

## An approach to Asprotect Patching

MaDMan\_H3rCuL3s of ARTeam

Version 1.0 – May 2006

1.	Abstract.....	2
2.	New Approach on Asprotect Patching.....	3
3.	ATTACK ON ASPROTECT SKE REVISITED.....	30
4.	References.....	39
5.	Conclusions.....	39
6.	History.....	39
7.	Greetings.....	39

### Keywords

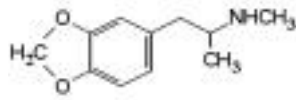
Patch, Asprotect, SKE

Targets used:

[DVD2One v2.0.5](#)

[Asprotect SKE v2.3 Build 426](#)





## Synopsis on Asprotect Patching by MaDMan\_H3rCuL3s

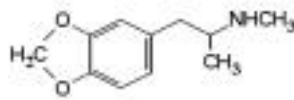
---

### 1. Abstract

In the now infamous tutorial by JohnWho he discussed Asprotect patching by means of decryptor blocks, and doing what I call a daisy chain through each block. In this tutorial we will discuss a new approach, with the same basic idea as his tutorial, but I will eliminate the need for the blocks. This tutorial is going to be very long and hopefully in-depth. So please, only start reading this if you have a few hours to kill. Or if you have a whole day to kill, so you can master the technique. Either way you should walk away from this tutorial with a lot more know how then you did previously.

The techniques described here are general and not specific to any commercial applications. Although I use specific targets, I am trying to teach you the way, and not the crack. The whole document must be intended as a document on reversing advanced techniques, how you will use these information will be totally up to your responsibility.





## 2. New Approach on Asprotect Patching.

We will begin at the EP. Today's target is DVD2One and many more when I get more time to do it. Unfortunately DVD2One was Asprotect SKE 2.3 last I checked, today (v2.0.5) it is back to v2.1 SKE (so CRC is very simple) but chapter 2 of this tutorial will discuss how I patch Asprotect SKE v2.2 – 2.3 (and beyond). To make sure we have a good understanding, I will cover everything here. Please note that a bunch of things have already been covered by JohnWho and I DO NOT take credit for this, I am just applying my technique on top of his already completed steps. SO like usual we begin at the EP:

Address	Hex dump	Disassembly
00401000	68 01A08700	PUSH dvd2one2.0087A001
00401005	E8 01000000	CALL dvd2one2.0040100B
0040100A	C3	RETN
0040100B	C3	RETN
0040100C	BD	DB BD
0040100D	83	DB 83
0040100E	0F	DB 0F
0040100F	2E	DB 2E
00401010	81	DB 81
00401011	B4	DB B4
00401012	6D	DB 6D

At the EP

Now we trace with F7 to get to first decryptor block, until we reach here:

0087A0FD	81CF 50F70540	OR EDI,4003F750
0087A103	8B33	MOV ESI,DWORD PTR DS:[EBX]
0087A105	66:81DF FF50	SBB DI,50FF
0087A10A	81EE 552B683F	SUB ESI,3F682B55
0087A110	81C6 6A894C55	ADD ESI,554C896A
0087A116	68 64BBAE08	PUSH 8AEBB64
0087A11B	0FBFF8	MOVSX EDI,AX
0087A11E	5A	POP EDX
0087A11F	81C6 5BFE7055	ADD ESI,5570FE5B
0087A125	E8 0D000000	CALL dvd2one2.0087A137
0087A12A	01A6 E7943D32	ADD DWORD PTR DS:[ESI+323D94E7],ESP
0087A130	8300 39	ADD DWORD PTR DS:[EAX],39
0087A133	7E DF	JLE SHORT dvd2one2.0087A114
0087A135	2C F5	SUB AL,0F5
0087A137	E9 0D000000	JMP dvd2one2.0087A149
0087A13C	71 56	JNO SHORT dvd2one2.0087A194
0087A13E	D7	XLAT BYTE PTR DS:[EBX+AL]
0087A13F	C4AD E27330A9	LES EBP,FWORD PTR SS:[EBP+A93073E2]
0087A145	2E:CF	IRETD
0087A147	5C	POP ESP
0087A148	65:58	POP EAX
0087A14A	56	PUSH ESI
0087A14B	66:B8 C7F3	MOV AX,0F3C7
0087A14F	8F03	POP DWORD PTR DS:[EBX]
0087A151	0FB7D2	MOVZX EDX,DX
0087A154	83EB 04	SUB EBX,4
0087A157	66:BF EA96	MOV DI,96EA
0087A15B	83E9 01	SUB ECX,1
0087A15E	0F85 9FFFFFFF	JNZ dvd2one2.0087A103
0087A164	8BFA	MOV FPU,FRX

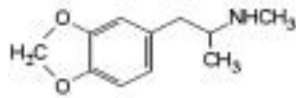
We see the JNZ, which loops until we are satisfied.

We are only interested in these:

0087A10A	81EE 552B683F	SUB ESI,3F682B55
0087A110	81C6 6A894C55	ADD ESI,554C896A
0087A11F	81C6 5BFE7055	ADD ESI,5570FE5B

So this is the first step of decryption. The code we got passes this, then goes on to the next block. There are 5 blocks in all (usually you will see between 4 or 5). Mostly 5 as this is standard now in v2.2 and above. What we will do is open up notepad and copy all this code to it.





## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s

And then at the end we will reverse it all to encrypt our code. So let's continue until we reach block #2. Set a BP below the JNZ and then break, then continue tracing with F7 until we reach the next block, which is here:

Address	Hex dump	Disassembly
0087A1AE	FF30	PUSH DWORD PTR DS:[EAX]
0087A1B0	5A	POP EDX
0087A1B1	66:81E7 5560	AND DI,6055
0087A1B6	81EA 25DA655F	SUB EDX,5F65DA25
0087A1BC	8ADD	MOV BL,CH
0087A1BE	81C2 FA577344	ADD EDX,447357FA
0087A1C4	81F2 ABD1464F	XOR EDX,4F46D1AB
0087A1CA	BF 2807526E	MOV EDI,6E520728
0087A1CF	52	PUSH EDX
0087A1D0	✓ E9 09000000	JMP dvd2one2.0087A1DE
0087A1D5	✓ 7D 72	JGE SHORT dvd2one2.0087A249
0087A1D7	C3	RET
0087A1D8	40	INC EAX
0087A1D9	^ 79 BE	JNS SHORT dvd2one2.0087A199
0087A1DB	1F	POP DS
0087A1DC	6C	INS BYTE PTR ES:[EDI],DX
0087A1DD	35 8F00FB7	XOR EAX,B70F008F
0087A1E2	CA 83E8	RET 0E883
0087A1E5	010F	ADD DWORD PTR DS:[EDI],ECX
0087A1E7	BF FE484848	MOV EDI,484848FE
0087A1EC	66:8BD8	MOV BX,AX
0087A1EF	83EE 01	SUB ESI,1
0087A1F2	✓ 0F85 0F000000	JNZ dvd2one2.0087A207
0087A1F8	8AD8	MOV BL,AL
0087A1FA	✓ E9 23000000	JMP dvd2one2.0087A222
0087A1FF	A0 591EFFCC	MOV AL,BYTE PTR DS:[CCFF1E59]
0087A204	15 2A1B5666	ADC EAX,66561B2A
0087A209	81C7 EFA45FE9	ADD EDI,E95FA4EF
0087A20F	9B	WAIT
0087A210	FFFF	???

I know it looks ugly but the code is obfuscated below a bit.

We are only interested in the following:

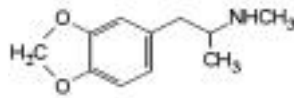
0087A1B6	81EA 25DA655F	SUB EDX,5F65DA25
0087A1BE	81C2 FA577344	ADD EDX,447357FA
0087A1C4	81F2 ABD1464F	XOR EDX,4F46D1AB

Proceeding...

After copying this stuff over, lets move on, set a BP on this JMP:

0087A1F2	/0F85 0F000000	JNZ dvd2one2.0087A207
0087A1F8	8AD8	MOV BL,AL
0087A1FA	E9 23000000	JMP dvd2one2.0087A222

Now break on the JMP and continue with F7 until next block here:



## Synopsis on Asprotect Patching by MaDMan\_H3rCuL3s

Address	Hex dump	Disassembly
0087A24E	FF343B	PUSH DWORD PTR DS:[EBX+EDI]
0087A251	50	PUSH EAX
0087A252	80D1 72	ADC CL,72
0087A255	5A	POP EDX
0087A256	58	POP EAX
0087A257	81F0 8025794B	XOR EAX,4B792580
0087A25D	68 B19A8048	PUSH 48809AB1
0087A262	0F83 05000000	JNB dvd2one2.0087A26D
0087A268	BA 70CBFC0C	MOV EDX,0CFCCB70
0087A26D	5E	POP ESI
0087A26E	81C0 B927020D	ADD EAX,0D0227B9
0087A274	68 A5908E00	PUSH 8E90A5
0087A279	0F88 12000000	JS dvd2one2.0087A291
0087A27F	E8 0C000000	CALL dvd2one2.0087A290
0087A284	34 5D	XOR AL,5D
0087A286	D2A3 A0591EFF	SHL BYTE PTR DS:[EBX+FF1E59A0],CL
0087A28C	CC	INT3
0087A28D	15 2A1B5959	ADC EAX,59591B2A
0087A292	81C0 FE771B64	ADD EAX,641B77FE
0087A298	81D9 D0C1920D	SBB ECX,0D92C1D0
0087A29E	50	PUSH EAX
0087A29F	68 FC2A2A76	PUSH 762A2AFC
0087A2A4	53	PUSH EBX
0087A2A5	8AE8	MOV CH,AL
0087A2A7	59	POP ECX
0087A2A8	5A	POP EDX
0087A2A9	8F041F	POP DWORD PTR DS:[EDI+EBX]
0087A2AC	80C1 2C	ADD CL,2C
0087A2AF	0FBFD0	MOVSX EDX,AX
0087A2B2	83EB 04	SUB EBX,4
0087A2B5	B1 D1	MOV CL,0D1
0087A2B7	81FB 9CF9FFFF	CMP EBX,-664
0087A2BD	0F85 18000000	JNZ dvd2one2.0087A2DB
0087A2C3	0F87 00000000	JA dvd2one2.0087A2C9
0087A2C9	E9 1E000000	JMP dvd2one2.0087A2EC
0087A2CE	E1 06	LOOPDE SHORT dvd2one2.0087A2D6
0087A2D0	C7	???
0087A2D1	F4	HLT
0087A2D2	1D 92636019	SBB EAX,19606392
0087A2D7	DEBF 8CD5E96E	FIDIVR WORD PTR DS:[EDI+6EE9D58C]
0087A2DD	FFFF	???
0087A2DF	FF248D 42539081	JMP DWORD PTR DS:[ECX*4+89905342]

Again the code is obfuscated a bit.

