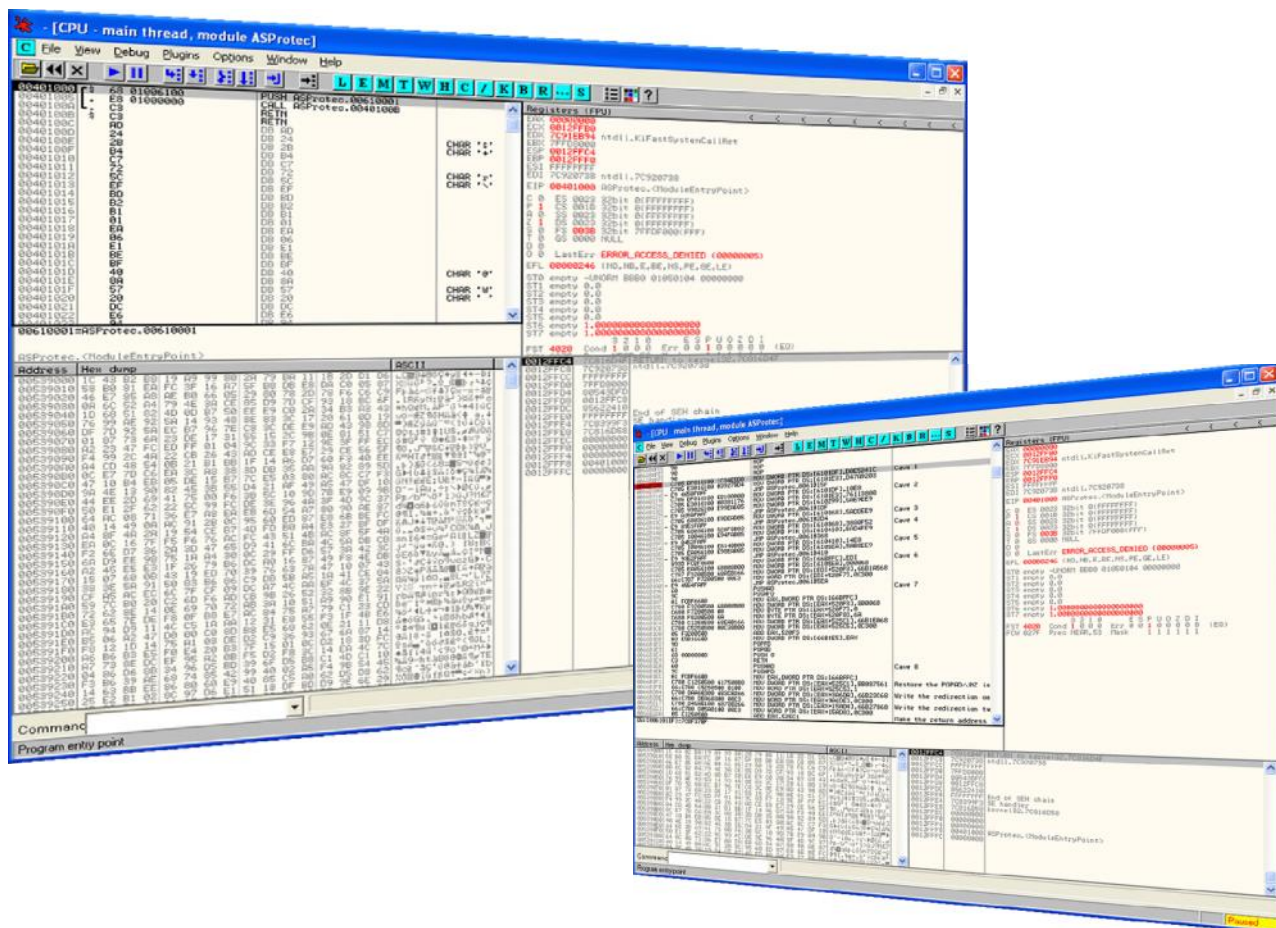




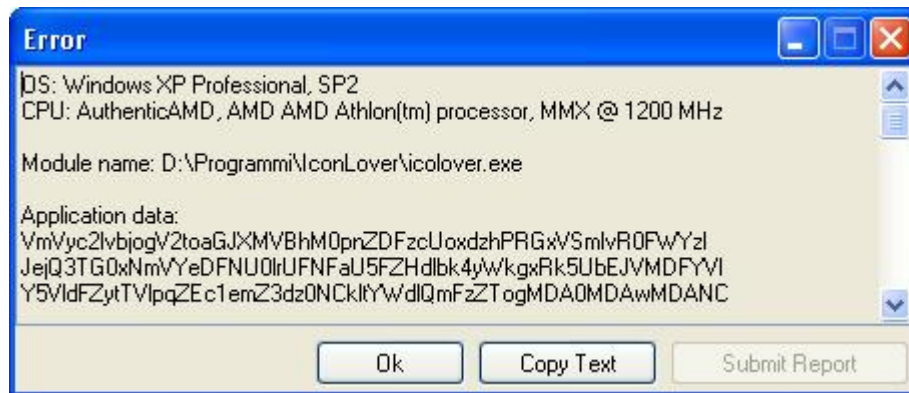
Inline Patching ASProtect 2.x SKE And Defeating The Memory CRC

ThunderPwr of ARTeam



Ver.	Date	Description
1.0	22/08/2006	First public release.

What I've to show you now will be about the ASProtect memory checking feature, this is responsible for this error nag:



This arise mainly by two reason:

1. when some patch into the ASProtect code has been detected
2. when some patch into the decompressed target code has been detected

Our goal now is search a way to defeat this boring nag and check. To do that first we've to reach the ASProtect dll, as I've previously show in other tutorial we can use the **VirtualFree** API breakpoint methods (3 break and you're right) see [2].

To proof the concepts I've keep in touch with a real target but remember I'll don't teach you how crack this one, but I'm using it simply because it was protected by ASProtect:

- Target name: IconLover
- Target release: 4.11 (09-August-2006)
- Target URL: <http://www.aha-soft.com>
