Внедрение дополнительного кода в исполняемый файл

1. Модификация заголовка и структуры исполняемого кода

2. Вставка дополнительного кода

Требуемый инструментарий

ОТЛАДЧИК ИСПОЛНЯЕМОГО КОДА
OLLY DEBUGGER

РЕДАКТОР ЗАГОЛОВКА ИСПОЛНЯЕМОГО ФАЙЛА ФОРМАТА PE
LORD PE, PE TOOLS И ДР.

ШЕСТНАДЦАТИРИЧНЫЙ РЕДАКТОР
FLEX HEX, WIN HEX И ДР.

ОБЪЕКТ ВОЗДЕЙСТВИЯ
ИСПОЛНЯЕМЫЙ ФАЙЛ

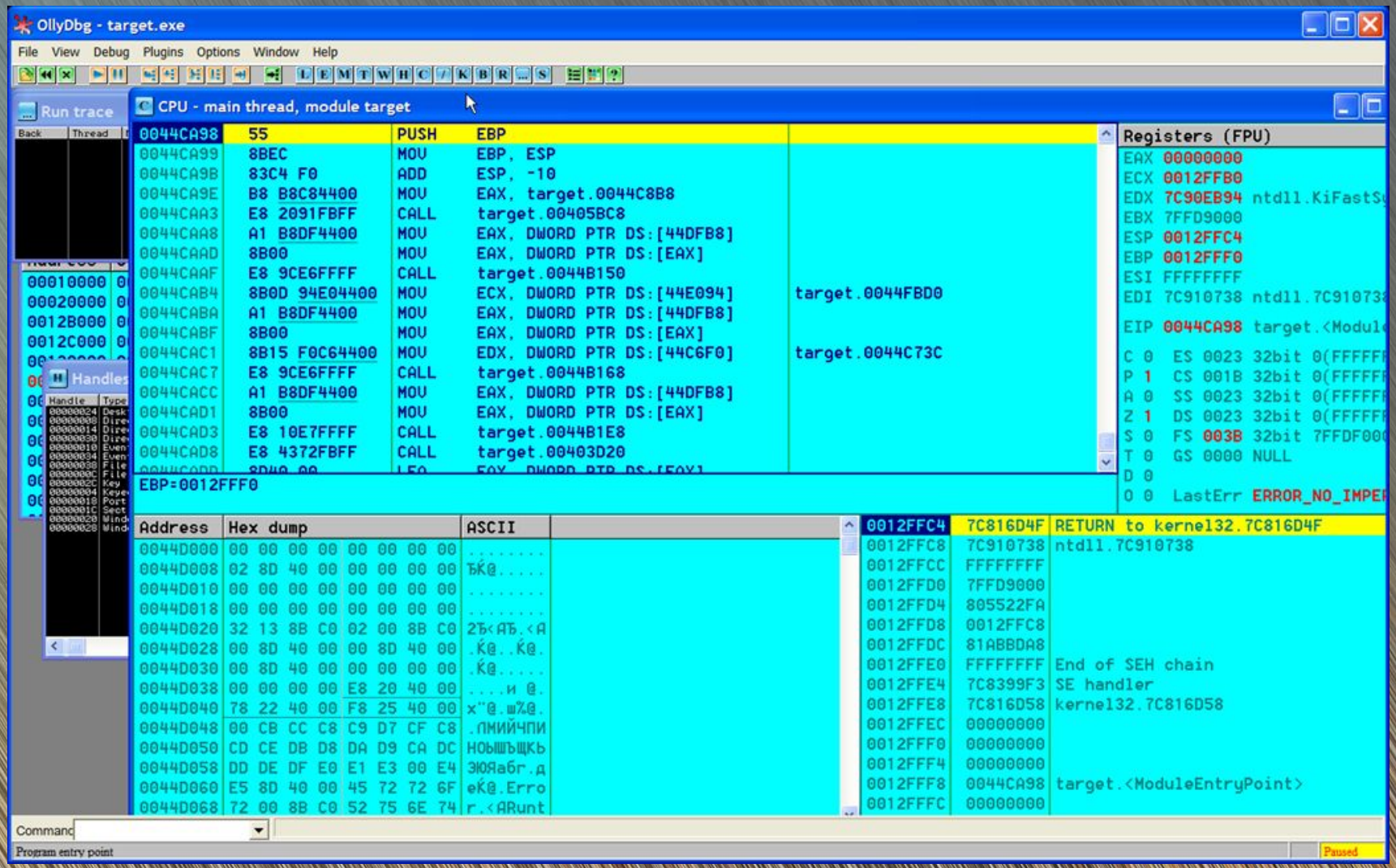
Алгоритм действий

1. Создаем проект в среде программирования с пустой графической формой.