### We only care about this:

0087A257	81F0 8025794B	XOR EAX,4B792580
0087A26E	81C0 B927020D	ADD EAX,0D0227B9
0087A292	81C0 FE771B64	ADD EAX,641B77FE

Let's proceed.....

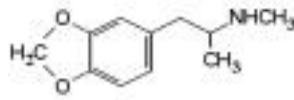
### Set bp on this JMP:

0087A2BD	/0F85 18000000	JNZ dvd2one2.0087A2DB
0087A2C3	0F87 00000000	JA dvd2one2.0087A2C9
0087A2C9	E9 1E000000	JMP dvd2one2.0087A2EC

Now break on the JMP and continue with F7 until we reach the next block here:







## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s

Address	Hex dump	Disassembly
0087A339	8B39	MOV EDI,DWORD PTR DS:[ECX]
0087A33B	66:81E3 E968	AND BX,68E9
0087A340	81F7 5F357564	XOR EDI,6475355F
0087A346	8BDE	MOV EBX,ESI
0087A348	81F7 ACDBA462	XOR EDI,62A4DBAC
0087A34E	B7 A8	MOV BH,0A8
0087A350	81EF 7520284E	SUB EDI,4E282075
0087A356	8BD0	MOV EDX,EAX
0087A358	8939	MOV DWORD PTR DS:[ECX],EDI
0087A35A	83E9 04	SUB ECX,4
0087A35D	66:BA 82C4	MOV DX,0C482
0087A361	81E8 01000000	SUB EAX,1
0087A367	^ 0F85 CFFFFFFF	JNZ dvd2one2.0087A339
0087A36D	66:81F3 855F	XOR BX,5F85
0087A372	05 1C6E8A6C	ADD EAX,6C8A6E1C

This time the code looks fine.

We are only interested in the following:

0087A340	81F7 5F357564	XOR EDI,6475355F
0087A348	81F7 ACDBA462	XOR EDI,62A4DBAC
0087A350	81EF 7520284E	SUB EDI,4E282075

To continue.....

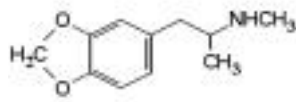
Set a BP on this:

0087A36D	66:81F3 855F	XOR BX,5F85
----------	--------------	-------------

And then break on it; now continue with F7 until the last block:

0087A397	FF343B	PUSH DWORD PTR DS:[EBX+EDI]
0087A39A	68 4E1E9949	PUSH 49991E4E
0087A39F	B9 8BBFB850	MOV ECX,50B8BF8B
0087A3A4	5A	POP EDX
0087A3A5	5E	POP ESI
0087A3A6	80F1 B2	XOR CL,0B2
0087A3A9	81C6 F242893D	ADD ESI,3D8942F2
0087A3AF	0FBFD1	MOVSX EDX,CX
0087A3B2	81EE 436D033F	SUB ESI,3F036D43
0087A3B8	0FB7D3	MOVSX EDX,BX
0087A3BB	81EE C01F9264	SUB ESI,64921FC0
0087A3C1	81C8 626B3875	OR EAX,75386B62
0087A3C7	56	PUSH ESI
0087A3C8	^ E9 13000000	JMP dvd2one2.0087A3E0
0087A3CD	4F	DEC EDI
0087A3CE	DCE5	FSUBR ST(5),ST
0087A3D0	BA 6BC86186	MOV EDX,8661C86B
0087A3D5	47	INC EDI
0087A3D6	^ 74 9D	JE SHORT dvd2one2.0087A375
0087A3D8	12E3	ADC AH,BL
0087A3DA	^ E0 99	LOOPDNE SHORT dvd2one2.0087A375
0087A3DC	5E	POP ESI
0087A3DD	3F	AAS
0087A3DE	0C 55	OR AL,55
0087A3E0	8F041F	POP DWORD PTR DS:[EDI+EBX]
0087A3E3	8BD0	MOV EDX,EAX
0087A3E5	66:8BD1	MOV DX,CX
0087A3E8	83EB 04	SUB EBX,4
0087A3EB	0FBFC9	MOVSX ECX,CX
0087A3EE	81FB D8FAFFFF	CMP EBX,-528
0087A3F4	^ 0F85 1D000000	JNZ dvd2one2.0087A417
0087A3FA	81C0 D43BD066	ADD EAX,66BD3BD4
0087A400	^ E9 21000000	JMP dvd2one2.0087A426
0087A405	^ 72 C3	JB SHORT dvd2one2.0087A3CA
0087A407	40	INC EAX
0087A408	^ 79 BE	JNS SHORT dvd2one2.0087A3C8
0087A40A	1F	POP DS
0087A40B	6C	INS BYTE PTR ES:[EDI],DX
0087A40C	35 CA3B58B1	XOR EAX,B1583BCA
0087A411	96	XCHG EAX,ESI
0087A412	17	POP SS
0087A413	04 ED	ADD AL,0ED
0087A415	22B3 E97BFFFF	AND DH,BYTE PTR DS:[EBX+FFFF7BE9]





## Synopsis on Asprotect Patching by MaDMan\_H3rCuL3s

Figures the last block is ugly again ☺

### All we care about is this:

0087A3A9	81C6 F242893D	ADD ESI, 3D8942F2
0087A3B2	81EE 436D033F	SUB ESI, 3F036D43
0087A3BB	81EE C01F9264	SUB ESI, 64921FC0

And now we have all the info we need to actually see how to encrypt our code so it is decrypted at runtime and patches the program for us. If we look at notepad we should have the following in notepad:

### NOTEPAD:

```
Block 1.
81EE 552B683F SUB ESI, 3F682B55
81C6 6A894C55 ADD ESI, 554C896A
81C6 5BFE7055 ADD ESI, 5570FE5B

Block 2.
81EA 25DA655F SUB EDX, 5F65DA25
81C2 FA577344 ADD EDX, 447357FA
81F2 ABD1464F XOR EDX, 4F46D1AB

Block 3.
81F0 8025794B XOR EAX, 4B792580
81C0 B927020D ADD EAX, 0D0227B9
81C0 FE771B64 ADD EAX, 641B77FE

Block 4.
81F7 5F357564 XOR EDI, 6475355F
81F7 ACDBA462 XOR EDI, 62A4DBAC
81EF 7520284E SUB EDI, 4E282075

Block 5.
81C6 F242893D ADD ESI, 3D8942F2
81EE 436D033F SUB ESI, 3F036D43
81EE C01F9264 SUB ESI, 64921FC0
```

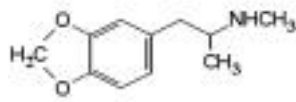
And since most of us know what we are not doing, I will assume you have no idea what to do now, which is the reason for this tutorial. Let's sit for a moment. If something is decrypted one way, to encrypt it, we go the other. Right? Well yes, but it's not as easy as 1-2-3. The way I developed, is to go from each block, and reverse it. So we will start at Block #5 and then go to block #4, and so on.... So what we will need is a bit of time, and patience. Let us first think of what we want to have the code to become. Well I chose to use this:

```
0087A5E3 68 00800000 PUSH 8000
0087A5E8 6A 00      PUSH 0
```

And I will make it hold this:

```
0087A5E3 68 00B08900 PUSH dvd2one2.0089B000
0087A5E8 C3          RETN
```





## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s

And it will go to my injection.

So we start from the last block. But we need to reverse the algo so it encrypts our code. Well if we look at our notepad, we see it's not too hard to do this.

Notepad Reversed:		
Block 5		
81C6 C01F9264	ADD ESI,64921FC0	
81C6 436D033F	ADD ESI,3F036D43	
81EE F242893D	SUB ESI,3D8942F2	
Block 4		
81C6 7520284E	ADD EDI,4E282075	
81F7 ACDBA462	XOR EDI,62A4DBAC	
81F7 5F357564	XOR EDI,6475355F	
Block 3		
81C0 FE771B64	SUB EAX,641B77FE	
81C0 B927020D	SUB EAX,0D0227B9	
81F0 8025794B	XOR EAX,4B792580	
Block 2		
81F2 ABD1464F	XOR EDX,4F46D1AB	
81C2 FA577344	SUB EDX,447357FA	
81EA 25DA655F	ADD EDX,5F65DA25	
Block 1		
81C6 5BFE7055	SUB ESI,5570FE5B	
81C6 6A894C55	SUB ESI,554C896A	
81EE 552B683F	ADD ESI,3F682B55	

So we just fix the code to do this like above.

But we must do one more thing. Please take note that Asprotect does not just do a normal DWORD, I mean in nice intervals. Like 00401000, 00401004, 00401008.. etc.

Sometimes we do:

00401001, 00401005, etc.

00401002, 00401006, etc.

So to make our patch immune from this crazy DWORDing, we will take the code from a few extra bytes before and after the actual patched code.

So in notepad I would copy this over:

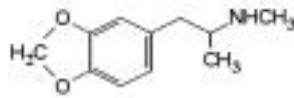
DWORDing	
0087A5DD	8B 5E 08 03 DF 53 68 00 B0 89 00 C3 90 56 FF <^□□ßSh.°%.Ã□Vÿ

This way we won't get screwed in the long run.

DWORDing 2:		
0087A5DD	8B5E 08	MOV EBX,DWORD PTR DS:[ESI+8]
0087A5E0	03DF	ADD EBX,EDI
0087A5E2	53	PUSH EBX
0087A5E3	68 00B08900	PUSH dvd2one2.0089B000
0087A5E8	C3	RETN
0087A5E9	90	NOP
0087A5EA	56	PUSH ESI
0087A5EB	FF95 F4030000	CALL DWORD PTR SS:[EBP+3F4]
0087A5F1	68 00000000	PUSH 0







## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s

Another note is that this NOP you see is not actually part of the patch, just the PUSH, RETN.

So it's really this code we will patch:

DWORDing 3:			
0087A5DD	8B5E 08	MOV EBX,DWORD PTR DS:[ESI+8]	
0087A5E0	03DF	ADD EBX,EDI	; dvd2one2.0040100C
0087A5E2	53	PUSH EBX	; dvd2one2.0087C2DC
0087A5E3	68 00B08900	PUSH dvd2one2.0089B000	
0087A5E8	C3	RETN	
0087A5E9	0056 FF	ADD BYTE PTR DS:[ESI-1],DL	

Hopefully I did what I set out to do... confuse you ☺

Anyways.. The main goal is to patch in our 6 bytes. Now that we understand this, let's get on to Block #5 and place out code in the way we want it.

