

Swat It Pro V2.1

<http://swatit.org/download.html>

In-line patching tutorial with and without adding section

1. Target analysis

As usual we have to check the target executable in order to see if this is packed:

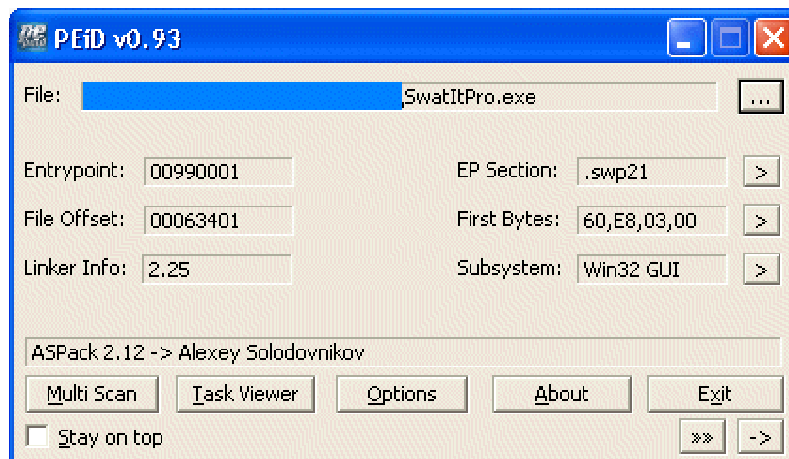


Fig. 1 PEiD scanning

Now we can use Stripper V2.07 final to unpack the target, manual unpack is also very simple to do and will not cover in this tutorial because we are focused into the in-line patching theme.

About in-line patching we will cover two different techniques:

- In-line patching with adding a section
- In-line patching into the main code (without adding a section)

The first one is useful when you have a big size in patching or simply when you have not enough space into the target to fit your patching code, this solution give flexibility into the patching but have the drawback to target size increasing.

The last method keep the target size untouched and usually is useful for small patching.

All this things is developed with a real case packed with Aspack but the concept is more general and can be applied also to other packer like UPX, ASProtect, Armadillo and so on.

If you run the target **SwatItPro.exe**, we can see the EVALUATION text string and the trial day counter into the top of the window:

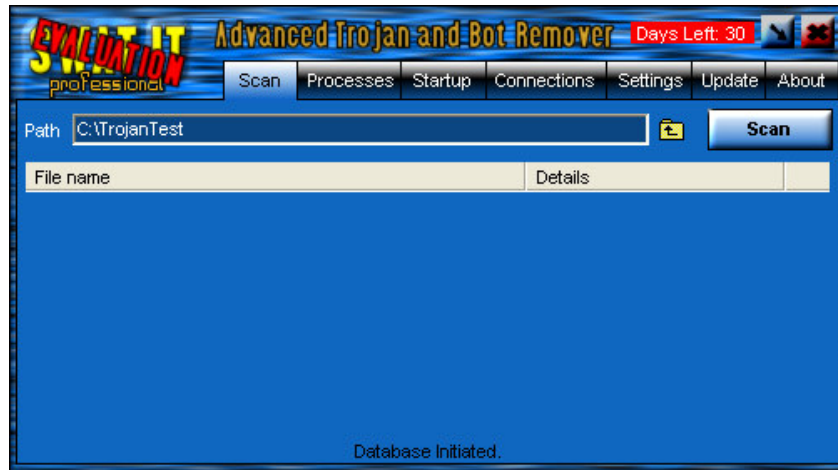


Fig. 2 Scan dialog.

Select also the update option and see the purchasing/registering button at the bottom:

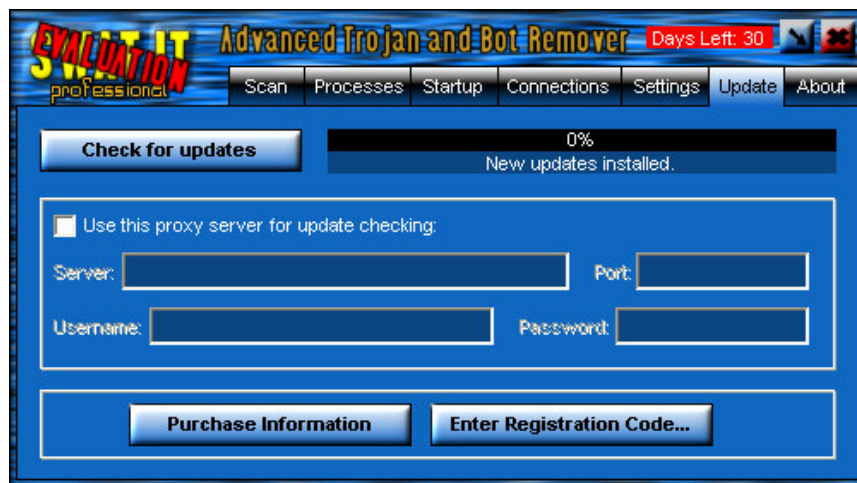


Fig. 3 Update/registration dialog window.

If you try with a fake code like this one:

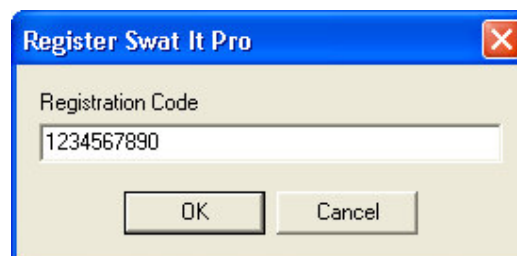


Fig. 4 Fake code at the registration dialog.

And press the OK button.

Yup we keep a thank you message box:



Fig. 5

but no pain :-P really the target isn't registered, the EVALUATION text and the day counter is still present, also the registering button is untouched and if you restart the target again we have the trial calculation and the EVALUATION text string inside the window.

2. Cracking stage (trial time defeating and registration)

In this section you can see some patching code, first about the trial limit defeating and finally about the target registration, I'll not cover in great detail this steps because the purpose of this tutorial is about in-line patching, but cracking is really simple.

As the first point we can search about the trial time expiration check, this one is easily to found, simply load the target in OllyDbg and run it, when you get the main dialog go to the OEP at 0x004C227C and perform a search about **all referenced text string** (right click) with text EXPIRED:

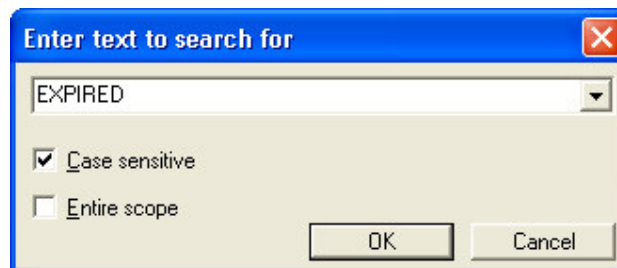


Fig. 6 Text to search.

