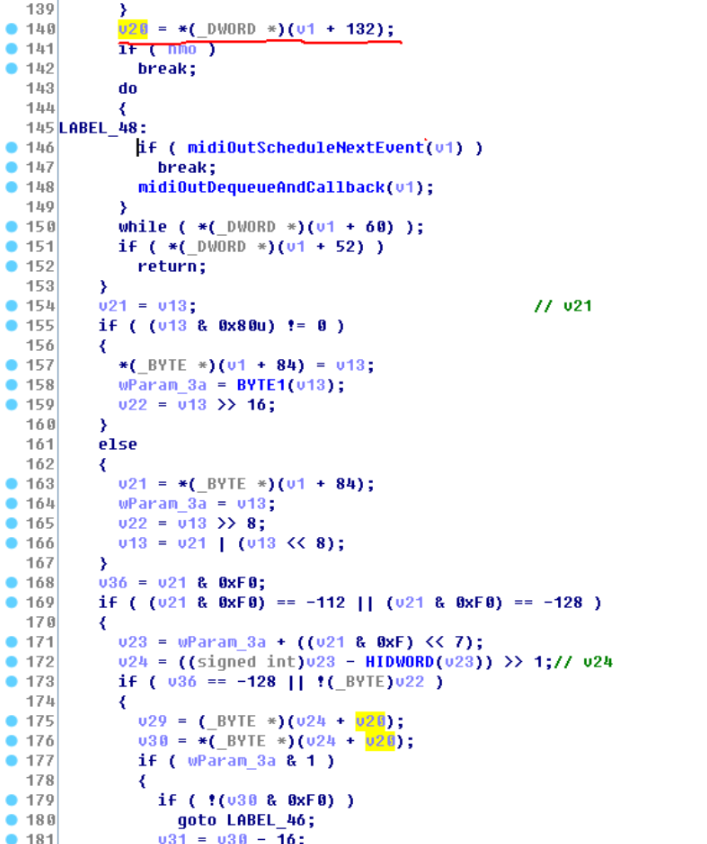


v25 受 v24 和 v20 影响，通过观察可以发现 v24 受 v2、v9、v11、v13、v21影响（v2 由 wParam 控制）

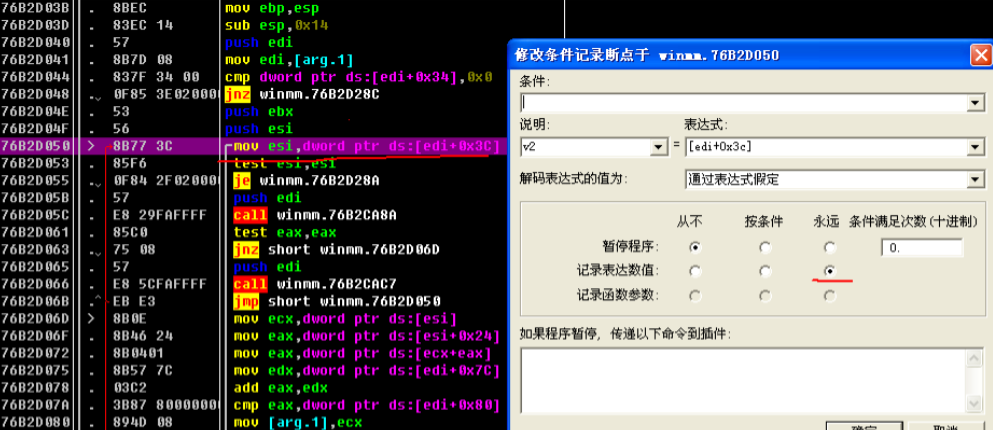




v20 受 v1 影响（v1 也由 wParam 控制）

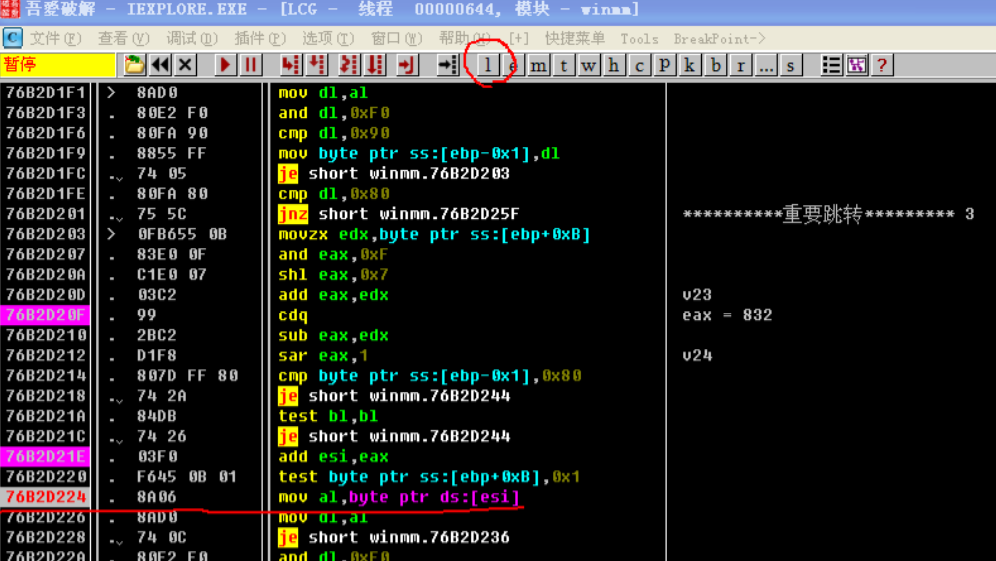


将 IE 载入 OD 运行，先不要将 html.poc 拖入到 IE 中，之后 shift + F4 对 v2、v9、v11、v13、v21、v23 和 v20 下条件记录断点



接着设置v9,v11,v13,v21,v23,v20

将 poc.html 文件放入刚刚运行的 IE 之中，之后会在异常触发点断下，按 l 这个快捷小按钮查看日志



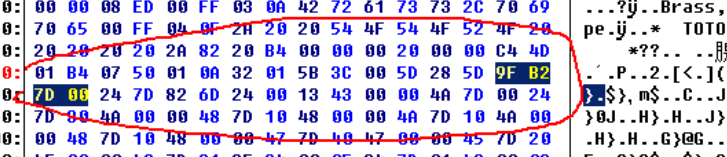
可以看出 v11、v13、v21 相等

