**notepad++堆缓冲区溢出漏洞CVE-2023-40031**

**分析与复现**

**漏洞概述**

Notepad++ 是一款知名的开源代码编辑器，运行系统为Windows，支持多种编程语言。近日，安全研究人员针对 Notepad++进行检查，发现了不少安全漏洞。其中评分为 7.8 分（CVSS3，总分10分）的堆缓冲区溢出漏洞CVE-2023-40031，属于高危漏洞。该漏洞位于Utf8\_16\_Read::convert函数中，进行UTF-16 到 UTF-8 转换时，错误计算转换后的UTF-8堆缓冲区的大小，导致该缓冲区之外的内存空间被覆盖，可能导致任意代码执行。

**影响范围**

<=8.5.6

**复现环境**

操作系统：Win7 sp1。

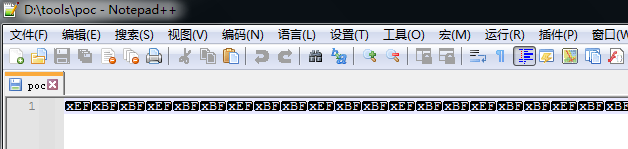
分析工具：IDA、WinDbg、OLLYDBG。

**漏洞复现**

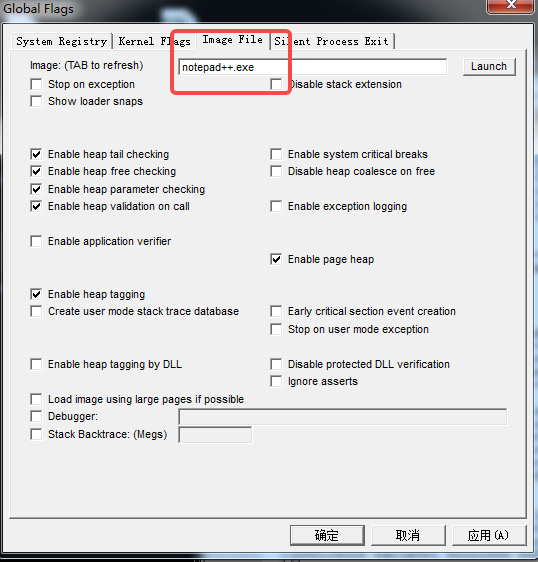
根据安全研究人员公开的报告（https://securitylab.github.com/advisories/GHSL-2023-092\_Notepad\_\_/），使用python生成poc文件。poc代码，如下图所示：



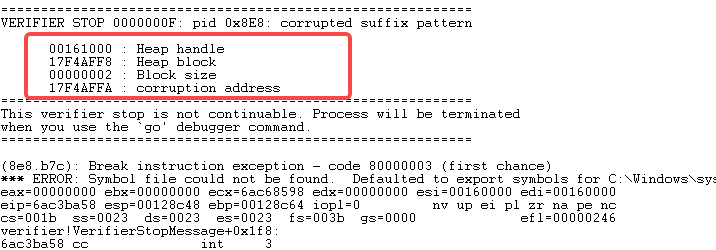
安装8.5.2版本的notepad++程序（https://github.com/notepad-plus-plus/notepad-plus-plus/releases/download/v8.5.2/npp.8.5.2.Installer.exe），安装完成后将生成的poc文件在notepad++程序中打开，但并未发生崩溃，可能是由于溢出后的数据并未影响程序的正常运行。如下图所示：



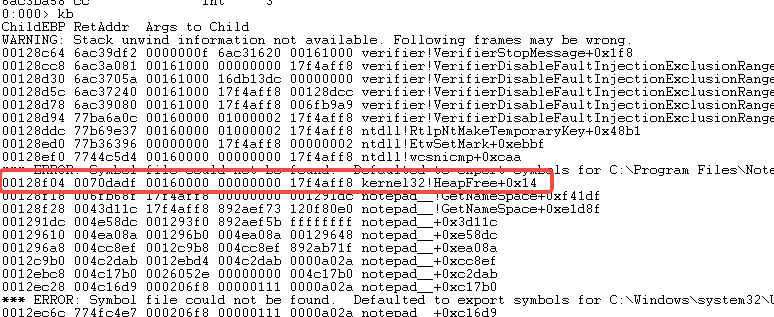
使用Windbg调试工具中的组件gflags.exe，打开堆检查相关选项，如下图所示：