004B9931	PUSH SwatItPr.004B99A4	ASCII "This evaluation version of Swat It Pro has expired
004B9958	PUSH SwatItPr.004B9A98	ASCII "button on the Update tab"
004B9D55	MOV EDX,SwatItPr.004B9DA4	ASCII "EXPIRED"
004B9D70	MOV EDX,SwatItPr.004B9DB4	ASCII "Days Left; "
004B9DF3	MOV EDX,SwatItPr.004B9E20	ASCII "Software\Microsoft\Windows\CurrentVersion\Run"

Fig. 7

Double click on the entry, you land in 0x004B9D55, scroll up a little into the code then place a breakpoint into the 0x004B9CD8 address, with the following patch the trial time will be fixed to 30 days forever.

```

004B9008 $ E8 2B13F5FF CALL SwatItPr.0040B008
004B9009 . DC25 687ECF00 FSUB QWORD PTR DS:[CF7E68]
004B900A . E8 6C90F4FF CALL SwatItPr.00402D54
004B900B . 50 PUSH EAX
004B900C . B8 1E000000 MOV EAX,1E
004B900D . 5A POP EDX
004B900E . 33D2 XOR EDX,EDX
004B900F . C3 RETN
004B9010 . 8BC0 MOV EAX,EAX
004B9011 . 53 PUSH EBX
004B9012 . 33D8 XOR EBX,EBX
004B9013 . 833D F0534C00 CMP DWORD PTR DS:[4C53F0],0
004B9014 . 74 09 JE SHORT SwatItPr.004B9D09
004B9015 . 833D F4534C00 CMP DWORD PTR DS:[4C53F4],0
004B9016 . 75 1D JNZ SHORT SwatItPr.004B9D26
004B9017 . E8 CAFFFFFF CALL SwatItPr.004B9C08
004B9018 . 85C0 TEST EAX,EAX
004B9019 . 7C 09 JL SHORT SwatItPr.004B9D1B
004B901A . 83F8 1E CMP EAX,1E
004B901B . 7F 04 JG SHORT SwatItPr.004B9D1B
004B901C . 33D8 XOR EBX,EBX
004B901D . EB 02 JMP SHORT SwatItPr.004B9D1D
004B901E . B3 01 MOV BL,1
004B901F . 84D8 TEST BL,BL
004B9020 . 74 05 JE SHORT SwatItPr.004B9D26
004B9021 . E8 F6FBFFFF CALL SwatItPr.004B991C
004B9022 . 8BC3 MOV EAX,EBX
004B9023 . 5B POP EBX
004B9024 . C3 RETN
004B9025 . 8BC0 MOV EAX,EAX
004B9026 $ 55 PUSH EBP
004B9027 . 8BEC MOV EBP,ESP
004B9028 . 6A 00 PUSH 0
004B9029 . 53 PUSH EBX
004B902A . 56 PUSH ESI
004B902B . 8BF0 MOV ESI,EAX
004B902C . 33C0 XOR EAX,EAX
004B902D . 55 PUSH EBP
004B902E . 68 909D4B00 PUSH SwatItPr.004B9D90
004B902F . 64:FF30 PUSH DWORD PTR FS:[EAX]
004B9030 . 64:8920 MOV DWORD PTR FS:[EAX],ESP
004B9031 . E8 90FFFFFF CALL SwatItPr.004B9C08
004B9032 . 8BD8 MOV EBX,EAX
004B9033 . 85D8 TEST EBX,EBX
004B9034 . 7C 05 JL SHORT SwatItPr.004B9D53
004B9035 . 83FB 1E CMP EBX,1E
004B9036 . 7E 0E JLE SHORT SwatItPr.004B9D61
004B9037 . 8BC6 MOV EAX,ESI
004B9038 . BA A49D4B00 MOV EDI,SwatItPr.004B9DA4
004B9039 . E8 B1ACF4FF CALL SwatItPr.00404A10
004B903A . EB 19 JMP SHORT SwatItPr.004B9D7A
004B903B . 8D55 FC LEA EDI,DWORD PTR SS:[EBP-4]
004B903C . 8BC3 MOV EAX,EBX
004B903D . E8 89F1F4FF CALL SwatItPr.00408EF4
004B903E . 8B4D FC MOV ECX,DWORD PTR SS:[EBP-4]
004B903F . 8BC6 MOV EAX,ESI
004B9040 . BA B49D4B00 MOV EDI,SwatItPr.004B9DB4
004B9041 . E8 4EAF4FF CALL SwatItPr.00404CC8
004B9042 . 33C0 XOR EAX,EAX
004B9043 . 5A POP EDX
004B9044 . 59 POP ECX
004B9045 . 59 POP ECX
004B9046 . 64:8910 MOV DWORD PTR FS:[EAX],EDX
004B9047 . 68 979D4B00 PUSH SwatItPr.004B9D97
004B9048 . 8D45 FC LEA EAX,DWORD PTR SS:[EBP-4]
004B9049 . E8 2DACF4FF CALL SwatItPr.004049BC
004B904A . C3 RETN
004B904B . E9 A7A5F4FF JMP SwatItPr.0040433C

```

TRIAL TIME CALCULATION

Checking residual trial days

Trial time calculation

ASCII "EXPIRED"

EBX = residual trial days

ASCII "Days Left: "

Fig. 8 Trial time patching.

As our final goal we can keep the target registered.