v2 不变，v20 最后才被记录，且指向的地址都在 0x07EC0000

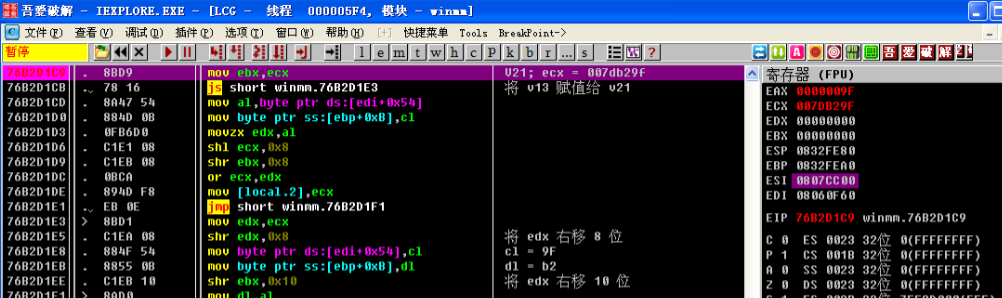
这个就是样本中的数据，刚好是倒过来的 0x007DB29F往后的堆中从资料中可以得知最后 v23 的值，也就是 0x007DB29F 是来自漏洞样本的数据，结合MIDI 格式分析可以知道，9F 代表的就是打开音符（Note On），F 就是通道号



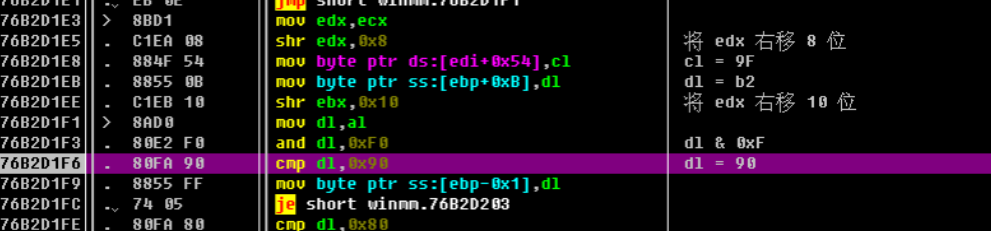
这个就是样本中的数据，刚好是倒过来的 0x007DB29F



重新运行后断下，注意此时 ecx = 007DB29F，之后接着往下调试

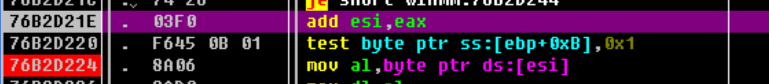


首先会将 9F 和 0xF0 做按位与运算，之后将结果与 90 作比较，判断这是打开音符（Note On）还是关闭音符（Note Off）；由于 9F0 | 0xF = 90，所以这个是打开音符