然后重新打开notepad++程序，再使用Windbg调试工具附加后运行，最后在notepad++程序中打开生成poc文件。Windbg中立即捕获异常，在地址为0x17F4AFF8，大小为2个字节的堆中出现异常，异常的地址为0x17F4AFFA，已经超出了当前堆的地址范围，出现溢出，如下图所示：

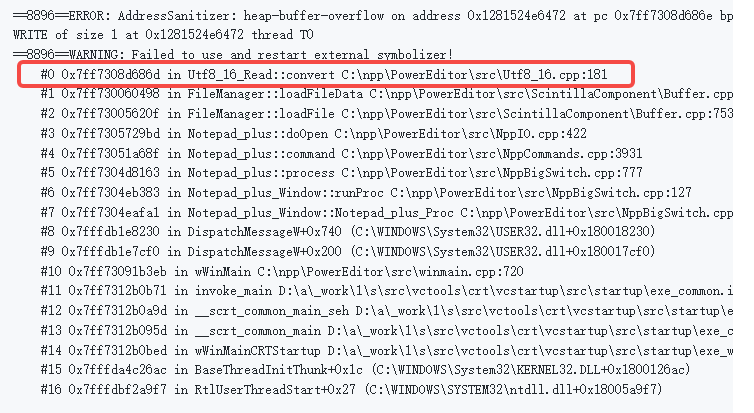


查看函数调用堆栈，发现该异常是发生在HeapFree函数中，也就是说在释放堆时该内存已被破坏。函数调用堆栈，如下图所示：

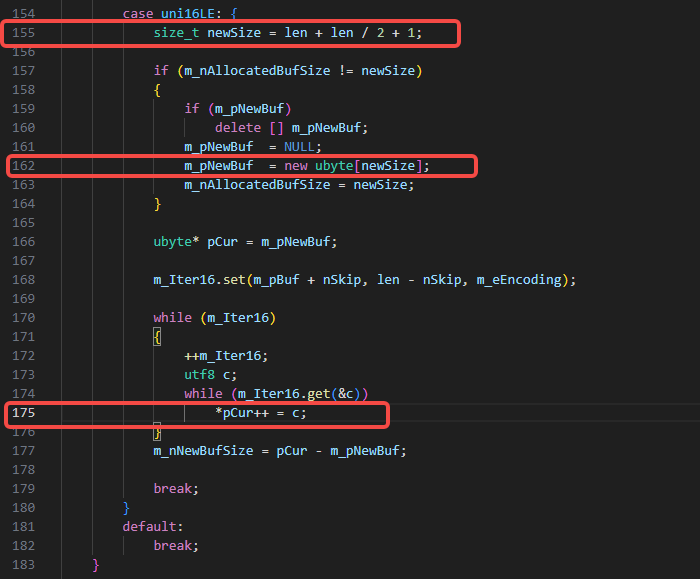


**漏洞分析**

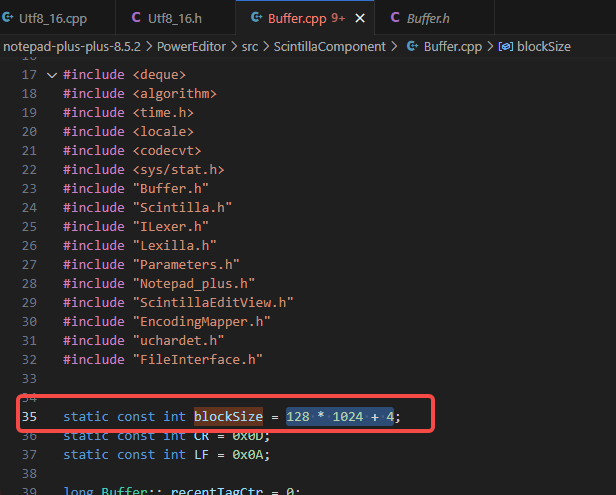
根据以上复现的情况，已经知道发生了堆溢出，但是还没有找到溢出的原因。一般可以根据函数调用堆栈，往上依次查找。不过这是开源软件，源码可下载自行编译，所以结合AddressSanitizer （ASan）技术，可以更为准确的定位溢出原因。在公开的poc报告中，已经使用该技术定位到溢出原因。溢出发生在Utf8\_16\_Read::convert函数中，如下图所示：