This task is easily solved by a simple backtracing from the trial time checking routine, the registration checking will be before the expiration checking and we have:

```

004BEB47 00 DE 00      DB 00
004BEB48 55          PUSH EBP
004BEB49 8BEC      MOV EBP,ESP
004BEB4B 6A 00      PUSH 0
004BEB4D 6A 00      PUSH 0
004BEB4F 6A 00      PUSH 0
004BEB51 53          PUSH EBX
004BEB52 56          PUSH ESI
004BEB53 8BF2      MOV ESI,EDX
004BEB55 8BD8      MOV EBX,EAX
004BEB57 33C0      XOR EAX,EAX
004BEB59 55          PUSH EBP
004BEB5A 68 A2EC4B00 PUSH SwatItPr.004BEC42
004BEB5F 64:FF30    PUSH DWORD PTR FS:[EAX]
004BEB62 64:8920    MOV DWORD PTR FS:[EAX],ESP
004BEB65 85F6      TEST ESI,ESI
004BEB67 90          NOP
004BEB68 90          NOP
004BEB69 8D45 FC    LEA EAX,DWORD PTR SS:[EBP-4]
004BEB6C 50          PUSH EAX
004BEB6D 8BD6      MOV EDI,ESI
004BEB6F 68 B8EC4B00 MOV EAX,SwatItPr.004BECB8
004BEB74 68 4764F4FF CALL SwatItPr.00404FC0
004BEB79 8BC8      MOV ECX,EAX
004BEB7B 49          DEC ECX
004BEB7C BA 01000000 MOV EDI,1
004BEB81 8BC6      MOV EAX,ESI
004BEB83 68 5463F4FF CALL SwatItPr.00404EDC
004BEB88 8D45 F8    LEA EAX,DWORD PTR SS:[EBP-8]
004BEB8B 50          PUSH EAX
004BEB8C 8BD6      MOV EDI,ESI
004BEB8E 68 B8EC4B00 MOV EAX,SwatItPr.004BECB8
004BEB93 68 2864F4FF CALL SwatItPr.00404FC0
004BEB98 40          INC EAX
004BEB99 50          PUSH EAX
004BEB9A 8BC6      MOV EAX,ESI
004BEB9C 68 DB60F4FF CALL SwatItPr.00404C7C
004BEBA1 8BC8      MOV ECX,EAX
004BEBA3 8BC6      MOV EAX,ESI
004BEBA5 50          POP EDI
004BEBA6 68 3163F4FF CALL SwatItPr.00404EDC
004BEBAB 68 58564C00 MOV EAX,DWORD PTR DS:[4C5658]
004BEBB0 8B55 FC    MOV EDI,DWORD PTR SS:[EBP-4]
004BEBB3 68 585EF4FF CALL SwatItPr.00404A10
004BEBB8 68 B4564C00 MOV EAX,DWORD PTR DS:[4C56B4]
004BEBBD 8B55 F8    MOV EDI,DWORD PTR SS:[EBP-8]
004BEBC0 68 4B5EF4FF CALL SwatItPr.00404A10
004BEBC5 68 A6B3FFFF CALL SwatItPr.0040B9F0
004BEBCA 68 A1B3FFFF CALL SwatItPr.0040B9F0
004BEBCF > A1 58564C00 MOV EAX,DWORD PTR DS:[4C5658]
004BEBD4 8338 00    CMP DWORD PTR DS:[EAX],0
004BEBD7 < 74 0A     JE SHORT SwatItPr.004BEBE3
004BEBD9 > A1 B4564C00 MOV EAX,DWORD PTR DS:[4C56B4]
004BEBDE 8338 00    CMP DWORD PTR DS:[EAX],0
004BEBE1 < 75 04     JNZ SHORT SwatItPr.004BEBE7
004BEBE3 > 33C0      XOR EAX,EAX
004BEBE5 90          NOP
004BEBE6 90          NOP
004BEBE7 > B0 01     MOV AL,1
004BEBE9 > 84C0      TEST AL,AL
004BEBEB < 74 43     JE SHORT SwatItPr.004BEC30
  
```

Fig. 9 Registration patching

Then restart the target with CTRL+F2 and when you reach the OEP simply apply this patches.

YUP the target is still registered, look at the update window:

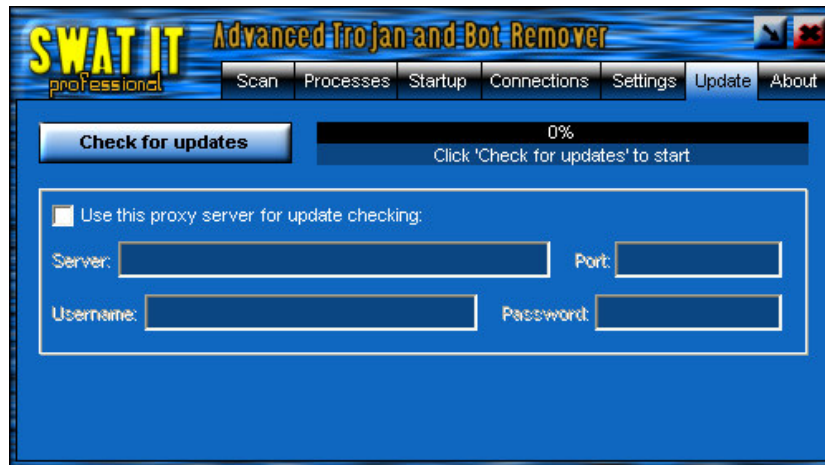


Fig. 10 Update window after patching.

3. In-line patching with adding section

Now we can think to make a patching by using the in-line patching technique, first we've to find the point related the OEP jumping, this is simple to found, place a breakpoint into the VirtualAlloc API function, after some break and simple tracing with F8 we land in this interesting code:

00D90376	8907	MOV DWORD PTR DS:[EDI],EAX	
00D90378	8385 49050000	ADD DWORD PTR SS:[EBP+549],4	
00D9037F	^ E9 32FFFFFF	JMP SwatItPr.00D902B6	
00D90384	8906	MOV DWORD PTR DS:[ESI],EAX	
00D90386	8946 0C	MOV DWORD PTR DS:[ESI+C],EAX	
00D90389	8946 10	MOV DWORD PTR DS:[ESI+10],EAX	
00D9038C	83C6 14	ADD ESI,14	
00D9038F	8B95 22040000	MOV EDX,DWORD PTR SS:[EBP+422]	
00D90395	^ E9 EBFFFFFF	JMP SwatItPr.00D90285	
00D9039A	8B 7C220C00	MOV EAX,0C227C	
00D9039F	50	PUSH EAX	
00D903A0	8385 22040000	ADD EAX,DWORD PTR SS:[EBP+422]	
00D903A6	59	POP ECX	
00D903A7	0BC9	OR ECX,ECX	
00D903A9	8985 A8030000	MOV DWORD PTR SS:[EBP+3A8],EAX	
00D903AF	61	POPAD	
00D903B0	^ 75 08	JNZ SHORT SwatItPr.00D903BA	
00D903B2	B8 01000000	MOV EAX,1	
00D903B7	C2 0C00	RETN 0C	
00D903BA	68 7C224C00	PUSH SwatItPr.004C227C	
00D903BF	C3	RETN	
00D903C0	8B85 26040000	MOV EAX,DWORD PTR SS:[EBP+426]	
00D903C6	8D8D 3B040000	LEA ECX,DWORD PTR SS:[EBP+43B]	
00D903CC	51	PUSH ECX	
00D903CD	50	PUSH EAX	
00D903CE	FF95 490F0000	CALL NEAR DWORD PTR SS:[EBP+F49]	
00D903D4	8985 55050000	MOV DWORD PTR SS:[EBP+555],EAX	
00D903DA	8D85 47040000	LEA EAX,DWORD PTR SS:[EBP+447]	