2. Внедряем свой код в созданный проект. Код должен исполняться при запуске графического пользовательского приложения (пустая форма).

Варианты внедрения кода

Открытие созданного приложения в Olly Debugger.

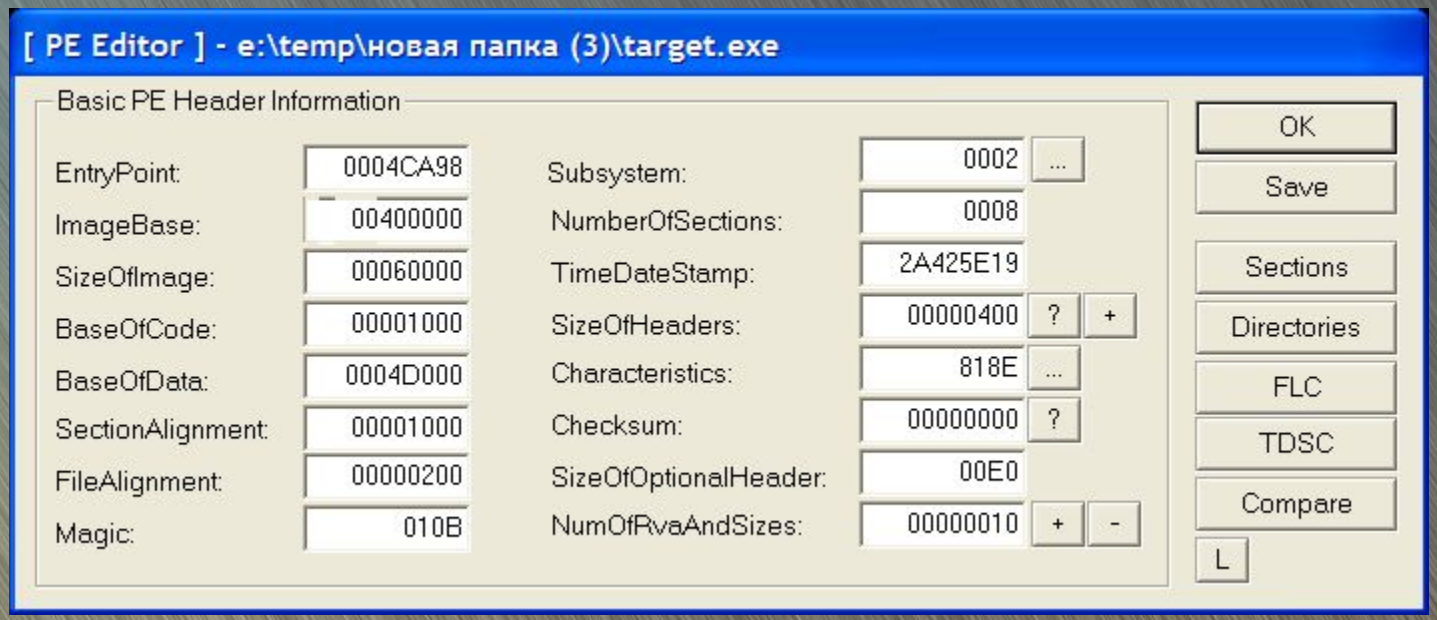


Варианты внедрения кода

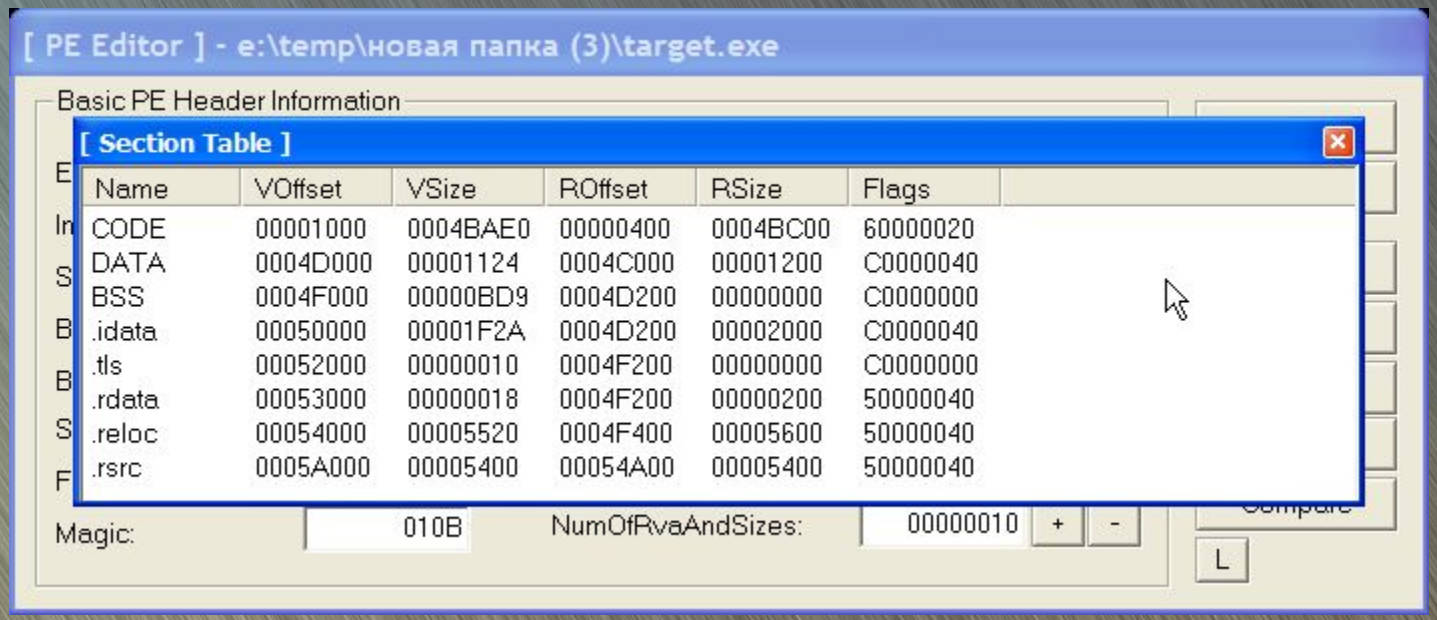
1. Запись кода в память процесса.
2. Запись кода в новую секцию (данных или кода), которая создается дополнительно.

Обоснование:
Места в существующей секции кода мало (RawSize – размер в исполняемом файле), а размер процесса в памяти гораздо больше (VirtualSize) => VirtualSize >> RawSize.