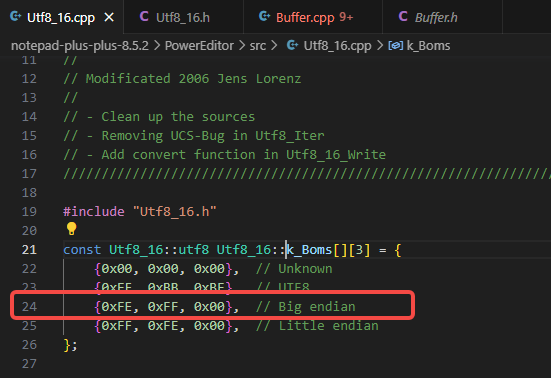
初步分析，在该函数中第162行使用new申请堆缓冲区，大小是经过155行的len计算得来，175行向申请的缓冲区复制数据，溢出点可能是这里，如下图所示：



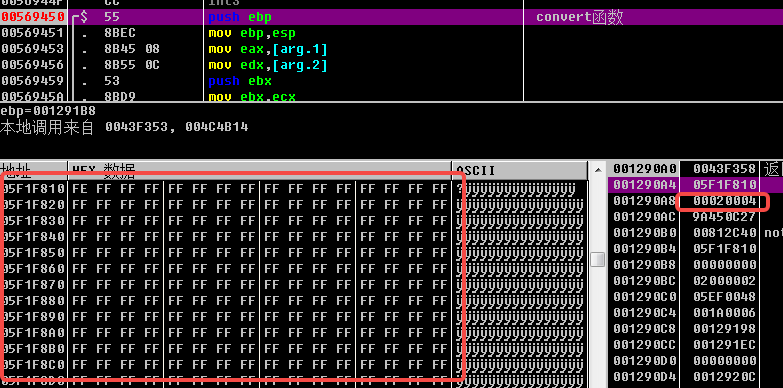
使用IDA反编译notepad++.exe，再使用OLLYDBG动态调试，打开poc文件进行跟踪，详细分析该函数功能。经过一段时间的调试分析，可知notepad++每次从文件中读取指定大小的内容，这个大小是128 \* 1024 + 4=0x20004，如下图所示：



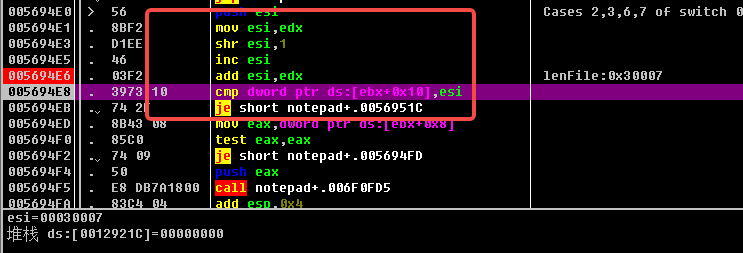
然后再判断文件内容编码，poc文件的前两个字节为0xFE和0xFF，说明其内容编码为UTF-16大端字节序，如下图所示：



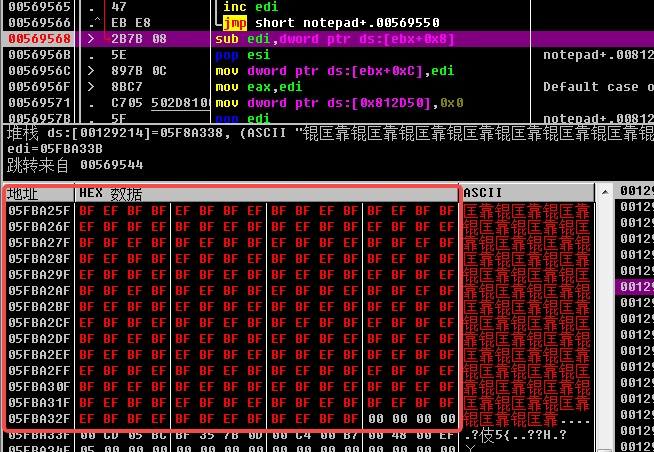
最后将进入Utf8\_16\_Read::convert函数，将UTF-16编码的内容转换为UTF-8编码。在转换时，重新计算转换后的内容的大小，计算公式为newSize = len + len / 2 + 1，len为转换前内容的大小，也就是新大小比原大小的3/2倍还多1个字节。在读取poc文件时，第一次读取的大小为0x20004，如下图所示：