Write OEP in 00D903BA

OEP=00D903BA
RETN used as a jump to the OEP

Fig. 11 Code before reach the OEP.

after the RETN instruction we're into the OEP:

```

004C2270 55          PUSH EBP
004C2271 8BEC       MOV EBP, ESP
004C2272 83C4 F0    ADD ESP, -10
004C2273 B0 A41E4C00 MOV EAX, SwatItPr.004C1E94
004C2274 E8 9448F4FF CALL SwatItPr.00406B20
004C2275 A1 04584C00 MOV EAX, DWORD PTR DS:[4C5804]
004C2276 8B00       MOV EAX, DWORD PTR DS:[EAX]
004C2277 E8 3400FBFF CALL SwatItPr.0047F2C0
004C2278 A1 04584C00 MOV EAX, DWORD PTR DS:[4C5804]
004C2279 8B00       MOV EAX, DWORD PTR DS:[EAX]
004C227A BA DC224C00 MOV EDI, SwatItPr.004C22DC
004C227B E8 18CCFBFF CALL SwatItPr.0047EEC4
004C227C 8B00 58554C00 MOV ECX, DWORD PTR DS:[4C5558]
004C227D A1 04584C00 MOV EAX, DWORD PTR DS:[4C5804]
004C227E 8B00       MOV EAX, DWORD PTR DS:[EAX]
004C227F 8B15 C0B14B00 MOV EDI, DWORD PTR DS:[4BB1C0]
004C2280 E8 2300FBFF CALL SwatItPr.0047F2E4
004C2281 A1 04584C00 MOV EAX, DWORD PTR DS:[4C5804]
004C2282 8B00       MOV EAX, DWORD PTR DS:[EAX]
004C2283 E8 9700FBFF CALL SwatItPr.0047F364
004C2284 E8 7625F4FF CALL SwatItPr.00404848
004C2285 0000       ADD BYTE PTR DS:[EAX], AL
004C2286 FFFF       ???
004C2287 FFFF       ???
004C2288 0B00       OR EAX, DWORD PTR DS:[EAX]
004C2289 0000       ADD BYTE PTR DS:[EAX], AL
004C228A 53         PUSH EBX
004C228B 77 61      JA SHORT SwatItPr.004C2340
  
```

Fig. 12 Code at OEP.

Well now restart the target with CTRL+F2 and press CTRL+G then write the address

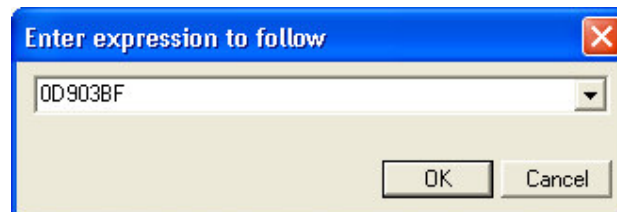


Fig. 13

This is the code:

```

0009037F ^ E9 32FFFFFF JMP SwatItPr.000902B6
00090380 8906       MOV DWORD PTR DS:[ESI], EAX
00090381 8946 0C    MOV DWORD PTR DS:[ESI+C], EAX
00090382 8946 10    MOV DWORD PTR DS:[ESI+10], EAX
00090383 83C6 14    ADD ESI, 14
00090384 8B95 22040000 MOV EDI, DWORD PTR SS:[EBP+4221]
00090385 ^ E9 EBFFFFFF JMP SwatItPr.000902B5
00090386 B8 7C220C00 MOV EAX, 0C227C
00090387 50         PUSH EAX
00090388 0395 22040000 ADD EAX, DWORD PTR SS:[EBP+4221]
00090389 59         POP ECX
0009038A 0BC9       OR ECX, ECX
0009038B 0905 A0030000 MOV DWORD PTR SS:[EBP+3A0], EAX
0009038C 61         POPAD
0009038D 75 08     JNZ SHORT SwatItPr.000903BA
0009038E B8 01000000 MOV EAX, 1
0009038F C2 0C00    RETN 0C
00090390 68 00000000 PUSH 0
00090391 C3         RETN
00090392 8B85 26040000 MOV EAX, DWORD PTR SS:[EBP+426]
00090393 8D8D 3B040000 LEA ECX, DWORD PTR SS:[EBP+43B]
00090394 51         PUSH ECX
  
```

Fig. 14

Now we can think to put the redirection to our patching code at the 0x00D903B0 address then we can change the JNZ with a JMP to our cave code.

Writing the patch require also find some free space inside the target I choose to add a new section and place the code into this new section.

To add a section simply press ALT+M, now we can choose a section to use as a sample for the new section that we will go to add later, if you search a useful free section with small size (0x1000) is the .tls section, then double click into the highlighted section and dump all the section to hard-disk.

003B0000	00002000	Swat ItPr			Map	R	R
00400000	00001000	Swat ItPr	CODE	PE header	Imag	R	RWE
00401000	00002000	Swat ItPr	code	code	Imag	R	RWE
004C0000	00003000	Swat ItPr	DATA	data	Imag	R	RWE
004C6000	00003000	Swat ItPr	BSS		Imag	R	RWE
00CF8000	00003000	Swat ItPr	.idata		Imag	R	RWE
00CFB000	00001000	Swat ItPr	.tls		Imag	R	RWE
00CFC000	00001000	Swat ItPr	.rdata		Imag	R	RWE
00CFD000	0000E000	Swat ItPr	.reloc		Imag	R	RWE
00D0E000	00005000	Swat ItPr	.rsrc	resources	Imag	R	RWE
00D90000	00003000	Swat ItPr	.swp21	SFX, imports	Imag	R	RWE
00D93000	00001000	Swat ItPr	.adata		Imag	R	RWE
00D94000	00001000	Swat ItPr	InLine		Imag	R	RWE
00DA0000	00004000				Map	R E	R E

Fig. 15 Redirection.

Now you should have a file namely **SwatItPr_00CFB000.mem** related to the dumped memory section.