Address	Hex dump	Disassembly
0087A5DD	8B5E 08	MOV EBX,DWORD PTR DS:[ESI+8]
0087A5E0	03DF	ADD EBX,EDI
0087A5E2	53	PUSH EBX
0087A5E3	68 00B08900	PUSH dvd2one2.0089B000
0087A5E8	C3	RETN
0087A5E9	0056 FF	ADD BYTE PTR DS:[ESI-1],DL
0087A5EC	A2 5A144A0C	MOV BYTE PTR DS:[C4A145A],AL

So our code is the way we want it

Address	Hex dump	Disassembly	Comment
0087A397	FF343B	PUSH DWORD PTR DS:[EBX+EDI]	BLOCK 5
0087A39A	68 4E1E9949	PUSH 49991E4E	
0087A39F	B9 8BBFBA50	MOV ECX,50BABF8B	
0087A3A4	5A	POP EDX	dvd2one2.00
0087A3A5	5E	POP ESI	dvd2one2.00
0087A3A6	80F1 B2	XOR CL,0B2	
0087A3A9	81C6 C01F9264	ADD ESI,64921FC0	
0087A3AF	0FBFD1	MOVSX EDX,CX	
0087A3B2	81C6 436D033F	ADD ESI,3F036D43	
0087A3B8	0FB7D3	MOVSX EDX,BX	
0087A3BB	81EE F242893D	SUB ESI,3D8942F2	
0087A3C1	81C8 626B3875	OR EAX,75386B62	

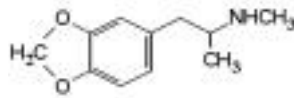
The code was reversed

Address	Hex dump	ASCII
0087A5DD	8B 5E 08 03 DF 53 68 00 B0 89 00 C3 00 56 FF A2	i?Sh.?.f.u.ø
0087A5ED	5A 14 4A 0C CE 11 4A 0C 66 D4 4A 0C 66 11 4A 0C	ZqJ.ifJ.fJ.fJ.
0087A5FD	66 11 9F 97 52 71 9F 97 DB 19 D5 89 72 00 FD 8C	fJfJRqfJfJfJ.

In dump.







## Synopsis on Asprotect Patching by MaDMA\_H3rCuL3s

Address	Hex dump	Disassembly
0087A32F	B8 77010000	MOV EAX,177
0087A334	66:81F3 17CD	XOR BX,0CD17
0087A339	8B39	MOV EDI,DWORD PTR DS:[ECX]
0087A33B	66:81E3 E968	AND BX,68E9
0087A340	81C7 7520284E	ADD EDI,4E282075
0087A346	8BDE	MOV EBX,ESI
0087A348	81F7 ACDBA462	XOR EDI,62A4DBAC
0087A34E	B7 A8	MOV BH,0A8
0087A350	81F7 5F357564	XOR EDI,6475355F
0087A356	8BD0	MOV EDX,EAX
0087A358	8939	MOV DWORD PTR DS:[ECX],EDI
0087A35A	83E9 04	SUB ECX,4
0087A35D	66:BA 82C4	MOV DX,0C482
0087A361	81E8 01000000	SUB EAX,1
0087A367	0F85 CFFFFFFF	JNZ dvd2one2.0087A339
0087A36D	2C 01E3 0EEF	VAD OV EEC

We copy the ADD and XOR's over.

Address	Hex dump														ASCII																															
0087A5BD	3B	F8	DC	E7	B2	75	1B	58	09	E2	1E	2A	B3	75	84	6C	0087A5CD	F0	61	19	00	C7	79	84	E5	01	03	84	4F	B2	03	85	5A													
0087A5DD	F1	6F	52	0F	45	64	B2	0C	16	9A	4A	CF	66	67	49	1B	0087A5ED	AE	7A	84	E5	1A	75	84	E5	B2	BA	85	E5	B2	75	84	E5													
0087A5FD	B2	75	51	6E	A6	15	51	6E	2F	7D	1B	60	C6	71	F3	64	0087A60D	38	7F	5E	6D	B0	3E	82	D7	2C	78	1A	9B	FC	68	D2	79													
0087A61D	A5	78	D2	7B	BF	E3	6F	AB	C0	AB	30	AF	E1	B5	83	D7	0087A62D	78	03	84	E5	01	03	84	4F	B2	03	85	5A	0087A63D	3B	F8	DC	E7	B2	75	1B	58	09	E2	1E	2A	B3	75	84	6C

The DWORD is copied over.

Now we just run it until our code is partially encrypted.

Address	Hex dump																ASCII																																																																																																																																																																																																																																																																																																
0087A5BD	3B	F8	DC	E7	B2	75	1B	58	09	E2	1E	2A	B3	75	84	6C	0087A5CD	F0	61	19	00	C7	79	84	E5	01	03	84	4F	B2	03	85	5A	0087A5DD	F1	6F	52	0F	45	64	B2	0C	16	9A	4A	CF	66	67	49	1B	0087A5ED	AE	7A	84	E5	1A	75	84	E5	B2	BA	85	E5	B2	75	84	E5	0087A5FD	B2	75	51	6E	A6	15	51	6E	2F	7D	1B	60	C6	71	F3	64	0087A60D	38	7F	5E	6D	B0	3E	82	D7	2C	78	1A	9B	FC	68	D2	79	0087A61D	A5	78	D2	7B	BF	E3	6F	AB	C0	03	BF	06	29	D9	4A	2E	0087A62D	7C	1E	D4	12	4C	13	1C	6C	79	D7	48	DC	07	03	1C	72	0087A63D	0B	AB	7E	02	08	D3	5F	9C	F8	D3	C1	C1	81	17	90	6E	0087A64D	C8	2C	0A	DF	C9	8E	4F	46	1C	47	5C	2E	1A	D9	4A	2E	0087A65D	7C	1E	D4	12	4C	13	1C	10	C6	80	50	53	21	F1	D2	C3	0087A66D	62	81	53	03	E1	B9	00	DE	07	19	4A	DE	C9	8E	4F	46	0087A67D	1C	47	5C	2E	1A	D9	4A	2E	7C	1E	D4	12	4C	13	1C	4F	0087A68D	E0	9C	30	DF	78	21	03	DE	07	19	4A	DE	C9	8E	4F	46	0087A69D	1C	47	5C	2E	1A	C2	4A	2E	7C	1E	D4	12	4C	13	1C	4F	0087A6AD	E0	57	D5	F3	33	0E	3E	E1	64	E2	A0	DB	07	18	93	9D	0087A6BD	E6	01	D4	C2	4D	A4	30	B6	07	19	4A	DC	F9	D3	C1	C1	0087A6CD	81	17	90	6E	C8	2C	13	DF	C9	8E	4F	46	1C	47	5C	2E

Almost there ☺

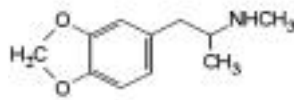
Address	Hex dump	ASCII						
0087A5BD	3B F8 DC E7 B2 75 1B 58 09 E2 1E 2A B3 19 4A 45	0087A5CD 38 25 D7 F9 13 10 4A DC 56 88 4A A6 06 88 4B 53	0087A5DD 39 17 9C E6 95 2A 3C E5 62 FC 85 26 B2 2F 87 92	0087A5ED FA 1C 4A DC 6F 19 4A DC 07 DC 48 DC 07 19 4A DC	0087A5FD 07 19 9F 47 F2 79 9F 47 7B 01 D5 59 12 15 FD 5C	0087A60D 38 07 90 44 00 40 4C 2E 7C 1E D4 12 4C 13 1C 70	0087A61D F5 03 1C 72 0B AB 7E 02 08 D3 BF 06 29 D9 4A 2E	0087A62D 7C 1E D4 12 4C 13 1C 6C 79 D7 48 DC 07 03 1C 72

Now we are finished with block #4. (Highlighted is our encrypted code)

So now our new code is this:







## Synopsis on Asprotect Patching by MaDMAn\_H3rCuL3s

### BLOCK #4 Encrypted Code:

0087A5DD 39 17 9C E6 95 2A 3C E5 62 FC 85 26 B2 2F 87 9□œœ•\*<âbü...&²/†

Binary:

39 17 9C E6 95 2A 3C E5 62 FC 85 26 B2 2F 87

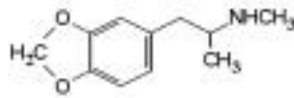
And now we move on to Block #3 using this new code. So we restart it, Trace with F7 until we hit Block #3:

Address	Hex dump	Disassembly
0087A24E	FF343B	PUSH DWORD PTR DS:[EBX+EDI]
0087A251	50	PUSH EAX
0087A252	80D1 72	ADC CL,72
0087A255	5A	POP EDX
0087A256	58	POP EAX
0087A257	81F0 8025794B	XOR EAX,4B792580
0087A25D	68 B19A8048	PUSH 48809AB1
0087A262	0F83 05000000	JNB dvd2one2.0087A26D
0087A268	BA 70CBFC0C	MOV EDX,0CFCCB70
0087A26D	5E	POP ESI
0087A26E	81C0 B927020D	ADD EAX,0D0227B9
0087A274	68 A5908E00	PUSH 8E90A5
0087A279	0F88 12000000	JS dvd2one2.0087A291
0087A27F	E8 0C000000	CALL dvd2one2.0087A290
0087A284	34 5D	XOR AL,5D
0087A286	D2A3 A0591EFF	SHL BYTE PTR DS:[EBX+FF1E59A0],CL
0087A28C	CC	INT3
0087A28D	15 2A1B5959	ADC EAX,59591B2A
0087A292	81C0 FE771B64	ADD EAX,641B77FE
0087A298	81D9 D0C1920D	SBB ECX,0D92C1D0
0087A29E	50	PUSH EAX
0087A29F	68 FC2A2A76	PUSH 762A2AFC
0087A2A4	53	PUSH EBX
0087A2A5	8AE8	MOV CH,AL
0087A2A7	59	POP ECX
0087A2A8	5A	POP EDX
0087A2A9	8F041F	POP DWORD PTR DS:[EDI+EBX]
0087A2AC	80C1 2C	ADD CL,2C
0087A2AF	0FBFD0	MOVSX EDX,AX
0087A2B2	83EB 04	SUB EBX,4
0087A2B5	B1 D1	MOV CL,0D1
0087A2B7	81FB 9CF9FFFF	CMP EBX,-664
0087A2BD	0F85 18000000	JNZ dvd2one2.0087A2DB
0087A2C3	0F87 00000000	JA dvd2one2.0087A2C9
0087A2C9	E9 1E000000	JMP dvd2one2.0087A2EC
0087A2CE	E1 06	LOOPDE SHORT dvd2one2.0087A2D6
0087A2D0	C7	???
0087A2D1	F4	HLT
0087A2D2	1D 92636019	SBB EAX,19606392
0087A2D7	DEBF 8CD5E96E	FIDIVR WORD PTR DS:[EDI+6EE9D58C]
0087A2DD	FFFF	???
0087A2DF	FF248D 4253908	JMP DWORD PTR DS:[ECX*4+89905342]

We are at block #3.