- PEiD signature: ASProtect 2.1x SKE -> Alexey Solodovnikov

About this we've to reach the ASProtect dll, then as usual you've:

012450C4	55	PUSH EBP	;
icolover.007B23FA			
012450C5	8BEC	MOV EBP,ESP	
012450C7	83C4 B4	ADD ESP,-4C	
012450CA	B8 A44E2401	MOV EAX,1244EA4	
012450CF	E8 3806FEFF	CALL 0122570C	
012450D4	E8 C3E4FDFF	CALL 0122359C	
012450D9	8D40 00	LEA EAX,DWORD PTR DS:[EAX]	
012450DC	0000	ADD BYTE PTR DS:[EAX],AL	

Now, for our purpose, we can defeat the ASProtect trial time protection, to do this I've found this check just before the **.key** point, then simply perform a search for all referenced strings and search for the **.key** text:

012398D4	33C9	XOR ECX,ECX	
012398D6	BA 1C9A2301	MOV EDX,1239A1C	; ASCII ".key"
012398DB	B8 00000080	MOV EAX,80000000	
012398E0	E8 CB09FFFF	CALL 0122A2B0	
012398E5	68 2C9A2301	PUSH 1239A2C	; ASCII "regfile"
012398EA	33C9	XOR ECX,ECX	
012398EC	BA 1C9A2301	MOV EDX,1239A1C	; ASCII ".key"
012398F1	B8 00000080	MOV EAX,80000000	
012398F6	E8 E509FFFF	CALL 0122A2E0	
012398FB	8D45 D6	LEA EAX,DWORD PTR SS:[EBP-2A]	