After that open PEditor and load the original (then the packed) target into it:

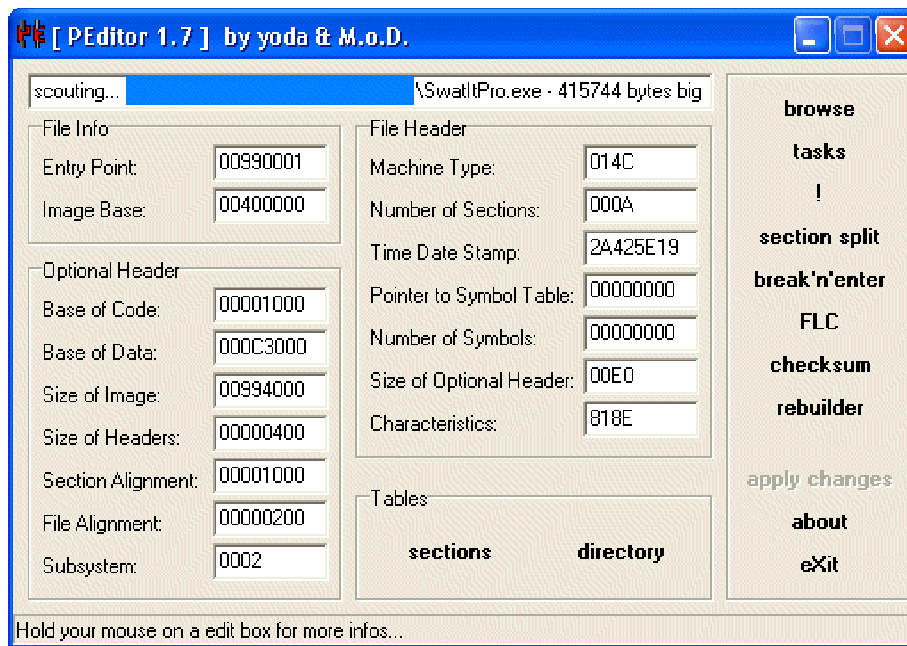


Fig. 16

Now press the **sections** button and scroll down in order to see the last section:

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
BSS	00832000	000C6000	00000000	0004B800	C0000040
.idata	00003000	008F8000	00001000	0004B800	C0000040
.tls	00001000	008FB000	00000000	0004C800	C0000040
.rdata	00001000	008FC000	00000200	0004C800	C0000040
.reloc	0000E000	008FD000	00000000	0004CA00	C0000040
.rsrc	00005000	0090B000	00016A00	0004CA00	C0000040
.swp21	00003000	00990000	00002400	00063400	C0000040
.adata	00001000	00993000	00000000	00065800	C0000040

Fig. 17

Now right click and choose the **copy a section from HD to EOF** option:

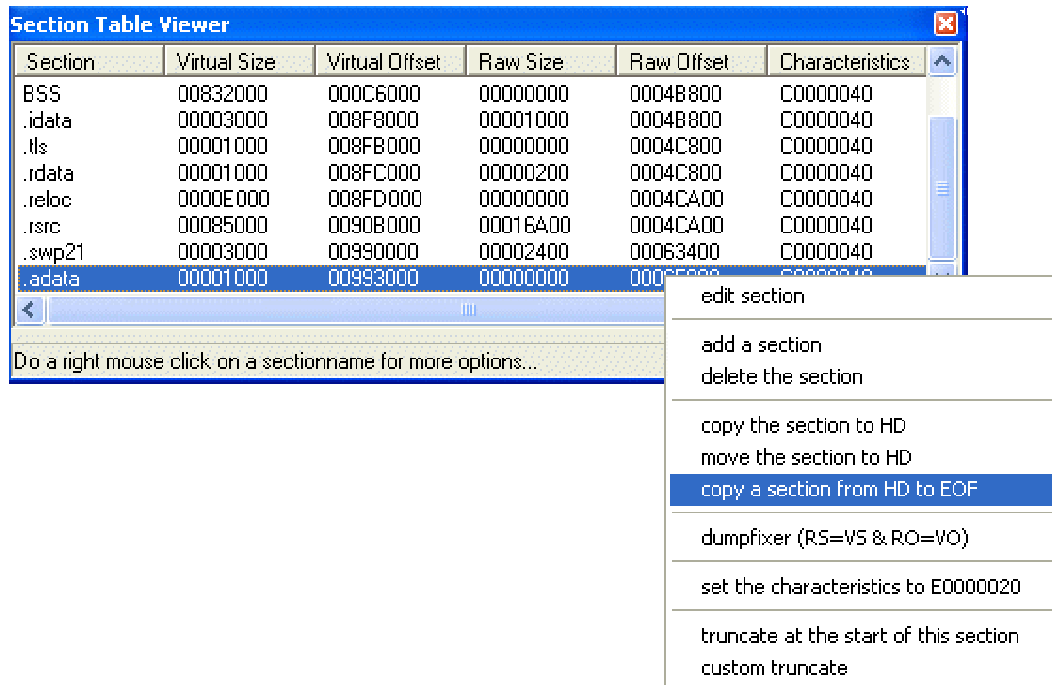


Fig. 18 Select to add a new section from the end of file.

And select the previously dumped section:

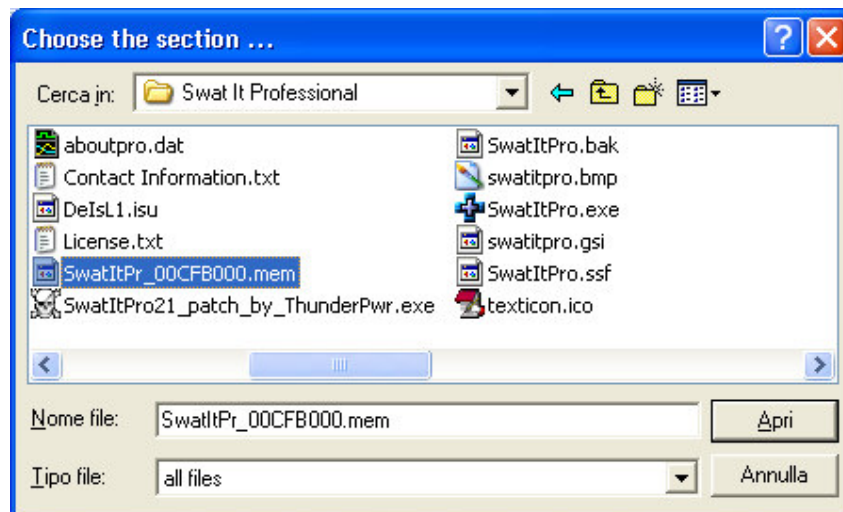


Fig. 19 Load the new section to add.