Now we must copy in the modified encryption code, and the DWORD.





## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s

Address	Hex dump	Disassembly
0087A24E	FF343B	PUSH DWORD PTR DS:[EBX+EDI]
0087A251	50	PUSH EAX
0087A252	80D1 72	ADC CL,72
0087A255	5A	POP EDX
0087A256	58	POP EAX
0087A257	2D FE771B64	SUB EAX,641B77FE
0087A25C	90	NOP
0087A25D	68 B19A8048	PUSH 48809AB1
0087A262	0F83 05000000	JNB dvd2one2.0087A26D
0087A268	BA 70CBFC0C	MOV EDX,0CFCCB70
0087A26D	5E	POP ESI
0087A26E	2D B927020D	SUB EAX,000227B9
0087A273	90	NOP
0087A274	68 A5908E00	PUSH 8E90A5
0087A279	0F88 12000000	JS dvd2one2.0087A291
0087A27F	E8 0C000000	CALL dvd2one2.0087A290
0087A284	34 5D	XOR AL,5D
0087A286	D2A3 A0591EFF	SHL BYTE PTR DS:[EBX+FF1E59A0],CL
0087A28C	CC	INT3
0087A28D	15 2A1B5959	ADC EAX,59591B2A
0087A292	35 8025794B	XOR EAX,4B792580
0087A297	90	NOP
0087A298	81D9 D0C1920D	SBB ECX,0D92C1D0
0087A29E	50	PUSH EAX
0087A29F	68 FC2A2A76	PUSH 762A2AFC
0087A2A4	E9	CALL EBX

Modified encryption code.

Now the DWORD.

Address	Hex dump	ASCII
0087A5B0	BE A3 20 B0 36 21 E1 21 4C BD E6 F3 36 21 58 35	"u :6tBtL#p56tX5
0087A5CD	75 3D E3 C9 02 25 58 AE 47 9C 59 18 37 9F 58 23	u=rr0%X<<G6Yt7fx#
0087A5D0	39 17 9C E6 95 2A 3C E5 62 FC 85 26 B2 2F 87 E4	996p6* <ab"588/92
0087A5ED	2B 24 58 AE 5E 2E 58 AE 36 E4 5F AE 36 21 58 AE	+sX<<^,X<<62<<6tX<<
0087A5FD	2B 21 AB 37 23 81 94 37 AA 26 E1 29 03 2D C9 2D	6t%7u07~8B)0-r-
0087A600	BD 18 A6 36 38 58 5A A0 A8 23 E2 64 79 37 2A 42	"t868XZad#fdy7*B
0087A610	20 27 2A 44 3A BF B5 3D 40 A0 C8 41 E1 BA A3 69	"*D:~q=@aAb  u
0087A620	2D 7F 3A 2D FC 60 F2 27 31 B2 A5 77 B3 70 F2 0D	-d:-"z'188w p2,
0087A630	BF D8 0F 3D 40 A0 A8 B7 B3 A0 2E 7A 39 72 7E 29	~t*=@a~ a,z9r")
0087A640	00 61 EC 78 01 0F 98 01 4C 44 B2 69 4F BA A3 69	.a~0*Y0L0  u
0087A650	2D 7F 3A 2D FC 60 F2 2B 72 0C BD EC 59 92 24 7C	-d:-"z+r,"wYEs
0087A660	96 F1 A4 3C 19 C8 ED 79 B3 7A AC 79 01 0F 98 01	u:R<~4py z%Y0*Y0
0087A670	4C 44 B2 69 4F BA A3 69 2D 7F 3A 2D FC 60 F2 08	L0L0  u -d:-"z
0087A680	18 F1 0D 78 3A 5A FD 79 B3 7A AC 79 01 0F 98 01	+z~xAPou z%u0*Y0

And now we set the BP on offset "0087A24E" and run it

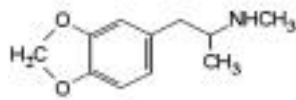
Address	Hex dump	ASCII
0087A5B0	BE A3 20 B0 36 21 E1 21 4C BD E6 F3 36 21 58 35	"u :6tBtL#p56tX5
0087A5CD	75 3D E3 C9 02 25 58 AE 47 9C 59 18 37 9F 58 23	u=rr0%X<<G6Yt7fx#
0087A5D0	39 17 9C E6 95 2A 3C E5 62 FC 85 26 B2 2F 87 E4	996p6* <ab"588/92
0087A5ED	2B 24 58 AE 5E 2E 58 AE 36 E4 5F AE 36 21 58 AE	+sX<<^,X<<62<<6tX<<
0087A5FD	2B 21 AB 37 23 81 94 37 AA 26 E1 29 03 2D C9 2D	6t%7u07~8B)0-r-
0087A600	BD 18 A6 36 38 58 5A A0 A8 23 E2 64 79 37 2A 42	"t868XZad#fdy7*B
0087A610	20 27 2A 44 3A BF B5 3D 40 A0 C8 41 E1 BA A3 69	"*D:~q=@aAb  u
0087A620	2D 7F 3A 2D FC 60 F2 27 31 B2 A5 77 B3 70 F2 0D	-d:-"z'188w p2,
0087A630	BF D8 0F 3D 40 A0 A8 B7 B3 A0 2E 7A 39 72 7E 29	~t*=@a~ a,z9r")
0087A640	00 61 EC 78 01 0F 98 01 4C 44 B2 69 4F BA A3 69	.a~0*Y0L0  u
0087A650	2D 7F 3A 2D FC 60 F2 2B 72 0C BD EC 59 92 24 7C	-d:-"z+r,"wYEs
0087A660	96 F1 A4 3C 19 C8 ED 79 B3 7A AC 79 01 0F 98 01	u:R<~4py z%Y0*Y0
0087A670	4C 44 B2 69 4F BA A3 69 2D 7F 3A 2D FC 60 F2 08	L0L0  u -d:-"z
0087A680	18 F1 0D 78 3A 5A FD 79 B3 7A AC 79 01 0F 98 01	+z~xAPou z%u0*Y0

Almost there ☺

Address	Hex dump	ASCII
0087A5B0	BE A3 20 B0 36 21 E1 21 4C BD E6 F3 36 21 58 35	"u :6tBtL#p56tX5
0087A5CD	75 3D E3 92 46 7E AC 77 82 07 A3 E1 B2 F8 AC EC	u=rrEF"se~uB8%~
0087A5D0	BC 80 61 A5 D3 75 80 AE E6 A7 5F EF 37 68 5D AD	#Can"uC~p0_07k
0087A5ED	A9 7F AC 77 98 69 AC 77 B3 BF A5 77 B3 7A AC 77	r0%wci%w ~w z%w
0087A5FD	B3 7A 71 00 A6 1A 68 00 2F 70 3B F2 46 76 13 F6	zq.3+h./p:zFu  -
0087A600	38 82 7F 0F BD 43 A2 69 2D 7F 3A 2D FC 60 F2 08	8ed "C0i-d:-"z
0087A610	A5 70 F2 0D BF D8 0F 3D 40 A0 C8 41 E1 BA A3 69	~p2.~t*=@aAb  u
0087A620	2D 7F 3A 2D FC 60 F2 27 31 B2 A5 77 B3 70 F2 0D	-d:-"z'188w p2,
0087A630	BF D8 0F 3D 40 A0 A8 B7 B3 A0 2E 7A 39 72 7E 29	~t*=@a~ a,z9r")
0087A640	00 61 EC 78 01 0F 98 01 4C 44 B2 69 4F BA A3 69	.a~0*Y0L0  u
0087A650	2D 7F 3A 2D FC 60 F2 2B 72 0C BD EC 59 92 24 7C	-d:-"z+r,"wYEs
0087A660	96 F1 A4 3C 19 C8 ED 79 B3 7A AC 79 01 0F 98 01	u:R<~4py z%Y0*Y0
0087A670	4C 44 B2 69 4F BA A3 69 2D 7F 3A 2D FC 60 F2 08	L0L0  u -d:-"z
0087A680	18 F1 0D 78 3A 5A FD 79 B3 7A AC 79 01 0F 98 01	+z~xAPou z%u0*Y0

There!





## Synopsis on Asprotect Patching by MaDMAn\_H3rCuL3s

So now we copy the new code over to Notepad and use it for Block #2. You should have this (or similar) as your new code.

Block #3 Encrypted Code:	
0087A5DD BC 80 61 AF D3 75 80 AE E6 A7 5F EF 37 6B 5D	¼Ca`ÓuE@æ\$ i7k]
Binary:	
BC 80 61 AF D3 75 80 AE E6 A7 5F EF 37 6B 5D	

Now we proceed to Block #2, and then on to Block #1, then finally close this chapter, and get ready to patch the program.

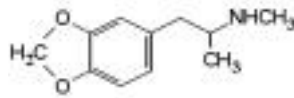
At Block #3.

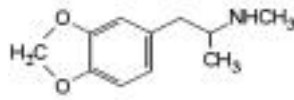
Address	Hex dump	Disassembly
0087A1AE	FF30	PUSH DWORD PTR DS:[EAX]
0087A1B0	5A	POP EDX
0087A1B1	66:81E7 5560	AND DI,6055
0087A1B6	81EA 25DA655F	SUB EDX,5F65DA25
0087A1BC	8ADD	MOV BL,CH
0087A1BE	81C2 FA577344	ADD EDX,447357FA
0087A1C4	81F2 ABD1464F	XOR EDX,4F46D1AB
0087A1CA	BF 2807526E	MOV EDI,6E520728
0087A1CF	52	PUSH EDX
0087A1D0	✓ E9 09000000	JMP dvd2one2.0087A1DE
0087A1D5	✓ 7D 72	JGE SHORT dvd2one2.0087A249
0087A1D7	C3	RET
0087A1D8	40	INC EAX
0087A1D9	^ 79 BE	JNS SHORT dvd2one2.0087A199
0087A1DB	1F	POP DS
0087A1DC	6C	INS BYTE PTR ES:[EDI],DX
0087A1DD	35 8F000FB7	XOR EAX,B70F008F
0087A1E2	CA 83E8	RETN 0E883
0087A1E5	010F	ADD DWORD PTR DS:[EDI],ECX
0087A1E7	BF FE484848	MOV EDI,484848FE
0087A1EC	66:8BD8	MOV BX,AX
0087A1EF	83EE 01	SUB ESI,1
0087A1F2	✓ 0F85 0F000000	.INZ dvd2one2.0087A2A7

Now we fix the code in both places.









## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s

Now we save our code over to move on to Block #1. And then this is a done deal.

### Block #2 Encrypted Code:

```
0087A5DD 0E 56 32 DC B7 09 D4 DA C4 37 10 9C 93 EB 0E V2Ü·.ÖÜÄ7□œ"ë
Binary:
0E 56 32 DC B7 09 D4 DA C4 37 10 9C 93 EB 0E
```

And now moving on to Block #1, we would paste over our code, and modify the encryption.

Address	Hex dump	Disassembly
0087A103	8B33	MOV ESI,DWORD PTR DS:[EBX]
0087A105	66:81DF FF50	SBB DI,50FF
0087A10A	81EE 552B683F	SUB ESI,3F682B55
0087A110	81C6 6A894C55	ADD ESI,554C896A
0087A116	68 64BBAE08	PUSH 8AEBB64
0087A11B	0FBFF8	MOVSX EDI,AX
0087A11E	5A	POP EDX
0087A11F	81C6 5BFE7055	ADD ESI,5570FE5B
0087A125	E8 0D000000	CALL dvdZone2.0087A137
0087A12A	01A6 E7943D32	ADD DWORD PTR DS:[ESI+323D94E7],ESP
0087A130	8300 39	ADD DWORD PTR DS:[EAX],39
0087A133	^ 7E DF	JLE SHORT dvdZone2.0087A114
0087A135	2C F5	SUB AL,0F5
0087A137	^ E9 0D000000	JMP dvdZone2.0087A149
0087A13C	^ 71 56	JNO SHORT dvdZone2.0087A194
0087A13E	D7	XLAT BYTE PTR DS:[EBX+AL]
0087A13F	C4AD E27330A9	LES EBP,FWORD PTR SS:[EBP+A93073E2]
0087A145	2E:CF	IRET
0087A147	5C	POP ESP
0087A148	65:58	POP EAX
0087A14A	56	PUSH ESI
0087A14B	66:B8 C7F3	MOV AX,0F3C7
0087A14F	8F03	POP DWORD PTR DS:[EBX]
0087A151	0FB7D2	MOVSX EDX,DX
0087A154	83EB 04	SUB EBX,4
0087A157	66:BF EA96	MOV DI,96EA
0087A15B	83E9 01	SUB ECX,1
0087A15E	^ 0F85 9FFFFFFF	JNZ dvdZone2.0087A103
0087A164	0000	CALL EBP

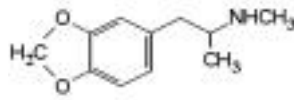
At the First Block.

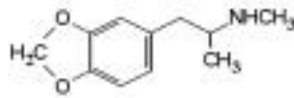
Address	Hex dump	Disassembly
0087A103	8B33	MOV ESI,DWORD PTR DS:[EBX]
0087A105	66:81DF FF50	SBB DI,50FF
0087A10A	81EE 5BFE7055	SUB ESI,5570FE5B
0087A110	81EE 6A894C55	SUB ESI,554C896A
0087A116	68 64BBAE08	PUSH 8AEBB64
0087A11B	0FBFF8	MOVSX EDI,AX
0087A11E	5A	POP EDX
0087A11F	81C6 552B683F	ADD ESI,3F682B55
0087A125	E8 0D000000	CALL dvdZone2.0087A137
0087A12A	01A6 E7943D32	ADD DWORD PTR DS:[ESI+323D94E7],ESP
0087A130	8300 39	ADD DWORD PTR DS:[EAX],39
0087A133	^ 7E DF	JLE SHORT dvdZone2.0087A114
0087A135	2C F5	SUB AL,0F5
0087A137	^ E9 0D000000	JMP dvdZone2.0087A149
0087A13C	^ 71 56	JNO SHORT dvdZone2.0087A194
0087A13E	D7	XLAT BYTE PTR DS:[EBX+AL]
0087A13F	C4AD E27330A9	LES EBP,FWORD PTR SS:[EBP+A93073E2]

Modified the Encryption.









## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s

Address	Hex dump	ASCII
0087A5D0	A3 E6 D5 86 4C 99 77 85 A9 40 AF FB 27 7B B2 3F	ûµFßLÖwãr@»r' (??
0087A5E0	14 4A AF 85 C1 40 AF 85 29 0A B4 85 29 45 AF 85	ŲJ»ã+@»ã).+ã)E»ã
0087A5F0	29 45 A0 0E 1C E5 6A 0E 95 48 56 0C FC 41 3E 08	)EãßLσjßòHV. "A»
0087A600	A2 6E 9D 0D 27 AE B1 83 97 43 59 BF E5 57 21 A2	ón#. "«ßûCVŷσw†ó
0087A610	1F 47 21 A0 24 CF 89 CF F9 17 4B BA BA 05 0F A3	▲A†ãz=ã=.ãk■rã«ã

Original code.

Address	Hex dump	ASCII
0087A5D0	A2 E6 D5 86 4C 99 77 85 59 C7 B3 46 28 7B B2 3F	ûµFßLÖwãrVIF (??
0087A5E0	14 4A AF 85 C1 40 AF 85 29 0A B4 85 29 45 AF 85	ŲJ»ã+@»ã).+ã)E»ã
0087A5F0	29 45 A0 0E 1C E5 6A 0E 95 48 56 0C FC 41 3E 08	)EãßLσjßòHV. "A»
0087A600	A2 6E 9D 0D 27 AE B1 83 97 43 59 BF E5 57 21 A2	ón#. "«ßûCVŷσw†ó
0087A610	1F 47 21 A0 24 CF 89 CF F9 17 4B BA BA 05 0F A3	▲A†ãz=ã=.ãk■rã«ã

Patch code.

### DO YOU SEE ANY DIFFERENCES?

You do? Sure you do, that's our patch code. Ignore the A2, this one doesn't matter either A2 or A3 would turn out the same. We only care about the "59 C7 B3 46 28" so those 5 bytes are our code. Lets see if it works shall we? So remove the A3- A2 patch, it doesn't need to be here, and only keep the 5 bytes after it changed. And then Set a BP on VirtualAlloc, after the 2<sup>nd</sup> break look down at our offset at "0087A5E3" which is what we are trying to patch.

Address	Hex dump	Disassembly
0087A510	8985 31040000	MOV DWORD PTR SS:[EBP+431],EAX
0087A516	8985 00010000	MOV DWORD PTR SS:[EBP+100],EAX
0087A51C	64:67:A1 0000	MOV EAX,DWORD PTR FS:[0]
0087A521	8985 2D040000	MOV DWORD PTR SS:[EBP+42D],EAX
0087A527	8B55 5B	MOV EDX,DWORD PTR SS:[EBP+5B]
0087A52A	8B85 00010000	MOV EAX,DWORD PTR SS:[EBP+100]
0087A530	8902	MOV DWORD PTR DS:[EDX],EAX
0087A532	8B85 00040000	MOV EAX,DWORD PTR SS:[EBP+408]
0087A538	8942 04	MOV DWORD PTR DS:[EDX+4],EAX
0087A53B	8D85 9F030000	LEA EAX,DWORD PTR SS:[EBP+39F]

Do ALT+F9 after 2<sup>nd</sup> break on VirtualAlloc (You should be here)

Scroll down.

0087A5E0	03DF	ADD EBX,EDI
0087A5E2	53	PUSH EBX
0087A5E3	68 00B08900	PUSH dword2one2.0089B000
0087A5E8	C3	RETN
0087A5E9	0056 FF	ADD BYTE PTR DS:[ESI-1],DL
0087A5EC	95	XCHG EAX,EBP
0087A5ED	F4	HLT
0087A5EE	0300	ADD EAX,DWORD PTR DS:[EAX]



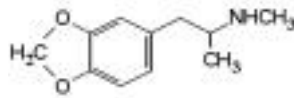
It works.

Now we can proceed to phase 2 of our evil plan..... Since we now know the bytes we need to totally bypass the decryptor blocks, we can now proceed to examine our target more in depth and get the rest of the info we need. What we will need is the following:

### Rest of Needed Info:

1. Three VirtualAlloc Blocks, so we know when we hit the Aspr.dll.
2. CRC Check, which we now need to fix only 5 bytes from our encryption.
3. Would normally be how to fix the registration, in this particular app it uses its own system, so I will not cover this here, remember... I am not teaching you to crack, but rather the way to bypass a particular protection system.





## Synopsis on Asprotect Patching by MaDMAn\_H3rCuL3s

So now we must know our target, let us restart it and get the rest of the needed info so we can proceed to have this target running with our patch applied. So restart it, and BP VirtualAlloc, after second break (we already have reached this with our patch), hit F9 once more to pop on next Allocation.

Address	Hex dump	Disassembly	Comment
0087A4FD	68 00100000	PUSH 1000	
0087A502	FFB5 08040000	PUSH DWORD PTR SS:[EBP+408]	
0087A508	6A 00	PUSH 0	
0087A50A	FF95 F0030000	CALL DWORD PTR SS:[EBP+3F0]	kernel32.VirtualAlloc
0087A510	8985 31040000	MOV DWORD PTR SS:[EBP+431],EAX	
0087A516	8985 D0010000	MOV DWORD PTR SS:[EBP+1D0],EAX	
0087A51C	64:67:A1 0000	MOV EAX,DWORD PTR FS:[0]	
0087A521	8985 2D040000	MOV DWORD PTR SS:[EBP+42D],EAX	
0087A527	8B55 5B	MOV EDX,DWORD PTR SS:[EBP+5B]	dvd2one2.0087A048
0087A52A	8B85 D0010000	MOV EAX,DWORD PTR SS:[EBP+1D0]	
0087A530	8902	MOV DWORD PTR DS:[EDX],EAX	

Below this is what we already accomplished. So we hit F9 once more, then ALT+F9.