Now to deal about trial time limit scroll up a little until you can see the code below:

```

012397C6  MOV EAX,DWORD PTR DS:[12467A4]
012397CB  CALL 0122C344                ; Trial time calculation
012397D0  MOV EAX,DWORD PTR DS:[12467A8]
012397D5  MOV BYTE PTR DS:[EAX],0E7
012397D8  CMP DWORD PTR DS:[ESI+14],0
012397DC  JE SHORT 01239806
012397DE  MOV EAX,DWORD PTR DS:[1246750]
012397E3  CMP BYTE PTR DS:[EAX+31],0
012397E7  JE SHORT 012397ED
012397E9  XOR EAX,EAX                  ; Trial expired -> set residual days to 0
012397EB  JMP SHORT 012397F6
012397ED  MOV EAX,DWORD PTR DS:[1246750]
012397F2  MOVZX EAX,WORD PTR DS:[EAX+C] ; Residual trial time days
012397F6  PUSH EAX
012397F7  MOV EAX,DWORD PTR DS:[1246750]
012397FC  MOVZX EAX,WORD PTR DS:[EAX+8] ; Full trial time days
01239800  PUSH EAX
01239801  MOV EAX,DWORD PTR DS:[ESI+14]
01239804  CALL EAX                     ; Set the trial time to address
used by the target
01239806  MOV EAX,DWORD PTR DS:[12467A8]
0123980B  MOV BYTE PTR DS:[EAX],0E6

```

Step into the CALL 0122C344 and scroll down until the last call then step in:

```

0122C3AA  JMP SHORT 0122C3BB
0122C3AC  MOV ECX,ESI
0122C3AE  LEA EDX,DWORD PTR SS:[ESP+20]
0122C3B2  MOV EAX,ESP
0122C3B4  CALL 0122C140                ; <-- Enter in this call
0122C3B9  MOV BL,1
0122C3BB  MOV EAX,EBX
0122C3BD  ADD ESP,40
0122C3C0  POP EDI
0122C3C1  POP ESI
0122C3C2  POP EBX
0122C3C3  RETN

```

Then scroll down into the call code until you look this sequence:

```

0122C1DE  CALL 012227A8
0122C1E3  PUSH EDX
0122C1E4  PUSH EAX
0122C1E5  FLD QWORD PTR SS:[ESP+30]
0122C1E9  CALL 012227A8
0122C1EE  SUB DWORD PTR SS:[ESP],EAX    ; <-- Force [ESP]=EAX
0122C1F1  SBB DWORD PTR SS:[ESP+4],EDX
0122C1F5  POP EAX
0122C1F6  POP EDX
0122C1F7  MOV ESI,EAX
0122C1F9  INC ESI
0122C1FA  MOV WORD PTR DS:[EBX+A],SI
0122C1FE  MOV AX,WORD PTR SS:[ESP+8]
0122C203  CMP SI,AX
0122C206  JBE SHORT 0122C210
0122C208  MOV WORD PTR DS:[EBX+C],0
0122C20E  JMP SHORT 0122C218
0122C210  SUB AX,SI
0122C213  INC EAX

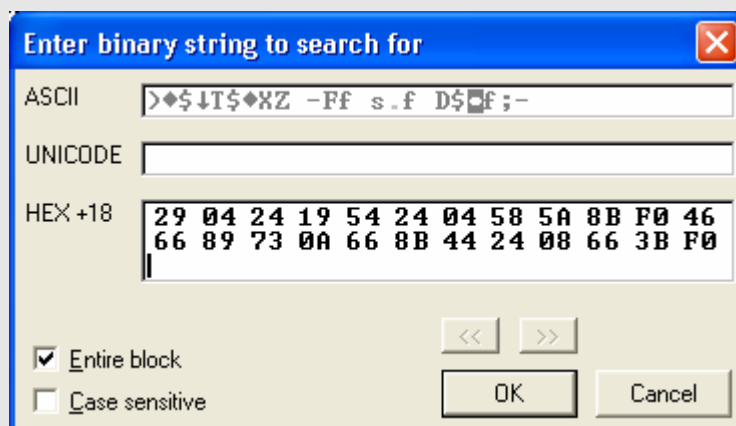
```

Ok. Now we got the point, the SUB instruction put in [ESP] the days elapsed from the trial start, to defeat the trial limit we've first to force in [ESP] the value which is into the EAX registers.