And press Yes in order to update the PE Header:

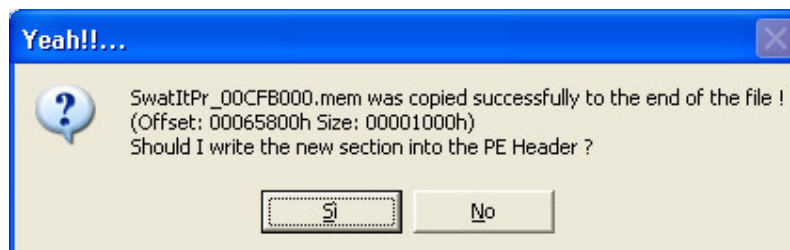
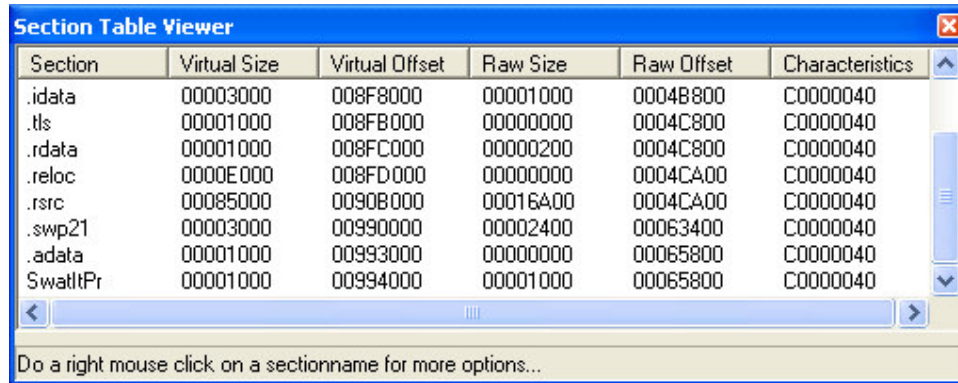


Fig. 20 Confirm the PE Header changing.

Now take a look again into the section viewer:

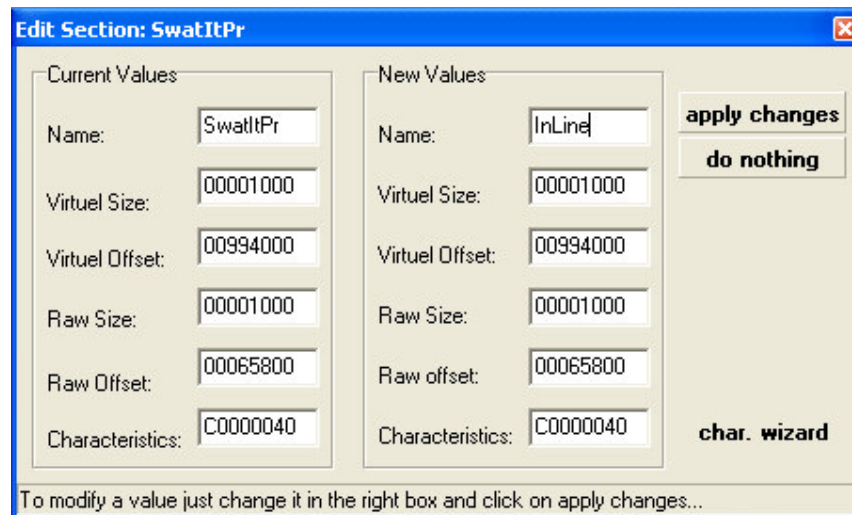


Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
.idata	00003000	008F8000	00001000	0004B800	C0000040
.tls	00001000	008FB000	00000000	0004C800	C0000040
.rdata	00001000	008FC000	00000200	0004C800	C0000040
.reloc	0000E000	008FD000	00000000	0004CA00	C0000040
.rsrc	00085000	0090B000	00016A00	0004CA00	C0000040
.swp21	00003000	00900000	00002400	00063400	C0000040
.adata	00001000	00993000	00000000	00065800	C0000040
SwatItPr	00001000	00994000	00001000	00065800	C0000040

Do a right mouse click on a sectionname for more options...

Fig. 21 The new section added

A new section namely SwatItPr was successfully added with size of 0x1000, now we can perform a cosmetics work and rename the new section with a better name like **InLine**. Simply select the new section and right click then choose the **edit section** option:



Current Values		New Values		
Name:	SwatItPr	Name:	InLine	<input type="button" value="apply changes"/> <input type="button" value="do nothing"/> char. wizard
Virtual Size:	00001000	Virtual Size:	00001000	
Virtual Offset:	00994000	Virtual Offset:	00994000	
Raw Size:	00001000	Raw Size:	00001000	
Raw Offset:	00065800	Raw offset:	00065800	
Characteristics:	C0000040	Characteristics:	C0000040	

To modify a value just change it in the right box and click on apply changes...

Fig. 22 Change the section name.

After changing the section name simply press the **apply changes** button:



Fig. 23

press OK.

Before test the target we have to perform a last operation related the target rebuilding, to do that simply press the **rebuilder** button into the PEditor main window and sure about this settings:

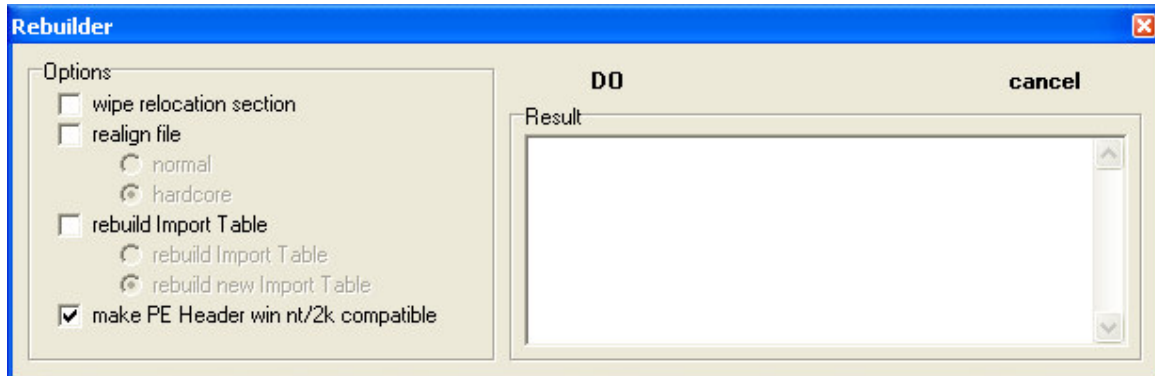


Fig. 24 Rebuilder window

Press **DO**:

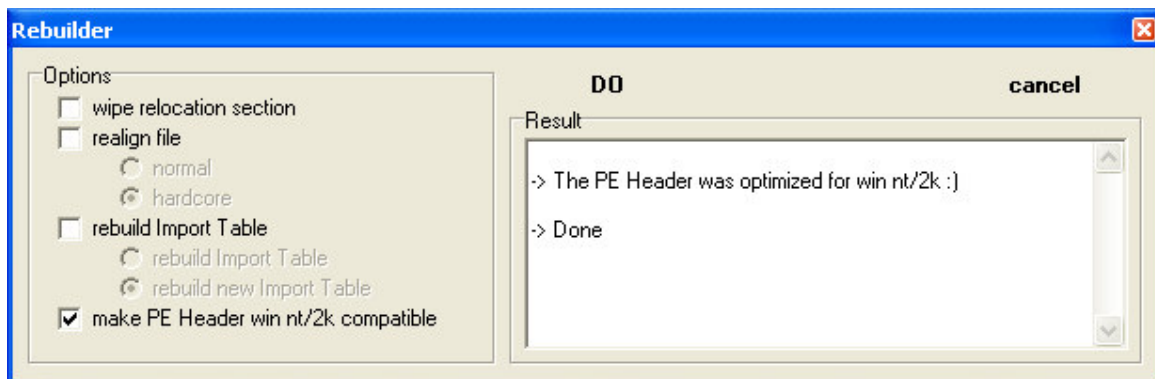


Fig. 25 Rebuilder window

In order to check our work until this point close the PEditor and run the rebuilt target with the new section added:



Fig. 26 Main screen after adding a new section.

Yup the target run right, this means no CRC control about our new section then we can go straight to the next section related to the injected code.

Load the target in OllyDbg and press ALT+M:

003B0000	00002000				Map	R	R
00400000	00001000	SwatItPr		PE header	Image	R	RWE
00401000	00002000	SwatItPr	CODE	code	Image	R	RWE
00403000	00003000	SwatItPr	DATA	data	Image	R	RWE
00406000	00003200	SwatItPr	BSS		Image	R	RWE
00CF8000	00003000	SwatItPr	.idata		Image	R	RWE
00CFB000	00001000	SwatItPr	.tls		Image	R	RWE
00CFC000	00001000	SwatItPr	.rdata		Image	R	RWE
00CFD000	0000E000	SwatItPr	.reloc		Image	R	RWE
00D0B000	00005000	SwatItPr	.rsxc	resources	Image	R	RWE
00D09000	00003000	SwatItPr	.swp21	SFX, imports	Image	R	RWE
00D03000	00001000	SwatItPr	.addata		Image	R	RWE
00D94000	00001000	SwatItPr	InLine		Image	R	RWE
00D90000	00005000				Map	R	R E

Fig. 27 The new section added at the target.

First we have to write the patching code, that simply is:

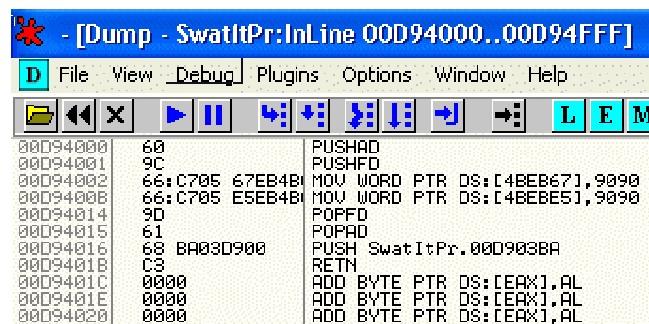


Fig. 28 Patching code.

This code save the registers and the flag before make the patch then the patch of figure 9 will be applied and finally the flag and the registers is restored (really we don't have to do this work because our patching don't need the use of register but if we think in a more general way this is a clean and safe method to inject some code without get side effects), then we have to return into the main code and this is performed by pushing into the stack the return address and then using a RETN instruction like a JMP instruction.

From the main code we have to redirect execution to our patch cave:

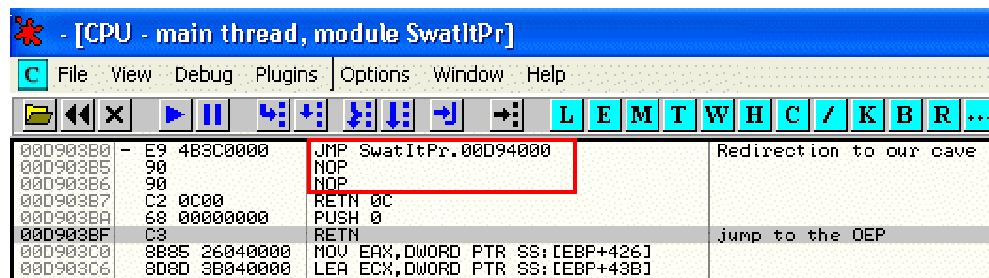


Fig. 29 Redirection to our cave.

Again we can restore the original code at the redirection, but really in this target this isn't important because we don't have any CRC or similar integrity check then we can simply return after the redirection.

Now save all the patching and test the target:



Fig. 30 Starting window after in-line patching.

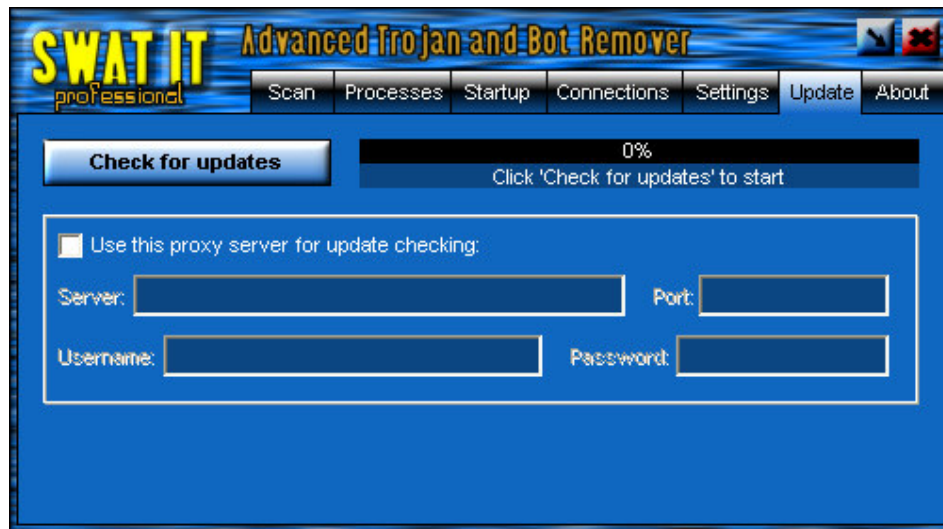


Fig. 31 Update window after in-line patching.

Job done!

4. In-line patching without need to adding a new section

Another approach is about searching available space for our patching cave inside the executable without need to add a new section, this keep unchanged the file size (just the size, CRC will change of course like to the adding section method).

