Создание Inline Patch для ExeCryptor 2.xx

ОГЛАВЛЕНИЕ

Введение

Как оказалось на паблике почти совсем нет инфы по инлайну криптора, а если и есть то такие «извращенные» приемы, хотя он не обновлялся уже очень давно, поэтому я решил это исправить.

Методы инлайна немного различаются для версий криптора до 2.4.1 и после 2.4.1. Это связано с тем что когда была версия 2.4.0 вышел мой анпакер и авторы сделали апдейт до версии 2.4.1 куда вставили несколько моментов для противостояния автоматической распаковки и по мелочам остального.

Поэтому я решил расписать отдельно создание патча для этих 2 групп.

Статья не претендует на оригинальность, просто описываю способы, которые сам использую.

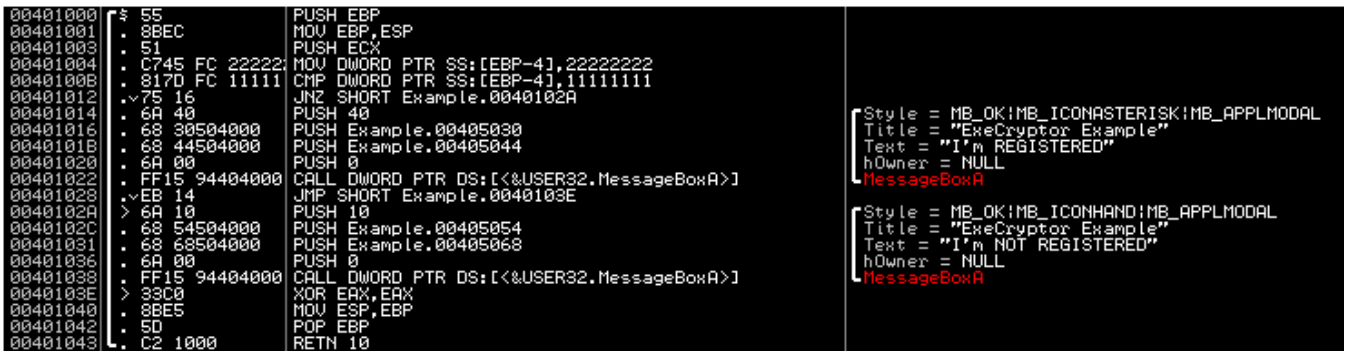
Создание примера

Нам нужна программа на которой мы будем тренироваться, я не стал брать в пример какую-нибудь shareware программу, ограничился созданием своего маленького и наглядного примера.

Итак, взял VC++ 6.0 и сделал простенькую заготовку с таким кодом(Example.exe) :

```
#include <windows.h>

#define     UNREG_VALUE     0x22222222
#define     REG_VALUE       0x11111111

int APIENTRY WinMain( HINSTANCE hInstance,
                      HINSTANCE hPrevInstance,
                      LPSTR     lpCmdLine,
                      int       nCmdShow    )
{
    DWORD   Reg = UNREG_VALUE;

    if( Reg == REG_VALUE )
    {
        MessageBox( NULL, "I'm REGISTERED", "ExeCryptor Example", MB_OK | MB_ICONINFORMATION );
    }
    else
    {
        MessageBox( NULL, "I'm NOT REGISTERED", "ExeCryptor Example", MB_OK | MB_ICONERROR );
    }

    return 0;
}
```

После компиляции в отладчике этот код выглядит следующим образом:



Собственно этого достаточно.

Обычный патч

Теперь рассмотрим пример обычного патча неупакованной программы, для того чтобы она работала как нам нужно.

Итак, если запустим сейчас тестовый пример (Example.exe) то он нам выдаст следующее:

Нам же нужно чтобы появилось сообщение об успешной регистрации.
Для этого нам нужно пропатчить код следующим образом (один из вариантов):

```
00401000  ┌$ 55              PUSH EBP
00401001  │. 8BEC            MOV EBP,ESP
00401003  │. 51              PUSH ECX
00401004  │  C745 FC 11111111 MOV DWORD PTR SS:[EBP-4],11111111
0040100B  │. 817D FC 11111111 CMP DWORD PTR SS:[EBP-4],11111111
00401012  │.v75 16           JNZ SHORT Example.0040102A
00401014  │. 6A 40           PUSH 40                                  ┌Style = MB_OK|MB_ICONASTERISK|MB_APPLMODAL
00401016  │. 68 30504000     PUSH Example.00405030                    │Title = "ExeCryptor Example"
0040101B  │. 68 44504000     PUSH Example.00405044                    │Text = "I'm REGISTERED"
00401020  │. 6A 00           PUSH 0                                   │hOwner = NULL
00401022  │. FF15 94404000   CALL DWORD PTR DS:[<&USER32.MessageBoxA>]└MessageBoxA
00401028  │.vEB 14           JMP SHORT Example.0040103E
0040102A  │> 6A 10           PUSH 10                                  ┌Style = MB_OK|MB_ICONHAND|MB_APPLMODAL
0040102C  │. 68 54504000     PUSH Example.00405054                    │Title = "ExeCryptor Example"
00401031  │. 68 68504000     PUSH Example.00405068                    │Text = "I'm NOT REGISTERED"
00401036  │. 6A 00           PUSH 0                                   │hOwner = NULL
00401038  │. FF15 94404000   CALL DWORD PTR DS:[<&USER32.MessageBoxA>]└MessageBoxA
0040103E  │> 33C0            XOR EAX,EAX
00401040  │. 8BE5            MOV ESP,EBP
00401042  │. 5D              POP EBP
00401043  └. C2 1000         RETN 10
```

Красным цветом выделена строка в которой произошли изменения.

Это сделали мы ручками, аналогия этого программно,
патч состоит из команды MOV DWORD PTR DS:[00401007], 11111111

после того как мы это сделали и сохранили (Example_.exe), запускаем и получаем:

То что и нужно, собственно на этом и остановимся с рассмотрением примера, теперь нам нужно сделать такой же патч, но только в упакованных криптором примерах.

Inline Patch версий до 2.4.1

Я запротектил наш пример версией криптора 2.2.6 (Example_2.2.6.exe)
С опциями сжатия кода и данных (по умолчанию).

Итак, собственно приступим:

Грузим файл в OllyDbg проблемы антиотладки и прочего тут не будут рассматриваться.

Логика создания любого инлайн патча это:

1) Дождаться пока распакуется нужный код
2) Пропатчить код
3) Пропатчить проверки CRC (если нужно).

1) Дождаться пока распакуется нужный код:

Т.к. интересующий нас участок кода находится в секции .text

Name:	VOffset:	VSize:	ROffset:	RSize:	Flags:
.text	00001000	00003000	00000400	00000000	E0000020

То ставим мемори бряк на запись в эту секцию, попадаем на такой код:

```
00442CFD  01DE            ADD ESI,EBX
00442CFF  EB 06           JMP SHORT Example_.00442D07
00442D01  85DB            TEST EBX,EBX
00442D03  74 0E           JE SHORT Example_.00442D13
00442D05  EB A9           JMP SHORT Example_.00442CB0
00442D07  56              PUSH ESI
00442D08  89D6            MOV ESI,EDX
00442D0A  89D9            MOV ECX,EBX
00442D0C  F3:A4           REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00442D0E  31DB            XOR EBX,EBX
00442D10  5E              POP ESI
00442D11  EB 9D           JMP SHORT Example_.00442CB0
00442D13  89F0            MOV EAX,ESI
00442D15  5B              POP EBX
00442D16  5F              POP EDI
00442D17  5E              POP ESI
00442D18  C3              RETN
```

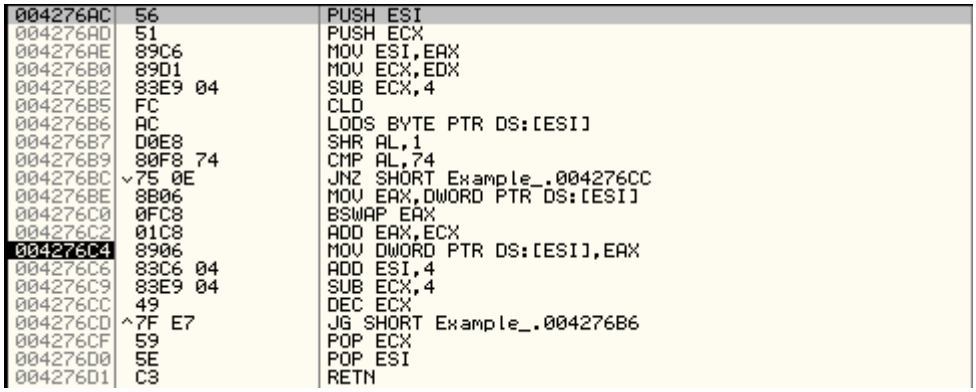
Это фунция которая распаковывает код, немножко не то что нам нужно, позже обьясню почему, а пока просто снимаем мемори бряк с секции .text и ставим бряк на код в конец функции распаковки.