Address	Hex dump	Disassembly	Comment
00ECE088	68 00100000	PUSH 1000	
00ECE08D	68 46050000	PUSH 546	
00ECE0C2	6A 00	PUSH 0	
00ECE0C4	FF95 79294400	CALL DWORD PTR SS:[EBP+442979]	kernel32.VirtualAlloc
00ECE0CA	8985 75294400	MOV DWORD PTR SS:[EBP+442975],EAX	
00ECE0D0	8D9D 452A4400	LEA EBX,DWORD PTR SS:[EBP+442A45]	
00ECE0D6	50	PUSH EAX	
00ECE0D7	53	PUSH EBX	
00ECE0D8	E8 74050000	CALL 00ECE651	
00ECE0DD	8BC8	MOV ECX,EAX	
00ECE0DF	8D8D 452A4400	LEA EDI,DWORD PTR SS:[EBP+442A45]	
00ECE0E5	8B85 75294400	MOV ESI,DWORD PTR SS:[EBP+442975]	
00ECE0EB	F3:A4	REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:	
00ECE0ED	8B85 75294400	MOV EAX,DWORD PTR SS:[EBP+442975]	
00ECE0F3	68 00000000	PUSH 0000	
00ECE0F8	6A 00	PUSH 0	

We are now in Allocated space now, as you can tell from the offsets.

Now we will hit F9 once more then followed by ALT+F9.

Address	Hex dump	Disassembly	Comment
00ECE33B	68 00100000	PUSH 1000	
00ECE340	50	PUSH EAX	
00ECE341	6A 00	PUSH 0	
00ECE343	FF95 79294400	CALL DWORD PTR SS:[EBP+442979]	kernel32.VirtualAlloc
00ECE349	8985 75294400	MOV DWORD PTR SS:[EBP+442975],EAX	
00ECE34F	56	PUSH ESI	
00ECE350	8B1E	MOV EBX,DWORD PTR DS:[ESI]	
00ECE352	039D D8304400	ADD EBX,DWORD PTR SS:[EBP+4430D8]	
00ECE358	50	PUSH EAX	
00ECE359	53	PUSH EBX	
00ECE35A	E8 F2020000	CALL 00ECE651	
00ECE35F	80BD 70294400	CMP BYTE PTR SS:[EBP+442970],0	
00ECE366	75 4C	JNZ SHORT 00ECE3B4	

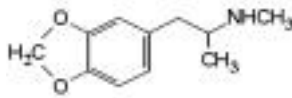
Now we scroll down a bit, and look for the ASPACK code.

00ECE5B8	8985 112F4400	MOV DWORD PTR SS:[EBP+442F11],EAX
00ECE5C1	61	POPAD
00ECE5C2	75 08	JNZ SHORT 00ECE5CC
00ECE5C4	B8 01000000	MOV EAX,1
00ECE5C9	C2 0C00	RETN 0C
00ECE5CC	68 00000000	PUSH 0
00ECE5D1	C3	RETN

And here it is ☺ Upon executing the RETN here, we will be in the ASPR.DLL.







## Synopsis on Asprotect Patching by MaDMAn\_H3rCuL3s

So set a break on the POPAD, then break and use F7 until we execute the RETN.

00EC5B8	8985 112F4400	MOV DWORD PTR SS:[EBP+442F11],EAX
00EC5C0	61	POPAD
00EC5C2	75 08	JNZ SHORT 00EC5CC
00EC5C4	B8 01000000	MOV EAX,1
00EC5C9	C2 0C00	RETN 0C
00EC5CC	68 C450EC00	PUSH 0EC50C4
00EC5D1	C3	RETN
00EC5E0	0000 00000000	MOV DWORD PTR SS:[EBP+442F11],0

Then F7.....

Address	Hex dump	Disassembly
00EC50C4	55	PUSH EBP
00EC50C5	8BEC	MOV EBP,ESP
00EC50C7	83C4 B4	ADD ESP,-4C
00EC50CA	B8 A44EEC00	MOV EAX,0EC4EA4
00EC50CF	E8 3806FEFF	CALL 00EA570C
00EC50D4	E8 C3E4FDFF	CALL 00EA359C
00EC50D9	8D40 00	LEA EAX,DWORD PTR DS:[EAX]
00EC50DC	0000	ADD BYTE PTR DS:[EAX],AL
00EC50DE	0000	ADD BYTE PTR DS:[EAX],AL
00EC50E0	0000	ADD BYTE PTR DS:[EAX],AL
00EC50E3	0000	ADD BYTE PTR DS:[EAX],AL

And here we are.

Now we must set a new BP on MapViewOfFileEx so we can break after the mapped image is created.

Depending on your system it may take you only the first break, on mine I break 2 times. In the stack you can see we are returning to a real location.

Address	Value	Comment
0006FDDC	7C80B7A8	CALL to MapViewOfFileEx from kernel32.7C80B7A8
0006FDE0	00000084	hMapObject = 00000084 (window)
0006FDE4	00000004	AccessMode = FILE_MAP_READ
0006FDE8	00000000	OffsetHigh = 0
0006FDEC	00000000	OffsetLow = 0
0006FDF0	00000000	MapSize = 0
0006FDF4	00000000	BaseAddr = NULL
0006FDF8	0006FE2C	
0006FDFC	00EB7C8B	RETURN to 00EB7C8B
0006FF00	00000000	

Hit ALT+F9 and we are here:

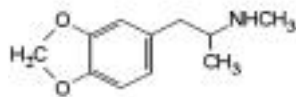
00EB7C60	0000 00000000	CALL 00EB7C60
00EB7C73	6A 00	PUSH 0
00EB7C75	6A 00	PUSH 0
00EB7C77	6A 00	PUSH 0
00EB7C79	6A 04	PUSH 4
00EB7C7B	A1 1C84EC00	MOV EAX,DWORD PTR DS:[EC841C]
00EB7C80	50	PUSH EAX
00EB7C81	A1 DC67EC00	MOV EAX,DWORD PTR DS:[EC67DC]
00EB7C86	8B40 08	MOV EAX,DWORD PTR DS:[EAX+8]
00EB7C89	FFD0	CALL EAX
00EB7C8B	8BD8	MOV EBX,EAX
00EB7C8D	891D 1884EC00	MOV DWORD PTR DS:[EC8418],EBX
00EB7C93	A1 9466FC00	MOV EAX,DWORD PTR DS:[FC6694]

Okay to make it easier for you to understand here...

If you look closely you see 3 PUSH 0 followed by a PUSH 4. The PUSH 4 is the (READ\_ONLY\_ACCESS Flag). So we must patch this to PUSH 1 (READ\_WRITE Flag). So we may patch in memory, the code we touched to get this far. The CALL EAX, well this is our CALL MapViewOfFileEx. So in EAX, we will hold our mapped imagebase. So when we do patch this, we will patch the PUSH 4, and the MOV EBX, EAX. The reason for this is that after we create the

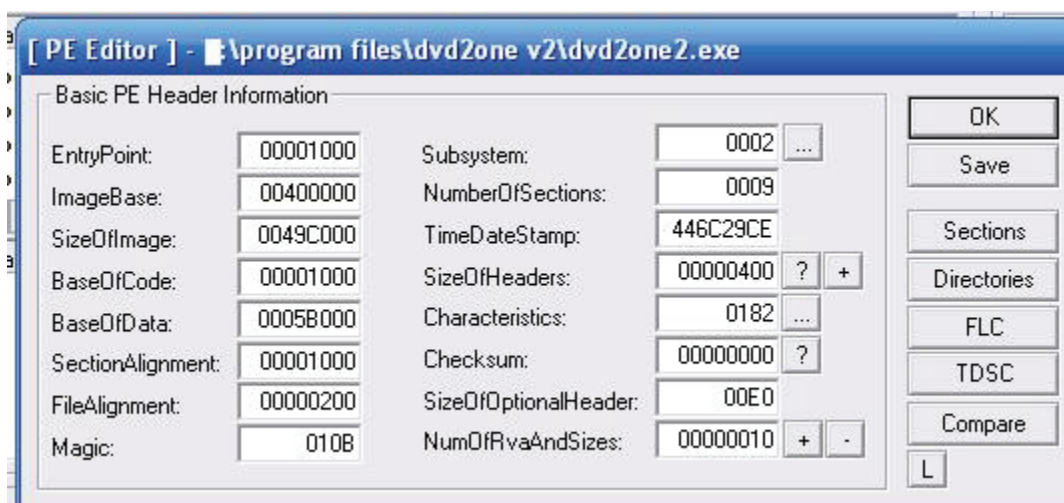






## Synopsis on Asprotect Patching by MaDMAn\_H3rCuL3s

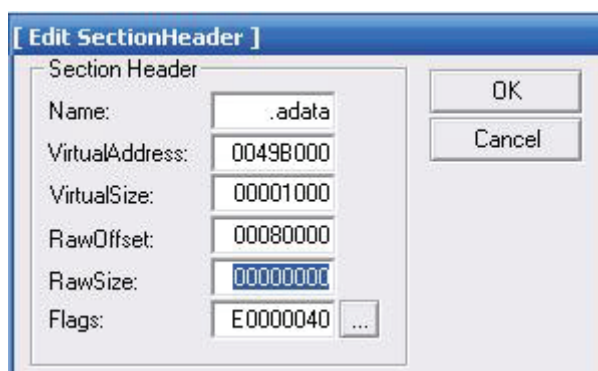
space as READ\_WRITE, we can redirect it, and get the imagebase, and then patch the image so it thinks it is an original exe. We now have all the info we need to continue our journey. So now we must modify the exe, so we have some extra space to add our code. This is taken from JohnWho's method. Using LORD-PE, we will add 1000 bytes to the .adata section then fix the 00's with NOP's so it is a valid executable. So save a copy of this original exe, then open up LORD-PE and open up the exe, then click on the sections tab and on the .adata section add 1000 bytes to the raw image. Follow along below:



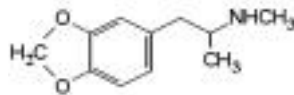
Click the Sections button.

Name	VOffset	VSize	ROffset	RSize	Flags
	0005D000	00029000	0002C000	00012600	E0000040
	00086000	00359000	00000000	00358200	E0000040
	003DF000	00001000	0003E600	00000200	E0000040
	003E0000	00006000	0003E800	00000000	E0000040
.rsrc	003E6000	00094000	0003E800	00020C00	E0000040
.data	0047A000	00021000	0005F400	00020C00	E0000040
.adata	0049B000	00001000	00080000	00000000	E0000040

Now right click the .adata section and "Edit Section Header".