To search free space inside the target we can use the ToPo tools:

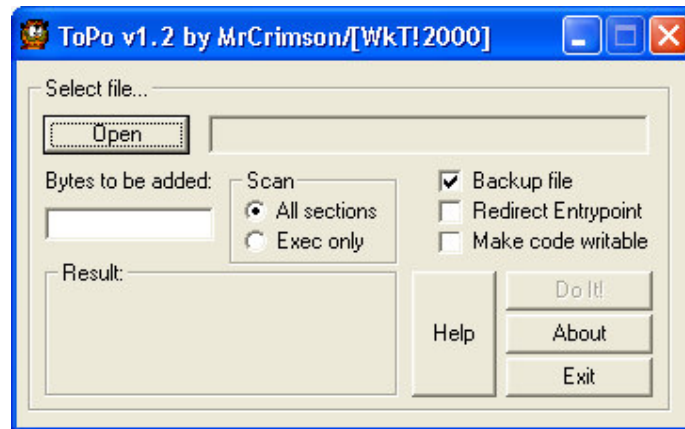


Fig. 32

Press Open:

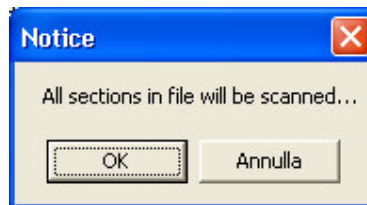


Fig. 33

Press OK and pick the file to scan:

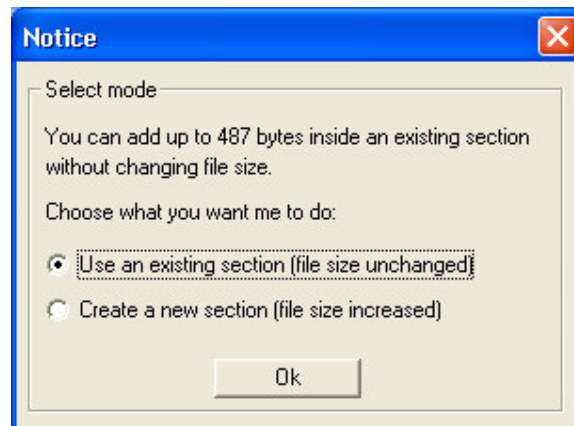


Fig. 34

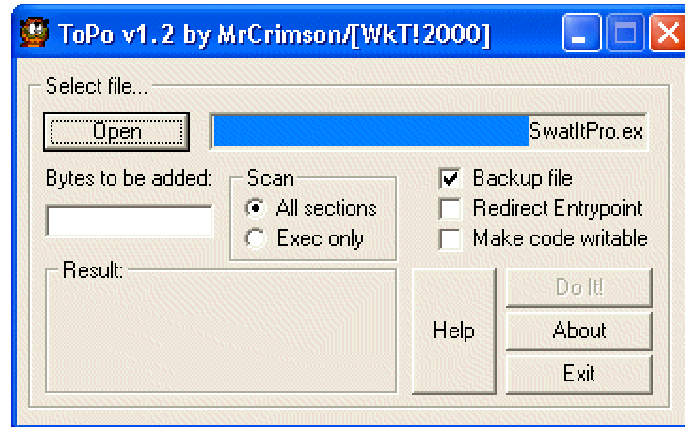


Fig. 35

We need a space for a patch of 112 bytes in length and we can perform this search this into the Executable section only:

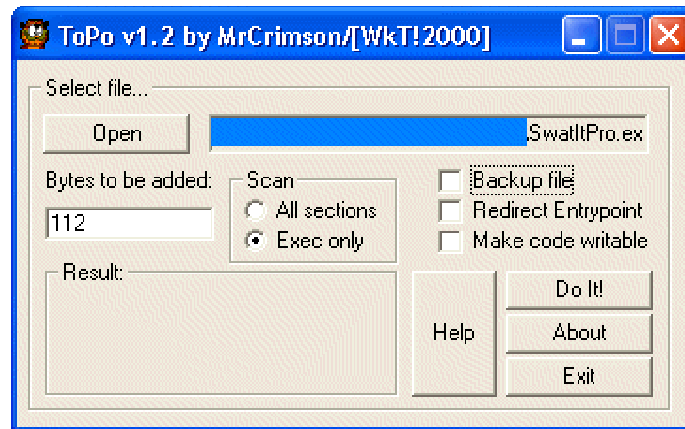


Fig. 36 Cave option

press the **Do it!** button:



Fig. 37 Cave offset.

From ToPo we have that our cave can start from 0x00CFC019 and may be 112 bytes in length.

Close ToPo and load the packed executable in OllyDbg and go to the address 0x00CFC019 and write the patching code:

00CFC018	0090 90909090	ADD BYTE PTR DS:[EAX+90909090],DL
00CFC01E	60	PUSHAD
00CFC01F	9C	PUSHFD
00CFC020	66:C705 67EB4B	MOV WORD PTR DS:[48EB67],9090
00CFC029	66:C705 E5EB4B	MOV WORD PTR DS:[48EBE5],9090
00CFC032	90	POPAD
00CFC033	61	POPAD
00CFC034	68 BA03D900	PUSH SwatItPr.00D903BA
00CFC039	C3	RETN
00CFC03A	90	NOP

Fig. 38 Patching code into the cave.

also write the redirection code:

00D903E0	E9 69BCF6FF	JMP SwatItPr.00CFC01E	Jump to our patch cave
00D903E5	90	NOP	
00D903E6	90	NOP	
00D903E7	C2 0C00	RETN 0C	
00D903E8	68 00000000	PUSH 0	
00D903EF	C3	RETN	

Fig. 39 Redirection to our patching cave.

Job done!

5. Conclusion

I hope this can be useful to learn something about in-line patching and how this can be made by adding a section or simply by searching some free space into the original packed executable and finally, of course, this tutorial show how you can in-line ASPack packed target.

Remember also the main things:

If you like this software you have to BUY IT or simply uninstall it after expiration.
This tutorial are used for educational purposes only.

6. Greetings

Thanks to the whole ARTeam:

**[Nilrem] [JDog45] [Shub - Nigurrath] [MaDMAn_H3rCuL3s] [Ferrari]
[Kruger] [Teerayoot] [R@dier] [ThunderPwr] [Eggi] [EJ12N]
[Gabri3I][Stickman 373] [Bone Enterprise] [JohnWho] [Condzero]**

Thanks to all the people who take time to write tutorials.

Thanks to all the people who continue to develop better tools.

Thanks to Exetools, Woodmann, SND, TSRH, MP2K and all the others for being a great place of learning.

Thanks also to The Codebreakers Journal, and the Anticrack forum.

As usual, if you have any suggestions, comments or corrections feel free to contact me in usual places.

Byez

ThunderPwr