Trick

To reach this point in easy way from the ASProtect DLL entry point perform a binary search for this sequence (Ctrl+b):

29 04 24 19 54 24 04 58 5A 8B F0 46 66 89 73 0A 66 8B 44 24 08 66 3B F0



After that place a software breakpoint (F2) into the previous SUB instruction and also scroll down until the RETN instruction, we've to set another patch:

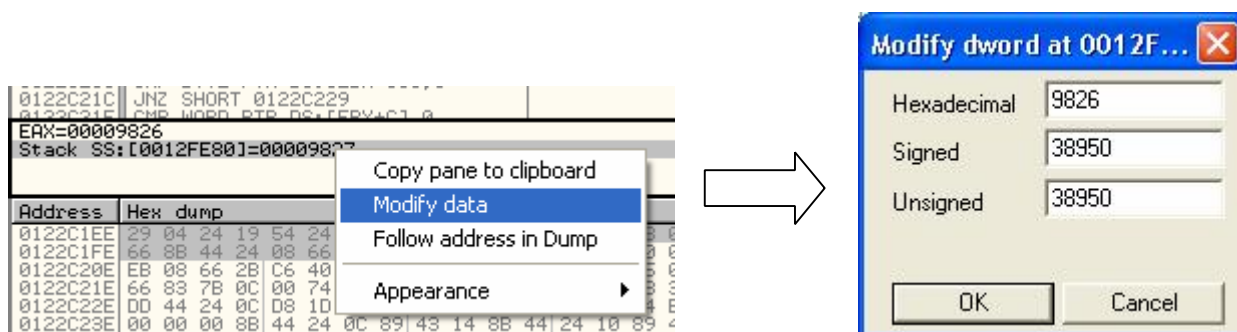
```

0122C326 CALL 0122C0C8
0122C32B TEST AL,AL
0122C32D JNZ SHORT 0122C333
0122C32F XOR EAX,EAX
0122C331 JMP SHORT 0122C335
0122C333 MOV AL,1 ; <-- Change in MOV AL,0
0122C335 MOV BYTE PTR DS:[EBX+31],AL
0122C338 ADD ESP,40
0122C33B POP EDI
0122C33C POP ESI
0122C33D POP EBX
0122C33E RETN
0122C33F ADD BYTE PTR DS:[EAX],AL
0122C341 ADD BYTE PTR DS:[EAX],AL

```

Apply the patch and put another software breakpoint.

Now press Shift+F9 to run the target, OllyDbg will break into the first breakpoint (the SUB instruction) change the stack value and put it equal to EAX one:



Now press again Shift+F9 and reach the last breakpoint.

0122C32D	75 04	JNZ SHORT 0122C333	
0122C32F	33C0	XOR EAX,EAX	
0122C331	EB 02	JMP SHORT 0122C335	
0122C333	00 00	MOV AL,0	<-- Change in MOV AL,0
0122C335	8B43 31	MOV BYTE PTR DS:[EBX+31],AL	
0122C338	83C4 40	ADD ESP,40	
0122C33B	5F	POP EDI	
0122C33C	5E	POP ESI	

Now think about the integrity check, to do this control ASProtect must scan all the code then also the our patching above, then simply put a memory breakpoint on access to the address where the byte 0 of the patch was written, then right click -> Follow in Dump -> Selection:

Address	Hex dump	ASCII
0122C333	B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C343	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C353	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C363	96 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C373	75 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C383	54 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C393	24 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C3A3	01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C3B3	C4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C3C3	C3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C3D3	F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C3E3	B1 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C3F3	C3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C403	64 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C413	63 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C423	79 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C433	79 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C443	59 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C453	EB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000
0122C463	33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0000000000000000

After that simply press Shift+F9, after a little OllyDbg will break, press CTRL+F9:

0122986F	8A5C0A FF	MOV BL, BYTE PTR DS:[EDX+ECX-1]	<<-- Memory breakpoint when reading [0122C333]
01229873	C1E3 18	SHL EBX, 18	
01229876	31D8	XOR EAX, EBX	
01229878	B3 08	MOV BL, 8	
0122987A	D1E0	SHL EAX, 1	
0122987C	73 05	JNB SHORT 01229883	
0122987E	35 410671DB	XOR EAX, DB710641	
01229883	FECB	DEC BL	
01229885	75 F3	JNZ SHORT 0122987A	
01229887	E2 E6	LOOPD SHORT 0122986F	
01229889	F7D0	NOT EAX	
0122988B	8945 F0	MOV DWORD PTR SS:[EBP-10], EAX	
0122988E	61	POPAD	
0122988F	8B03	MOV EAX, DWORD PTR DS:[EBX]	
01229891	5B	POP EBX	
01229892	8BE5	MOV ESP, EBP	
01229894	5D	POP EBP	
01229895	C3	RETN	
01229896	8BC0	MOV EAX, EAX	
01229898	55	PUSH EBP	
01229899	8BEC	MOV EBP, ESP	

Press F8 once you are into the key point now:

01229925	8945 FC	MOV DWORD PTR SS:[EBP-4], EAX	<<-- You are here
01229928	EB 10	JMP SHORT 0122993A	
0122992A	83C3 02	ADD EBX, 2	
0122992D	4E	DEC ESI	
0122992E	75 9D	JNZ SHORT 012298CD	Loop2
01229930	8B0B	MOV ECX, DWORD PTR DS:[EBX]	
01229932	85C9	TEST ECX, ECX	
01229934	0F85 7AFFFFFF	JNZ 012298B4	Loop1
0122993A	8B45 FC	MOV EAX, DWORD PTR SS:[EBP-4]	<<-- Place a hardware breakpoint on execution
0122993D	5F	POP EDI	
0122993E	5E	POP ESI	
0122993F	5B	POP EBX	
01229940	59	POP ECX	
01229941	59	POP ECX	
01229942	5D	POP EBP	
01229943	C2 1000	RETN 10	

Place a hardware breakpoint on execution into the MOV EAX, DWORD PTR SS:[EBP-4] instruction, this will retain the memory CRC.

Pressing Shift+F9 and run the target will be catch the memory error nag because actually we don't know what is the right CRC value, in fact we've altered the code then the check will fail.

Ok, restart the target and disable all the software breakpoint, press Shift+F9 you should land into the ASProtect dll entry point.

Hint

This target don't have the hardware breakpoint clearing feature (check may be done with my tools UASPRDbg [2] and nothing will be found), but from a general point of view you've to sure about that, if clearing feature was set patch the routine in order to use freely your hardware breakpoints.

Now press again Shift+F9 to reach the first CRC value:

Address	Disassembly	Registers (FPU)
0122993A	8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]	EAX 18CFD6E2
0122993D	5F POP EDI	ECX 32FE5D1F
0122993E	5E POP ESI	EDX 0000018C
0122993F	5B POP EBX	EBX 01248B8C
01229940	59 POP ECX	ESP 0012FF50
01229941	59 POP ECX	EBP 0012FF64
01229942	5D POP EBP	ESI 0000008A
01229943	C2 1000 RETN 10	EDI 0122C3DE
01229946	8BC0 MOV EAX,EAX	EIP 0122993A
01229948	55 PUSH EBP	
01229949	8BEC MOV EBP,ESP	

Address	Disassembly
0012FF50	01220000
0012FF54	010E0000
0012FF58	01238D88
0012FF5C	0122C000
0012FF60	18CFD6E2
0012FF64	0012FF98
0012FF68	0122C3E7 RETURN to 0122C3E7 from 01229898
0012FF6C	01220000
0012FF70	0002B000
0012FF74	0000A6F0
0012FF78	00001CD4
0012FF7C	18CE4CDA
0012FF80	0012FFE0 Pointer to next SEH record
0012FF84	01238DEA SE handler
0012FF88	0012FF98

Take a note for the EAX value, we've **18CFD6E2 (FIRST CHECK)**.