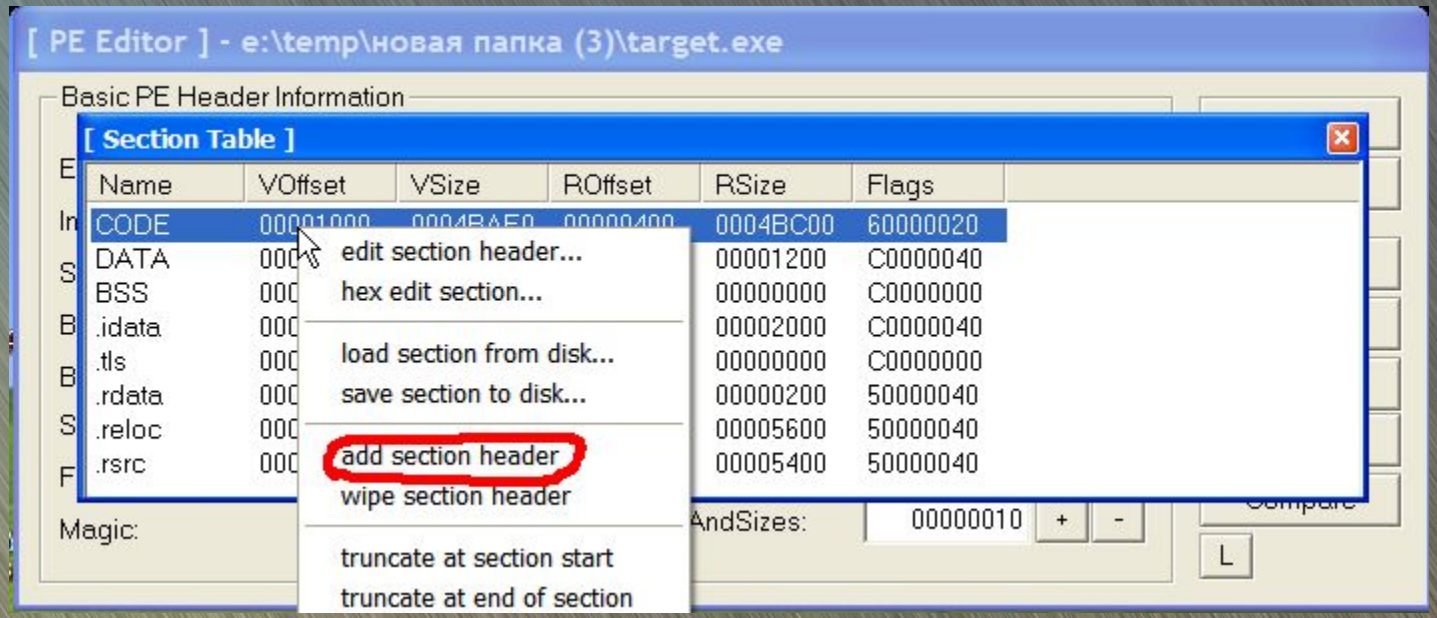
1. **Открываем модифицируемый файл в Lord PE или PE tools.**



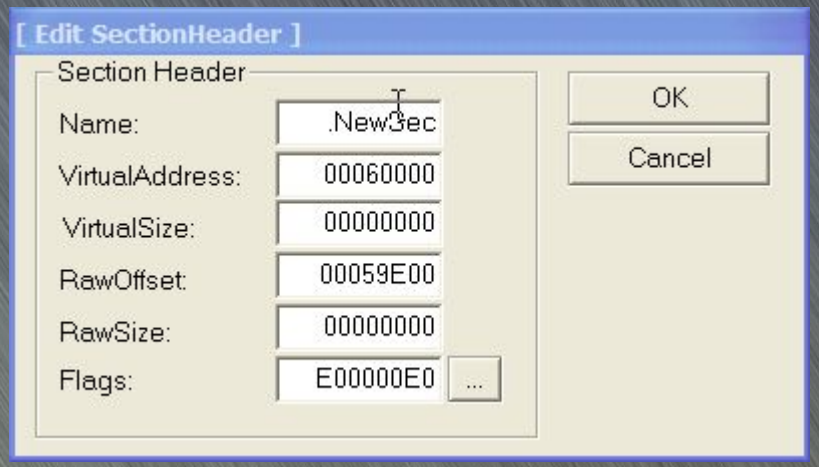
2. Открываем раздел Section.



3. Добавляем заголовок секции (Section).



4. Редактируем заголовок секции (Section).



a) **В поле Name вводим какое-нибудь название секции, например “.code”;**

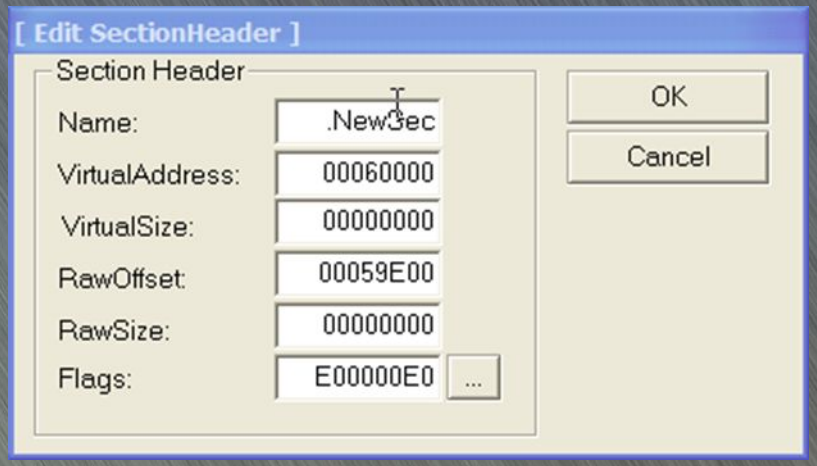
b) **Поля VirtualAddress и RawOffset не трогаем – это адреса секции в памяти и в файле соответственно, вычисляются как**