Если вы попадете не на команду REP, а там будет STOS это всего лишь говорит что это выставлена доп. опция максимальной компрессии по алгоритму JCALG1 от Jeremy Collake.

В любом случае ищите ниже команду конца цикла распаковки
MOV EAX, ESI тут она у нас по адресу 442D13.

После того как функция распаковки отработала. В секции .text содержится распакованный код, но это еще не финальный вариант, т.к. особенностью данного алгоритма компрессии данных (как в LZMA) является после распаковки поправка смещений инструкций с опкодами E8 (call) и E9(jmp).

Поэтому после того как мы стоим в конце функции распаковки данных, ставим мемори бряк на запись в секцию опять, чтобы попасть на функцию поправки смещений. И попадаем туда, выглядит она следующим образом:



Параметры функции:
EAX – Адрес начала распакованных данных = VA секции
EDX – Размер данных = VS секции

После того как закончится цикл, стоя на адресе 004276CF в ESI будет лежать число равное = Адрес начала данных + его размер – 4.

Для распаковки первой секции функцию вызовут с параметрами:
EAX = 00401000
EDX = 00003000

После выполнения ESI = 00401000 + 00003000 – 4 = 00403FFC
Зачем я обратил внимание на ESI, да потому что с этой функции мы и будем начинать делать свой патч, а ESI нам будет служить для идентификации какая секция в данный момент распаковалась.

Возникает вопрос зачем нам это знать, ведь мы можем просто вставить код патча который постоянно будет выполняться с этой функцией, но тут есть маленький нюанс, после того как криптор распаковал и поправил смещения, он выставляет секции оригинальные атрибуты доступа, по умолчанию у кода нет атрибутов для записи и если мы будем делать прыжок на наш патч позже, нам придется или ставить снова атрибуты на запись или программа просто упадет с ошибкой! Поэтому нам нужно сделать своеобразную switch конструкцию для ESI где для определенного значения вызывать свои действия.

Скажем, если ESI = 1 то была распакована секция кода и мы будем патчить необходимый нам код, если ESI = 2 то была распакована секция данных и мы там подправим строчку, если ESI = 3 то была распакована «Экстра» секция криптора и мы будем патчить там проверку CRC файла, т.е. думаю смысл понятен.

Функция которая правит смещения открыто лежит в секции протектора, т.е. тут нет многослойных распаковщиков как например в версиях ASProtect.

2) **Пропатчить код**

В общем начинаем делать наш патч. Ищем свободное место в файле, если патч будет маленький, то можно использовать PE заголовок, если большой, то добавить свою секцию к файлу. Я решил добавить свою секцию.

Name:	VOffset:	VSize:	ROffset:	RSize:	Flags:
.inline	00048000	00001000	00020A00	00000200	E00000E0

Итак, патчим функцию таким образом:

```
004276AC  56              PUSH ESI
004276AD  51              PUSH ECX
004276AE  89C6            MOV ESI,EAX
004276B0  89D1            MOV ECX,EDX
004276B2  83E9 04         SUB ECX,4
004276B5  FC              CLD
004276B6  AC              LODS BYTE PTR DS:[ESI]
004276B7  D0E8            SHR AL,1
004276B9  80F8 74         CMP AL,74
004276BC  75 0E           JNZ SHORT Example_.004276CC
004276BE  8B06            MOV EAX,DWORD PTR DS:[ESI]
004276C0  0FC8            BSWAP EAX
004276C2  01C8            ADD EAX,ECX
004276C4  8906            MOV DWORD PTR DS:[ESI],EAX
004276C6  83C6 04         ADD ESI,4
004276C9  83E9 04         SUB ECX,4
004276CC  49              DEC ECX
004276CD  7F E7           JG SHORT Example_.004276B6
004276CF  E9 2C090200     JMP Example_.00448000
```