Press again Shift+F9, what's up... we've another break this means a double check:

Address	Disassembly	Registers (FPU)
0122993A	8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]	EAX 1E0C1C5C
0122993D	5F POP EDI	ECX 40CC893E
0122993E	5E POP ESI	EDX 0000003C
0122993F	5B POP EBX	EBX 0124C5BC ASCII "N:w:Y>"
01229940	59 POP ECX	ESP 0012FF50
01229941	59 POP ECX	EBP 0012FF64
01229942	5D POP EBP	ESI 00000004
01229943	C2 1000 RETN 10	EDI 01239A4E
01229946	8BC0 MOV EAX,EAX	EIP 0122993A
01229948	55 PUSH EBP	
01229949	8BEC MOV EBP,ESP	

Address	Disassembly
0012FF50	01220000
0012FF54	010E0000
0012FF58	01238D88
0012FF5C	01239000
0012FF60	1E0C1C5C
0012FF64	0012FF98
0012FF68	01239A60 RETURN to 01239A60 from 01229898
0012FF6C	01220000
0012FF70	0002B000
0012FF74	00017954
0012FF78	000020E0
0012FF7C	1E0D94C0
0012FF80	0012FFE0 Pointer to next SEH record
0012FF84	01238DEA SE handler
0012FF88	0012FF98

Take a note for the EAX value, we've **1E0C1C5C (SECOND CHECK)**.

Well, we've near to finish now, at this point we know that we've two CRC check to fix.

Restart the target and reach the ASProtect dll entry point, then place a software breakpoint (F2) into the SUB instruction (trial time checking, look for it trough the binary search tip) and place also the MOV AL,0 patch.

Press Shift+F9 and when you're into the SUB instruction force [ESP] = EAX, press Shift+F9 and you're into the first CRC checking point:

0122993A	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]			
0122993D	5F	POP EDI			
0122993E	5E	POP ESI			
0122993F	5B	POP EBX			
01229940	59	POP ECX			
01229941	59	POP ECX			
01229942	5D	POP EBP			
01229943	C2 1000	RETN 10			
01229946	8BC0	MOV EAX,EAX			
01229948	55	PUSH EBP			
01229949	8BEC	MOV EBP,ESP			

Registers (FPU)	
EAX	7C32AF94
ECX	E946FF02
EDX	0000018C
EBX	01248B8C
ESP	0012FF50
EBP	0012FF64
ESI	0000008A
EDI	0122C3DE
EIP	0122993A

0012FF50	01220000	
0012FF54	010E0000	
0012FF58	01238D88	
0012FF5C	0122C000	
0012FF60	7C32AF94	
0012FF64	0012FF98	
0012FF68	0122C3E7	RETURN to 0122C3E7 from 01229898
0012FF6C	01220000	
0012FF70	0002B000	
0012FF74	0000A6F0	
0012FF78	00001CD4	
0012FF7C	18CE4CDA	
0012FF80	0012FFE0	Pointer to next SEH record
0012FF84	01238DEA	SE handler

Compare with the previous analysis, EAX is different then force the right value in [EBP-4]:

0012FF50	01220000	
0012FF54	010E0000	
0012FF58	01238D88	
0012FF5C	0122C000	
0012FF60	18CE4CDA	
0012FF64	0012FF98	
0012FF68	0122C3E7	RETURN to 0122C3E7 from 01229898
0012FF6C	01220000	
0012FF70	0002B000	
0012FF74	0000A6F0	
0012FF78	00001CD4	
0012FF7C	18CE4CDA	
0012FF80	0012FFE0	Pointer to next SEH record
0012FF84	01238DEA	SE handler

Press Shift+F9, we've another break but this time the value was equal to the previous one. Referring the inline patching way we've to put a redirection from then MOV EAX,DWORD PTR SS:[EBP-4] instruction to the our cave, then write back to [EBP-4 and EAX the right CRC value, also we've to set up a counter and when count was equal to 1 write the first check value and when was two the last one.

Now if you press again Shift+F9 it run, trial was fixed and no nag about CRC arise.



Work done!

REFERENCE

- [1] ThunderPwr of ARTeam, "Inline patching ASProtect 2.3 SKE", 11 August 2006
- [2] ThunderPwr of ARTeam, "ASProtect analysis of the Hardware Breakpoint clearing feature", 20 August 2006

GREETZ



ThunderPwr

**I would just say thanks to all ARTeam friends,
and to Ricardo Narvaja / Cracks Latinos.
Of course thanks to you that have keep on reading this tutorial.**