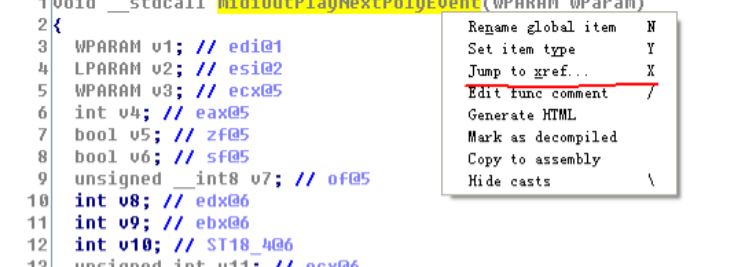


之后将 9F 和 0xF 做按位与操作，取出通道号 F，之后左移 7 位右移 1 位，最后等于 419

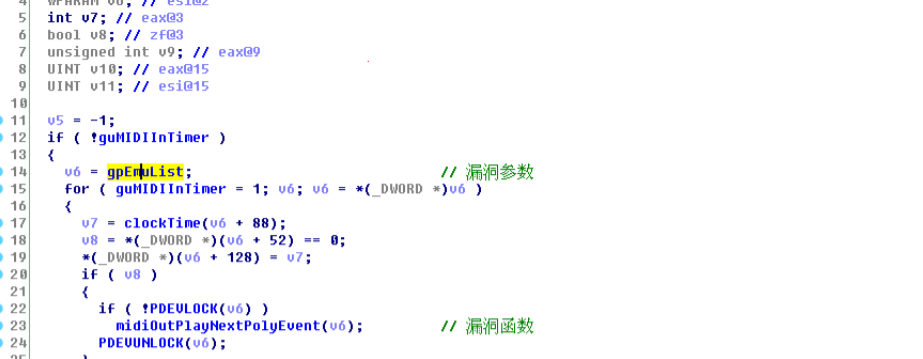


最后将 esi（堆基址）+ eax（偏移地址419）

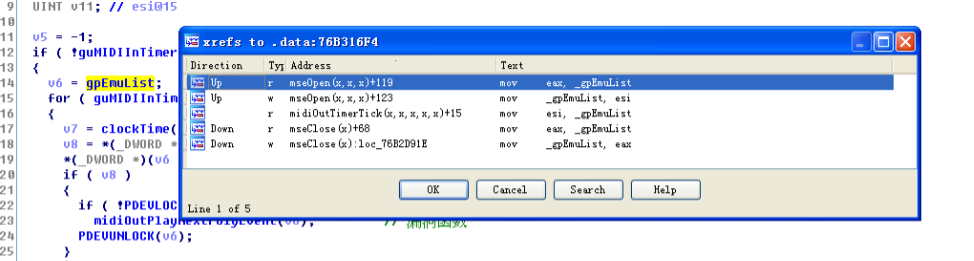
这时已经知道通道号 F 经过计算之后值为 419，控制着指针的偏移地址，也就是 v24 的值此时只需要知道 esi，也就是 v20 的值是怎么来的，就可以完全搞清楚漏洞的成因了首先 v20 由传入的参数 wParam 控制，对 midiOutPlayNextPolyEvent 函数进行交叉引用查询，看看是哪一个函数调用的这个函数



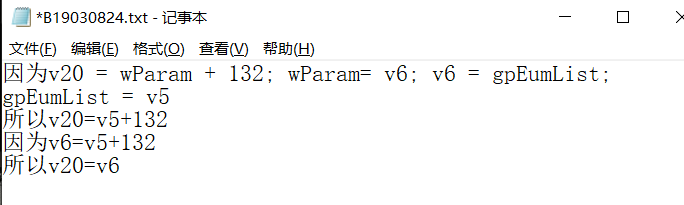
发现是 midiOutTimerTick函数调用midiOutPlayNextPolyEvent 函数，并且把 v6 当作参数传入，而 v6 = gpEmuList



gpEmuList 不是参数传入的，所以查询 gpEmuList 的交叉引用，看看是哪一个函数设置了 gpEmuList 这个值，发现是 mseOpen 函数中引用了这个值



这个就是 mseOpen 函数，gpEmuList 由 v5 传入，v5 等于 winmmAlloc 函数分配的堆空间地址整理一下调用关系



因为 v6 是堆块的基地址，且这个堆块的大小为 0x400，又因为 v20 = v6，所以 v20 也为这个 0x400 堆块的基地址

这样关系就很明了了，v25 的指针由 v24 和 v20 相加得出，v20 是堆的基址，而 v24 是样本数据解析运算获得，由于没有限制通道号的大小，导致运算后的偏移地址 v24 为 419，可是堆只有 0x400大小，很显然超出了申请的堆空间，照成了非法访问，故引发了堆页保护异常

后面的漏洞的应用太过于复杂，难以实施，就不做阐述。

**四、总结**

通过本次实验，我加强了对于安全编程技术的理解，并思考到了漏洞分析无处不在，让我认识到网络是很难完全安全的。

通过动手实验，让我对Microsoft Windows Media Player winmm.dll MIDI 文件堆溢出漏洞有了更深刻的理解，虽然这个漏洞目前已经被修复，但是通过学习这些知识，能让我对未来的学习有更深刻的理解。