И собстенно как выглядит функция патча.

```
00448000  81FE FC3F4000   CMP ESI,Example_.00403FFC
00448006  75 0A           JNZ SHORT Example_.00448012
00448008  C705 07104000 1 MOV DWORD PTR DS:[401007],11111111
00448012  59              POP ECX
00448013  5E              POP ESI
00448014  C3              RETN
```

3) **Пропатчить проверки CRC (если нужно).**

Если бы нужно было патчить еще CRC, то пришлось бы вставить еще одно сравнение ESI и в том участке сделать фикс для CRC.

Сохраняем изменения, запускаем. Все работает! (Example_2.2.6_inline.exe).

Inline Patch версии 2.4.1

Вынес это в отдельный раздел, т.к. для этой версии способ инлайна немного отличается. Это связано с тем, что вышел мой анпакер для криптора на паблик и авторы криптора поспешили сделать апдейт чтобы покупатели не гневались, так вот – в анпакере функция поправки смещений искалась по сигнатуре кода, они сделали дубликат этой функции ниже и расксоривали/заксоривали ее по ходу выполнения, а та оригинальный функция не юзалась, а просто осталась как пустышка(фейк) чтобы анпакер ставил туда бряк, но выполнение кода туда не передавалось.

Поэтому берем пример (Example_2.4.1.exe) и проделываем все те же действия как для версии 2.2.6 для поиска функции поправки смешений и попадаем:

```
004236C0  56              PUSH ESI                                 FAKE FUNCTION BEGIN
004236C1  51              PUSH ECX
004236C2  89C6            MOV ESI,EAX
004236C4  89D1            MOV ECX,EDX
004236C6  83E9 04         SUB ECX,4
004236C9  FC              CLD
004236CA  AC              LODS BYTE PTR DS:[ESI]
004236CB  D0E8            SHR AL,1
004236CD  80F8 74         CMP AL,74
004236D0 v75 0E           JNZ SHORT EXAMPL~3.004236E0
004236D2  8B06            MOV EAX,DWORD PTR DS:[ESI]
004236D4  0FC8            BSWAP EAX
004236D6  01C8            ADD EAX,ECX
004236D8  8906            MOV DWORD PTR DS:[ESI],EAX
004236DA  83C6 04         ADD ESI,4
004236DD  83E9 04         SUB ECX,4
004236E0  49              DEC ECX
004236E1 ^7F E7           JG SHORT EXAMPL~3.004236CA
004236E3  59              POP ECX
004236E4  5E              POP ESI
004236E5  C3              RETN                                     FAKE FUNCTION END
004236E6  8BC0            MOV EAX,EAX
004236E8  56              PUSH ESI
004236E9  51              PUSH ECX
004236EA  89C6            MOV ESI,EAX
004236EC  89D1            MOV ECX,EDX
004236EE  83E9 04         SUB ECX,4
004236F1  FC              CLD
004236F2  AC              LODS BYTE PTR DS:[ESI]
004236F3  D0E8            SHR AL,1
004236F5  80F8 74         CMP AL,74
004236F8 v75 0E           JNZ SHORT EXAMPL~3.00423708
004236FA  8B06            MOV EAX,DWORD PTR DS:[ESI]
004236FC  0FC8            BSWAP EAX
004236FE  01C8            ADD EAX,ECX
00423700  8906            MOV DWORD PTR DS:[ESI],EAX
00423702  83C6 04         ADD ESI,4
00423705  83E9 04         SUB ECX,4
00423708  49              DEC ECX
00423709 ^7F E7           JG SHORT EXAMPL~3.004236F2
0042370B  59              POP ECX
0042370C  5E              POP ESI
0042370D  C3              RETN
```

Как видим с адреса 004236C0 до 004236E5 идет фейковая функция.

А вот ниже идет нужная нам функция поправки смещений, только этот код перед выполнением расксоривается, а после выполения опять заксоривается.