Before.



## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s

[ Edit SectionHeader ]

Section Header	
Name:	.adata
VirtualAddress:	0049B000
VirtualSize:	00001000
RawOffset:	00080000
RawSize:	1000
Flags:	E0000040 ...

OK  
Cancel

After.

Click “OK” then save it all, then open up HexWorkshop, and again open up the modified exe.

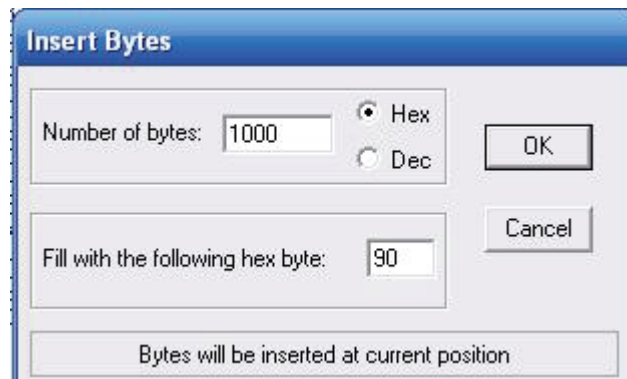
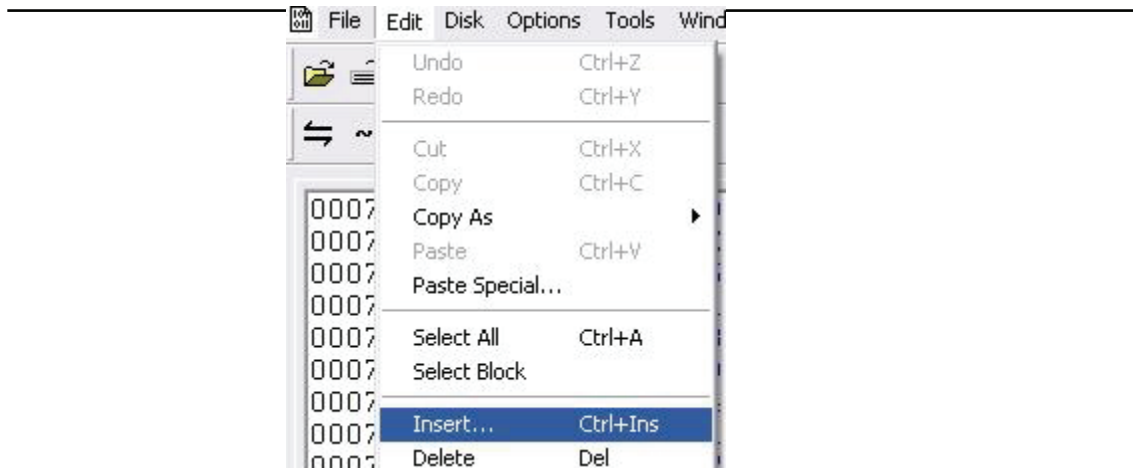
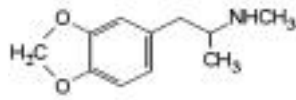
00000000	AD5A	8000	0100	0000	0400	0000	FFFF	0000	MZ.....
00000010	B800	0000	0000	0000	4000	0000	0000	0000	.....@.....
00000020	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000030	0000	0000	0000	0000	0000	0000	8000	0000	.....
00000040	0E1F	BA0E	00B4	09CD	21B8	014C	CD21	7468	.....!..L.!th
00000050	6973	2069	7320	6120	5769	6E64	6F77	7320	is is a Windows
00000060	4E54	2077	696E	646F	7765	6420	6578	6563	NT windowed exec
00000070	7574	6162	6C65	0D0A	2400	0000	0000	0000	utable..\$......
00000080	5045	0000	4C01	0900	CE29	6C44	0000	0000	PE..L....)lD....
00000090	0000	0000	E000	8201	0B01	0212	009C	0500	.....
000000A0	00AC	0200	0082	3500	0010	0000	0010	0000	.....5.....
000000B0	00B0	0500	0000	4000	0010	0000	0002	0000	.....@.....
000000C0	0100	0000	0000	0000	0400	0000	0000	0000	.....

Scroll down to the bottom and then do the following.

0007FF90	7AF0	8B21	B0F2	099B	FE31	7175	DC2F	D455	z..!.....lqu../.U
0007FFA0	E7A7	F767	5301	7FDB	C4B6	F8D0	59BE	B64D	...gS.....Y..M
0007FFB0	F21B	A61B	0420	91B1	761C	6778	756F	9505	.....v.gxuo..
0007FFC0	1D40	7B39	6B4D	854B	F19C	7A27	1B02	06B8	.@{9kM.K..z'....
0007FFD0	08B0	98F9	6065	0DEB	8CD0	C4CE	3A4F	2332	....`e.....:0#2
0007FFE0	2A1F	8FC1	9799	75E1	C41D	E383	77B8	B8BD	*.....u.....w...
0007FFF0	78C1	9112	53DB	D772	F228	A76F	DC9E	7618	x...S..r.(.o..v.
00080000									

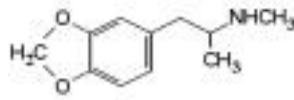
Put the cursor at the last byte, which is here, 00080000.

Then on the menu click the “EDIT – INSERT”



00080EC0	9090	9090	9090	9090	9090	9090	9090	9090
00080ED0	9090	9090	9090	9090	9090	9090	9090	9090
00080EE0	9090	9090	9090	9090	9090	9090	9090	9090
00080EF0	9090	9090	9090	9090	9090	9090	9090	9090
00080F00	9090	9090	9090	9090	9090	9090	9090	9090
00080F10	9090	9090	9090	9090	9090	9090	9090	9090
00080F20	9090	9090	9090	9090	9090	9090	9090	9090
00080F30	9090	9090	9090	9090	9090	9090	9090	9090
00080F40	9090	9090	9090	9090	9090	9090	9090	9090
00080F50	9090	9090	9090	9090	9090	9090	9090	9090
00080F60	9090	9090	9090	9090	9090	9090	9090	9090
00080F70	9090	9090	9090	9090	9090	9090	9090	9090
00080F80	9090	9090	9090	9090	9090	9090	9090	9090
00080F90	9090	9090	9090	9090	9090	9090	9090	9090
00080FA0	9090	9090	9090	9090	9090	9090	9090	9090
00080FB0	9090	9090	9090	9090	9090	9090	9090	9090
00080FC0	9090	9090	9090	9090	9090	9090	9090	9090
00080FD0	9090	9090	9090	9090	9090	9090	9090	9090
00080FE0	9090	9090	9090	9090	9090	9090	9090	9090
00080FF0	9090	9090	9090	9090	9090	9090	9090	9090
00081000								





## Synopsis on Asprotect Patching by MaDMAn\_H3rCuL3s

Make sure you save the file also, and know that from now on this executable is going to nag you about a CRC check. Until it's fixed this exe will not run.

So we restart again, at the EP of the modified program. Let us add our 5 bytes we discovered to autopatch it halfway. Go to our offset

0087A5E3	77 85	JA SHORT dvd2one2.0087A56A
0087A5E5	A9 40AFFB27	TEST EAX,27FBAF40
0087A5EA	7B B2	JPO SHORT dvd2one2.0087A59E
0087A5EC	3F	AAS
0087A5ED	14 4A	ADC AL,4A

Original!

Address	Hex dump	Disassembly
0087A5E3	77 85	JA SHORT dvd2one2.0087A56A
0087A5E5	59	POP ECX
0087A5E6	C7	???
0087A5E7	B3 46	MOV BL,46
0087A5E9	287B B2	SUB BYTE PTR DS:[EBX-4E],BH
0087A5EC	3F	AAS
0087A5ED	14 4A	ADC AL,4A
0087A5EF	AF	SCAS DWORD PTR ES:[EDI]

FIXED! And then save it.

Now we will break on offset "0089B000"

So we have just bypassed about 4-5 extras steps, which does really mean a lot. Now we have more time to smoke ☺. But anyways, we again restart the target, set a Bp on our pushed offset, just to make sure we break.

Address	Hex dump	Disassembly
0089B000	90	NOP
0089B001	90	NOP
0089B002	90	NOP
0089B003	90	NOP
0089B004	90	NOP
0089B005	90	NOP
0089B006	90	NOP

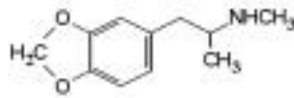
BREAK!

Good we now got this fixed. Now onto the rest of the patch. We remember we broke a few times on VirtualAlloc, then MapViewOfFileEx. Well we will have to wait for the Aspr.dll to fully decrypt itself, so then we can patch it in memory. Let us look at our Break we did on VirtualAlloc real closely.

VirtualAlloc		
00ECE0ED	8B85 75294400	MOV EAX,DWORD PTR SS:[EBP+442975]
00ECE0F3	68 00800000	PUSH 8000
00ECE0F8	6A 00	PUSH 0
00ECE0FA	50	PUSH EAX
00ECE0FB	FF95 7D294400	CALL DWORD PTR SS:[EBP+44297D] ; kernel32.VirtualFree

So what we will do here (Mind you your offsets and mine will vary), we are going to create a patched PUSH (offset) at the PUSH 8000, then followed by a RETN. So let's think about it quickly, we are now at the offset we just patched into the disk based exe, we should patch in now the next area of relocation. But do you see these offsets, they are unreal offsets. Luckily for us this problem was already solved in JohnWho's tutorial which is why I had it redirect me from the PUSH





## Synopsis on Asprotect Patching by MaDMan\_H3rCuL3s

8000. At this offset EDI equals our base address for this layer of encryption. Which in my case is "00EA0000"

```
ESI 00E70000
EDI 00EA0000
EIP 00000000
```

We need to use this offset.

So now in our code, we will add this line:

Address	Hex dump	Disassembly
0089B000	893D 5DBF8900	MOV DWORD PTR DS:[89BF5D],EDI
0089B006	90	NOP

So now we pick a place in our code area, to save the value, for later use of course.

Our new mission is to repair our entered place, meaning that PUSH we created, well we still need to execute it, if your feeling lucky you can patch it back in, or you can use this code area to execute it. Here we will patch it back into its rightful place. So we now add the following lines of code:

Address	Hex dump	Disassembly
0089B000	893D 5DBF8900	MOV DWORD PTR DS:[89BF5D],EDI
0089B006	66:C705 E8A58700 8000	MOV WORD PTR DS:[87A5E5],80
0089B00F	C605 E8A58700 6A	MOV BYTE PTR DS:[87A5E8],6A
0089B016	90	NOP

We only have to fix 3 bytes.