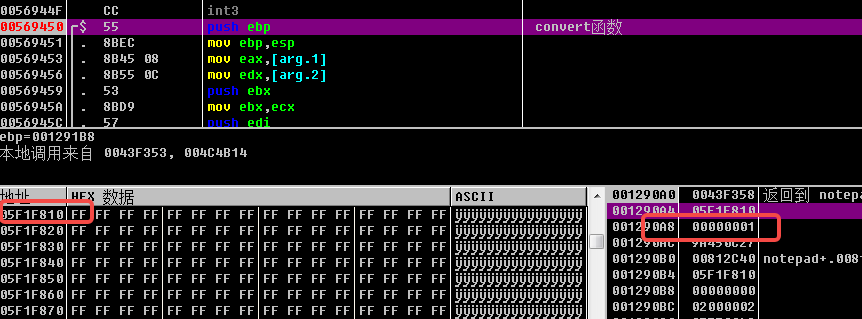
计算转换后新缓冲区的大小为0x30007，如下图所示：



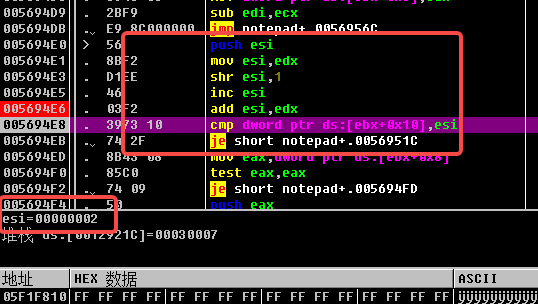
再进行编码转换，将转换后的内容存入新的缓冲区，未发生溢出，如下图所示：



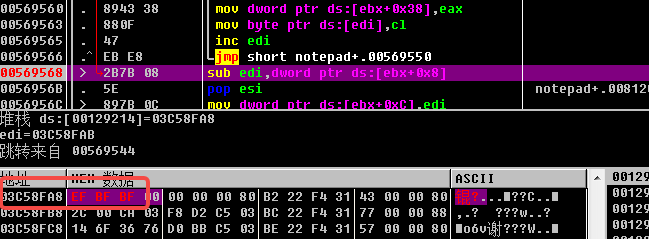
第二次读取poc剩余内容时，只剩下1个字节，但是上一次读取到缓冲区的内容未清除（关键点1），如下图所示：



接着计算新缓冲区的大小为2（关键点2），如下图所示：



再进行编码转换，此时由于只有一个字节，而UTF-16至少是2个字节，所有将在缓冲区中向后再读一个字节（关键点3），这个字节内容是上一次读取的内容。此时的UTF-16内容是0xFF，0xFF，转换后为0xEF，0xBF，0xBF，一共3个字节，超出2个字节大小的新缓冲区，发生溢出，如下图所示：



综上，当poc文件大小为奇数，在进行UTF-16到UTF-8转化时，计算转化后缓冲区大小将发生错误，同时对文件内容的结束位置也判断错误，导致转化后堆缓冲区溢出一个字节，覆盖其他内存，可能导致任意代码执行。

**漏洞利用**

溢出一个字节被称为off-by-one。一般来说，off-by-one被认为是难以利用的。常见的思路是利用该漏洞修改相邻堆大小造成块结构之间出现重叠，从而泄露其他块数据，或是覆盖其他块数据，这在CTF比赛中比较常见。想要直接进行rce的话还需要深入研究一些其他的组合利用方式。

**参考链接**

https://securitylab.github.com/advisories/GHSL-2023-092\_Notepad\_\_/

https://threatprotect.qualys.com/2023/08/30/multiple-vulnerabilities-in-notepad-allow-attackers-to-perform-arbitrary-code-execution/

https://community.notepad-plus-plus.org/topic/24889/notepad-v8-5-7-release-candidate

https://www.landiannews.com/archives/100096.html

https://nvd.nist.gov/vuln/detail/CVE-2023-40031