Поэтому нам нужно пропатчить ее как и в предыдущем примере, далее дождаться пока криптор ее заксорит и уже тот код сохранить.

```
004236E6  8BC0            MOV EAX,EAX
004236E8  56              PUSH ESI
004236E9  51              PUSH ECX
004236EA  89C6            MOV ESI,EAX
004236EC  89D1            MOV ECX,EDX
004236EE  83E9 04         SUB ECX,4
004236F1  FC              CLD
004236F2  AC              LODS BYTE PTR DS:[ESI]
004236F3  D0E8            SHR AL,1
004236F5  80F8 74         CMP AL,74
004236F8 v75 0E           JNZ SHORT Example_.00423708
004236FA  8B06            MOV EAX,DWORD PTR DS:[ESI]
004236FC  0FC8            BSWAP EAX
004236FE  01C8            ADD EAX,ECX
00423700  8906            MOV DWORD PTR DS:[ESI],EAX
00423702  83C6 04         ADD ESI,4
00423705  83E9 04         SUB ECX,4
00423708  49              DEC ECX
00423709 ^7F E7           JG SHORT Example_.004236F2
0042370B -E9 F0F80100     JMP Example_.00443000
00423710  0010            ADD BYTE PTR DS:[EAX],DL
```

Пропатчили (предполагаем что код патча по адресу 443000 в новой секции уже есть), ставим бряк на доступ к памяти 42370B. И попали вот сюда:

```
00440335  3010            XOR BYTE PTR DS:[EAX],DL
00440337  68 9C064400     PUSH Example_.0044069C
0044033C ^E9 44D1FFFF     JMP Example_.0043D485
00440341  0000            ADD BYTE PTR DS:[EAX],AL
00440343  13FE            ADC EDI,ESI
00440345  893C24          MOV DWORD PTR SS:[ESP],EDI
00440348  5F              POP EDI
00440349  99              CDQ
0044034A  F7F9            IDIV ECX
0044034C  51              PUSH ECX
0044034D  68 45ED4300     PUSH Example_.0043ED45
```

```
0042370B  E9 F0 F8 01 00  00 10 40  00 17 55 8B  EC 51 C7 45  йрш┌.  .┼@.┤U<мQ3E
0042371B  FC 22 40 00  13 81 7D FC  11 40 00 1A  75 16 6A 40  ь"@.  ‼Ґ}ъ◄@.→u┬j@
0042372B  68 30 50 40  00 68 44 40  04 1D 6A 00  FF 15 94 40  h0P@.  hD@♦.j.я┴″@
0042373B  40 00 EB 14  6A 10 68 54  60 15 10 68  27 15 00 02  @.л¶j►hT`§►h'┴.┐
```

Далее потрейсим этот цикл до конца его выполнения, а точнее до:

```
00440696  57              PUSH EDI
00440697 ^E9 1DE3FFFF     JMP Example_.0043E9B9
0044069C  40              INC EAX
0044069D  FEC2            INC DL
0044069F  49              DEC ECX
004406A0 ^0F85 08EBFFFF   JNZ Example_.0043F1AE
004406A6  58              POP EAX                  Example_.004255FD
004406A7 ^E9 EBB7FFFF     JMP Example_.0043BE97
004406AC  0000            ADD BYTE PTR DS:[EAX],AL
004406AE  68 87134400     PUSH Example_.00441387
```

```
0042370B  CD D5 DE 26 28  00 10 40  00 17 55 8B  EC 51 C7 45  НХЮ&(  .┼@.┤U<мQ3E
0042371B  FC 22 40 00  13 81 7D FC  11 40 00 1A  75 16 6A 40  ь"@.  ‼Ґ}ъ◄@.→u┬j@
0042372B  68 30 50 40  00 68 44 40  04 1D 6A 00  FF 15 94 40  h0P@.  hD@♦.j.я┴″@
0042373B  40 00 EB 14  6A 10 68 54  60 15 10 68  27 15 00 02  @.л¶j►hT`§►h'┴.┐
```

Как видим, криптор заксорил код с нашим джампом теперь этот код и нужно нам сохранить чтобы при следующем запуске он его раскорил и там уже был наш патч.

Сохраняем эти 5 байт + код патча в новой секции, запускаем и видим что работает!

Ну вот и Все!!! ;)