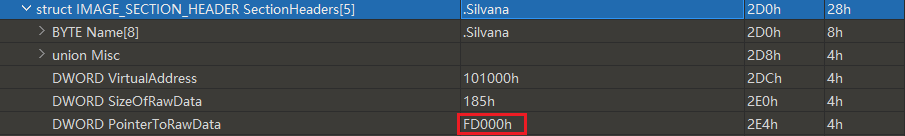
**简易PE Hooker开发 进度记录与控制**

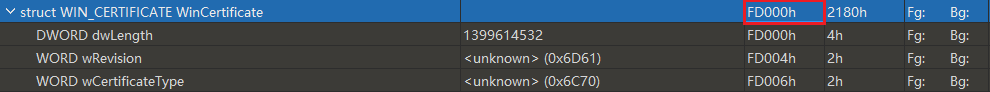
**TODO List**

1. 预备Hook代码的机器指令
   1. MASM汇编指令开发
2. 劫持PE文件控制流到Hook代码
   1. 解析PE文件结构
   2. 添加新段并为指令适配空间

**PE修改方案选择**

正常情况下, 如果需要劫持PE文件的控制流, 最简单的方法就是修改PE文件的导入表, 通过DLL加载需要的代码, 但为了项目后续的可拓展性, 本档选用另一种方案, 即首先预热Hook代码为机器指令, 随后将指令直接插入PE文件内, 修补IMAGE\_OPTIONAL\_HEADER.AddressOfEntryPoint以在较早的时机劫持控制流.





使用该方案时, 需要解决的问题也随之产生, 首先是如何存放Hook指令? 本档选择效仿工具LordPE的处理手法, 为PE文件添加新段, 并在段内存放所有必要数据, 但新问题随之而来, 在何处存放新段? LordPE选择将PE文件内的整个WIN\_CERTIFICATE结构作为新段以存储数据, 这么做的好处首先是, 覆盖该结构除了会让文件丢失数字签名, 大概率不会影响原PE文件的代码运行逻辑, 其次, 数字签名的体积足够大, 可以轻易存储大部分所需数据与Hook机器指令.

但考虑到PEHooker项目需要尽可能地处理大部分PE文件, 同时, 并非所有PE文件均携带数字签名, 且覆盖原文件仍有可能导致不可预料的错误, 因此, 考虑到兼容性因素, 本档选择将新段添加在PE文件的末尾, 以此保证原文件的大部分内容不发生改变, 且如此做的好处是, 相比于覆盖WIN\_CERTIFICATE结构, 在末尾插入数据并不受体积因素限制, 这反而增加了此方案的可拓展性.

插入新段的具体操作参考了此文: <https://www.sunshine2k.de/reversing/tuts/tut_addsec.htm>

**PEHandler结构**

需要提前记录的信息

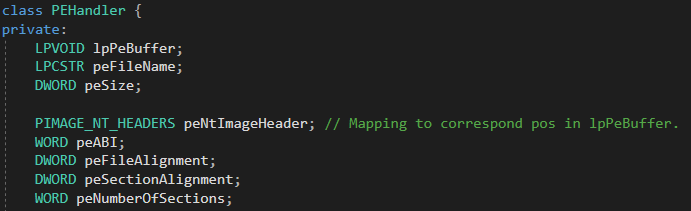
Magic PE魔数, 方便后续区分X86与X64的处理方案

Section/FileAlignment 插入新段时进行内存对齐

PEBuffer PE文件内容

PESize PE文件大小

更多细节详见代码片段, 不再赘述.



**MASM汇编指令开发**

原方案是希望在CPP层使用内联汇编, 或者直接编译完整方法为指令后, 动态地进行提取, 但考虑到编译器可能优化掉方法的某些指令细节, 且无法精细地控制指令, 这不利于为PE文件定制事实可行的注入指令, 因此为了兼顾后续的可拓展性与稳定性, 最终选定使用MASM手动编写指令.