VirtualAddress = ((VirtualAddress(предыдущая секция) + VirtualSize(предыдущая секция) – 1) div VirtualAlign) + 1) * VirtualAlign;

RawOffset = ((VirtualAddress(предыдущая секция) + VirtualSize(предыдущая секция) – 1) div FileAlign) + 1) * FileAlign;

Их LordPE считает автоматом, и менять их не надо!

4. Редактируем заголовок секции (Section).

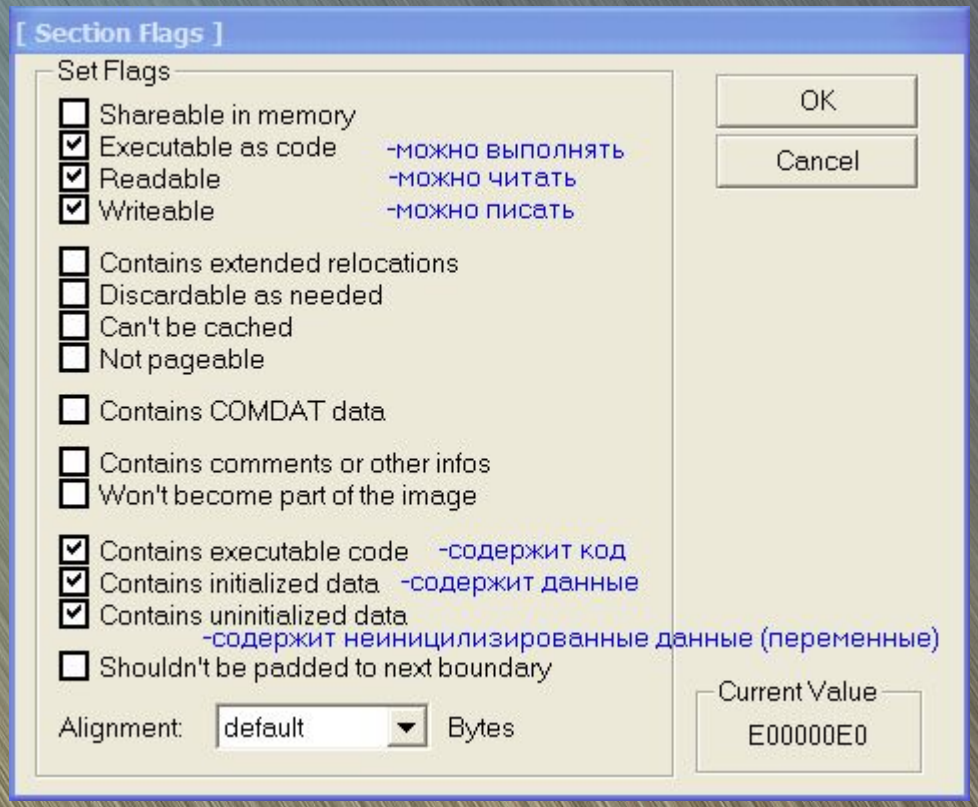


c) Расчитываем поля VirtualSize и RawSize

RawSize - размер секции в файле = 1000 (4кб – достаточно для нашего кода).

VirtualSize - размер секции в памяти = 4000 (16кб – писать в память не только код, но и данные, чтобы не создавать новую секцию).

5. Редактируем флаги секции. Нажимаем на кнопку рядом с полем Flags:

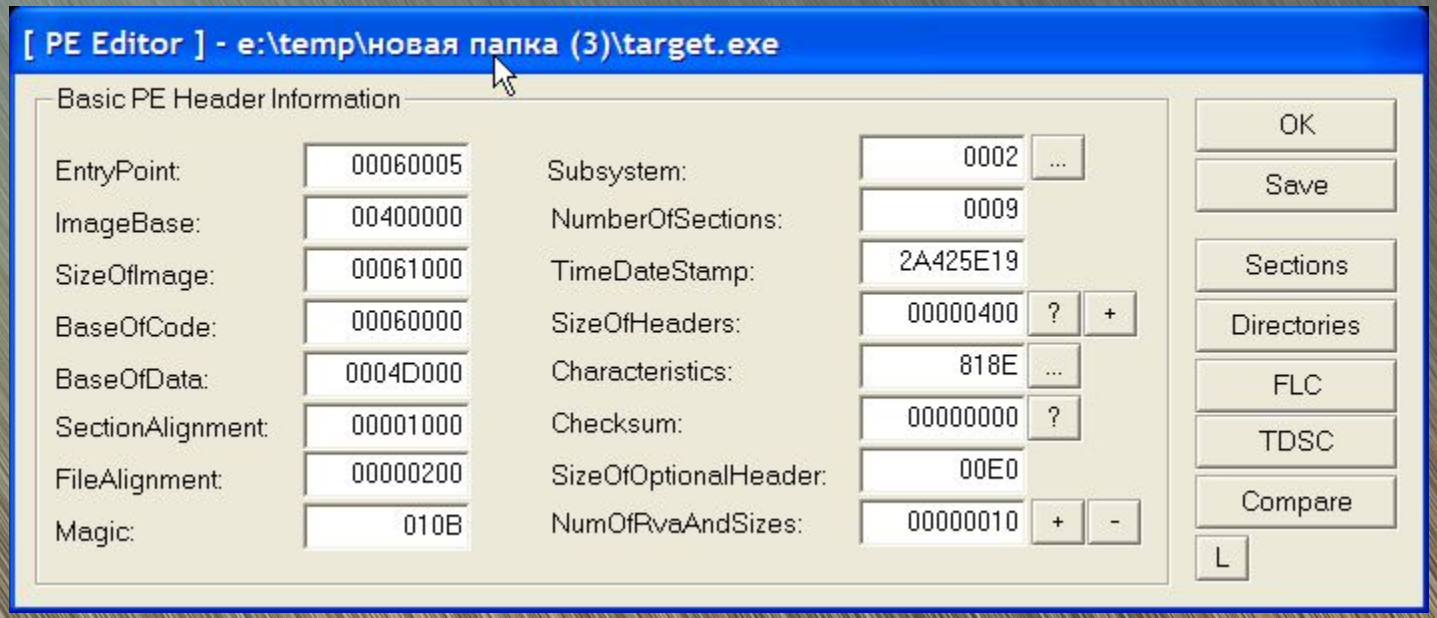


Отмечаем (ставим галочку):

- Что-то в секции можно выполнять
- Из неё можно читать
- В неё можно писать
- В ней есть код
- В ней есть инициализированные данные
- В ней есть неинициализированные данные

6. Запомнить и записать значения VirtualOffset (RVA) и RawOffset, EntryPoint, BaseOfCode

7. Пишем вместо BaseOfCode значение RVA и вместо EntryPoint значение RVA нашей секции + 5.



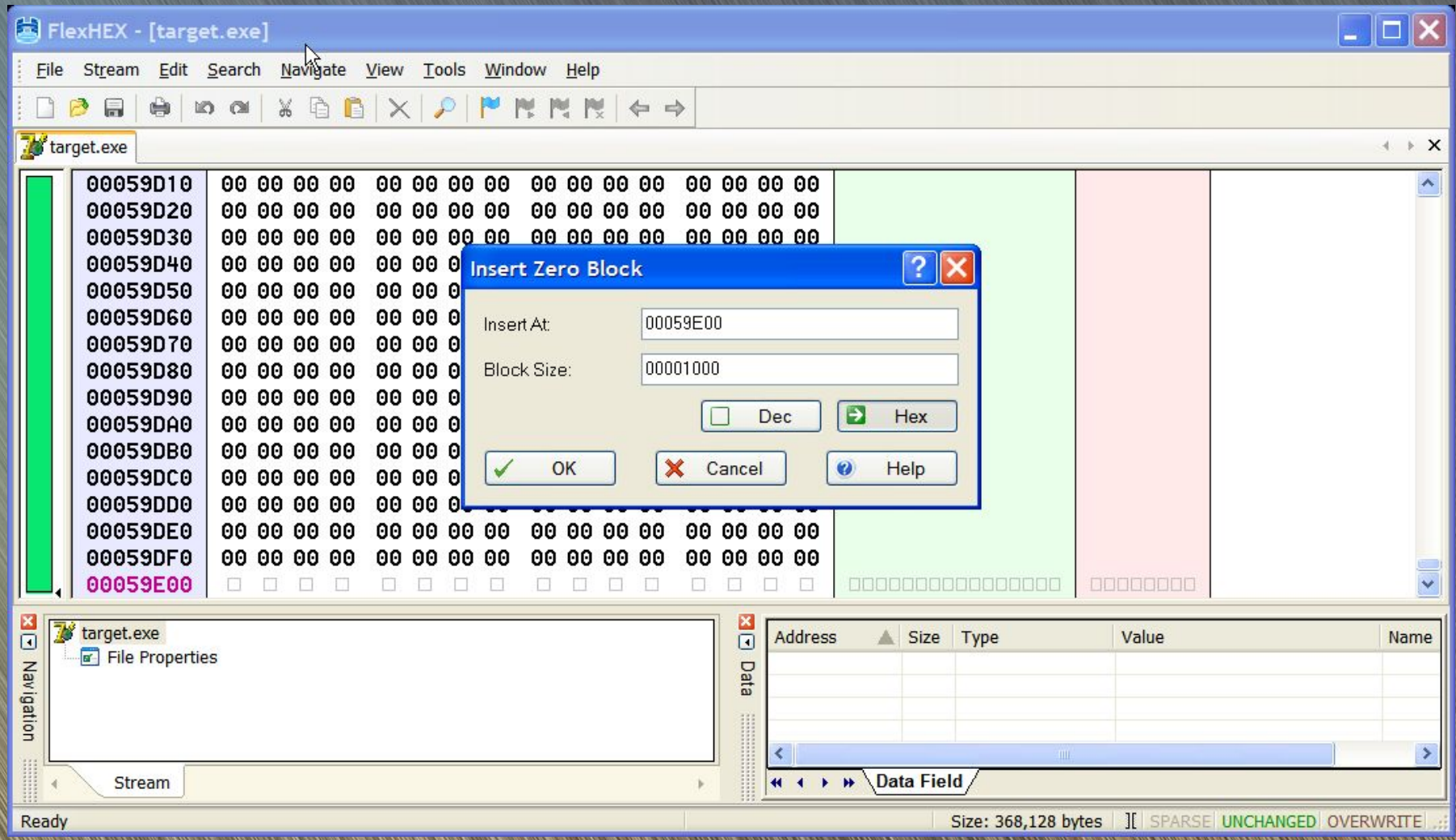
8. Нажимаем Save и «OK». Запускаем файл. Он не запускается…

Секцию объявили, а записать – не записали!

9. Открываем файл во FlexHex и переходим в самый конец файла. Теперь ищем то место, где будет новая секция. А она начинается с запомненного нами RawOffset (назовём его Raw = 59E00).

Если последний существующий байт файла имеет адрес, отличный от (Raw-1), то забиваем нулями всё место от конца файла до этого числа (Щёлкаем после последнего байта в файле, далее Edit->Insert Zero Block; Block Size = Raw - адрес последнего байта - 1.

Иначе щёлкаем на пустой квадратик по адресу Raw, далее Edit->Insert Zero Block; Block Size = размер нашей секции, RawSize, то есть 1000h.


FlexHEX - [target.exe]
File Stream Edit Search Navigate View Tools Window Help
target.exe
00059D10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00059D20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00059D30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00059D40 00 00 00 00 00 00 0
00059D50 00 00 00 00 00 00 0
00059D60 00 00 00 00 00 00 0
00059D70 00 00 00 00 00 00 0
00059D80 00 00 00 00 00 00 0
00059D90 00 00 00 00 00 00 0
00059DA0 00 00 00 00 00 00 0
00059DB0 00 00 00 00 00 00 0
00059DC0 00 00 00 00 00 00 0
00059DD0 00 00 00 00 00 00 0
00059DE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00059DF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00059E00
Insert Zero Block
Insert At: 00059E00
Block Size: 00001000
Dec
Hex
OK
Cancel
Help
Navigation
target.exe
File Properties
Stream
Data
Address Size Type Value Name
Data Field
Ready
Size: 368,128 bytes
SPARSE UNCHANGED OVERWRITE