Now we must patch the next area we intend to go to, in my case it was

```
00ECE0F3 68 00800000    PUSH 8000
00ECE0F8 6A 00           PUSH 0
```

So my base is 00EA0000, my place is 00ECE0F3:

$00ECE0F3 - 00EA0000 = 2E0F3$

0089B00F	C605 E8A58700 6A	MOV BYTE PTR DS:[87A5E8],6A
0089B016	C787 F3E00200 682EB089	MOV DWORD PTR DS:[EDI+2E0F3],89B02E68
0089B020	66:C787 F7E00200 00C3	MOV WORD PTR DS:[EDI+2E0F7],0C300
0089B029	E9 B5F5F0FF	JMP dvdZone2.0087A5E3
0089B02F	90	NOP

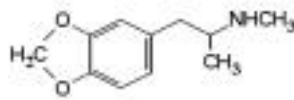
So we use that offset. Then we JMP back to the PUSH 8000 we redirected in the first place.

Now if we save changes we made, then restart it, set a BP on the NOP after the JMP (0089B02E), we can now assemble some more code. We need to first make sure we put back the code we took away just before this, so we will assemble this:

Address	Hex dump	Disassembly
0089B000	893D 5DBF8900	MOV DWORD PTR DS:[89BF5D],EDI
0089B006	66:C705 E8A58700 8000	MOV WORD PTR DS:[87A5E5],80
0089B00F	C605 E8A58700 6A	MOV BYTE PTR DS:[87A5E8],6A
0089B016	C787 F3E00200 682EB089	MOV DWORD PTR DS:[EDI+2E0F3],89B02E68
0089B020	66:C787 F7E00200 00C3	MOV WORD PTR DS:[EDI+2E0F7],0C300
0089B029	E9 B5F5F0FF	JMP dvdZone2.0087A5E3
0089B02E	90	NOP
0089B02F	90	NOP
0089B030	90	NOP
0089B031	90	NOP
0089B032	90	NOP
0089B033	90	NOP

Break on the Place after the JMP.





## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s

```
0089B029 - E9 B5F5D0FF JMP dvdZone2.0087A5E3
0089B02E C743 F2 68000000 MOV DWORD PTR DS:[EBX-E],dvdZone2.00800068 PATCH BACK THE PUSH 8000
0089B035 66:C743 F6 006A MOV WORD PTR DS:[EBX-A],6A00 ^^
0089B03B 90 NOP
```

Restoring the code back.

I chose EBX only because here it is almost exactly where we want to patch sub E from it though.

```
EAX 7C90EB94
EBX 00ECE101
ESP 0006FF6C
```

SEE ☺

So assemble what you see above, and then we continue to the POPAD.

We now need to patch to the POPAD we discussed a bit earlier here. So in this place we will assemble another PUSH (Offset), followed by another RETN. Look below for exactly how we do it.

```
0089B03B 66:C743 F6 006A MOV WORD PTR DS:[EBX-A],6A00
0089B03B C783 C0400000 685FB089 MOV DWORD PTR DS:[EBX+4C0],89B05F68
0089B045 66:C783 C4040000 00C3 MOV WORD PTR DS:[EBX+4C4],0C3000
0089B04E 53 PUSH EBX
```

So we are moving code to create a PUSH 0089B05F.

Now exactly how you want to return back is completely up to you. We can either save all our registers or even just save one. For this particular patch, I chose only one register. Since we know EBX is almost exactly what we need, we use this one.

```
0089B04E 53 PUSH EBX
0089B04F 83EB 0E SUB EBX,0E
0089B052 90 NOP
```

Save EBX, then sub E from it.

So now EBX is equal to our return destination. We must somehow get there now though. For this patch I chose to use a JMP.

```
0089B052 891D 00B58900 MOV DWORD PTR DS:[89B500],EBX
0089B058 5B POP EBX
0089B059 FF25 00B58900 JMP DWORD PTR DS:[89B500]
0089B05F 90 NOP
```

So here we are going to move the value in EBX to another offset I just randomly thought up, then restoring EBX to its original value, then finally JMP'ing to the offset contained inside the pointer.

Now that this is done, we can save everything, then restart it again; set a BP on the offset 0089B05F.

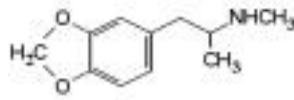
```
0089B058 5B POP EBX
0089B059 FF25 00B58900 JMP DWORD PTR DS:[89B500]
0089B05F 90 NOP
0089B060 90 NOP
0089B061 90 NOP
```

We have broken in our place.

Now to modify the CRC Check. Finally we are at the end of this long, yet informative tutorial. So if we look at what we had in our previous scope, we know we had to modify a PUSH 4 → PUSH 1, then after the CALL EAX, we create a PUSH Offset, followed by a RETN.







## Synopsis on Asprotect Patching by MaDMAn\_H3rCuL3s

```

00EB7C77 6A 00      PUSH 0
00EB7C79 6A 04      PUSH 4
00EB7C7B A1 1C84EC00 MOV EAX,D

```

We see that we need modify just one byte at offset 00EB7C7A (04 → 01).

So again we do some math.  $00EB7C7A - 00EA0000 = 17C7A$ . So, remember when we stored the value at EDI and saved it? Well now we will use it, we could in however use another register (for starters I used EBX again), but about every 4<sup>th</sup> run, it will crash. So to keep this patch working we will use what we have already accounted for. It does not matter what register you put the base address into, but always remember to save the register first, so it can be restored, or else your better off just leaving it like I mentioned earlier, and let it crash every 4<sup>th</sup> time. So we add the following code:

```

0089B059 FF25 00B58900 JMP DWORD PTR DS:[89B500]
0089B05F 50          PUSH EAX
0089B060 A1 5DBF8900 MOV EAX,DWORD PTR DS:[89BF5D]
0089B065 90          NOP

```

So we save EAX, and then move our base to it.

Now we must restore out patched code. So that POPAD we changed to a PUSH, will need to be reverted back to original flow, to keep us again from crashing. Follow along with the pictures to better help make you understand.

```

0089B065 C780 C1E50200 617508B8 MOV DWORD PTR DS:[EAX+2E5C1],B8087561
0089B06F 66:C780 C5E50200 0100 MOV WORD PTR DS:[EAX+2E5C5],1
0089B078 90          NOP

```

Again you would do some simple math here, and put back our code.

```

0089B06F 66:C780 C5E50200 0100 MOV WORD PTR DS:[EAX+2E5C5],1
0089B078 C680 7A7C0100 01 MOV BYTE PTR DS:[EAX+17C7A],1
0089B07F 90          NOP

```

Patch the PUSH 4 → PUSH 1

```

0089B07F C780 8B7C0100 68A3B089 MOV DWORD PTR DS:[EAX+17C8B],89B0A368
0089B089 66:C780 8F7C0100 00C3 MOV WORD PTR DS:[EAX+17C8F],0C300

```

Patch the code after the CALL EAX.

```

0089B092 05 C1E50200 ADD EAX,2E5C1
0089B097 A3 00B58900 MOV DWORD PTR DS:[89B500],EAX
0089B09C 58          POP EAX
0089B09D FF25 00B58900 JMP DWORD PTR DS:[89B500]
0089B0A3 90          NOP

```

Restore our register, and JMP back to the POPAD.

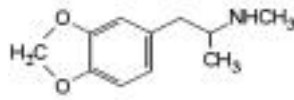
So now save the changes and then restart it, put BP on the offset after our RETN JMP, and now look in EAX.

```

0089B09D FF25 00B58900 JMP DWORD PTR DS:[89B500]
0089B0A3 90          NOP
0089B0A4 90          NOP
0089B0A5 90          NOP

```





## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s

Break on the BP.

```
EAX 01020000
ECX 0006FDA0
```

EAX = our Mapped Executable.

Well folks, we have finally reach the ending here. We now must do a few things:

1. Patch back the code we just used (PUSH 4, MOV EBX, EAX).
2. Patch the Mapped image so it thinks the disk based executable is just fine.

So first let's patch the disks mapped image, we had to add 1000 bytes to the header, speaking of .adata section. So we need to fix one byte there, to change a "10" to a "00", then we must patch back those 5 bytes we used to redirect us to the patch location. (Remember the beginning of this tutorial).

```
010202B0 00 00 00 00 40 00 00 E0 2E 61 64 61 74 61 00 00 .....adata...
010202C0 00 10 00 00 00 B0 49 00 00 10 00 00 00 00 08 00 .....I.....
010202D0 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 E0 .....@..@..
```

Here the byte we need to modify is highlighted for you.

So we must do this:

EAX = 01020000

Offset = 010202C9

$$010202C9 - 01020000 = 2C9$$

```
0089B09D - FF25 00B58900 JMP DWORD PTR DS:[89B500]
0089B0A3 C680 C9020000 00 MOV BYTE PTR DS:[EAX+2C9],0
0089B0AA 90 NOP
0089B0AB 90 NOP
```

We fix the headers byte.

```
0107F9E2 99 CDW
0107F9E3 ^ 77 85 JA SHORT 0107F96A
0107F9E5 59 POP ECX
0107F9E6 C7 ???
0107F9E7 B3 46 MOV BL,46
0107F9E9 287B B2 SUB BYTE PTR DS:[EBX-4E],BH
0107F9EC 3F AAS
0107F9ED 14 4A ADC AL,4A
0107F9EF 0F SCAS DWORD PTR ES:[EDI]
```

Remember this?

So we do this:

```
0089B0AA C780 E5F90500 A940AFFF MOV DWORD PTR DS:[EAX+5F9E5],FBAF40A9
0089B0B4 C680 E9F90500 27 MOV BYTE PTR DS:[EAX+5F9E9],27
0089B0BB 90 NOP
0089B0BD 90 NOP
```

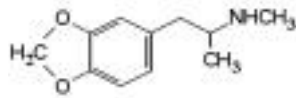
Easy ☺

Now we must do the some patching of the redirected code. Again we must save a register, and then move our value we stored, and then replace the PUSH 4 and MOV EBX, EAX.

```
0089B0B4 C680 E9F90500 27 MOV BYTE PTR DS:[EAX+5F9E9],27
0089B0BB 50 PUSH EAX
0089B0BC A1 5DBF8900 MOV EAX,DWORD PTR DS:[89BF5D]
0089B0C1 90 NOP
0089B0C3 90 NOP
```

Save our register, and then use our stored base address.





## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s

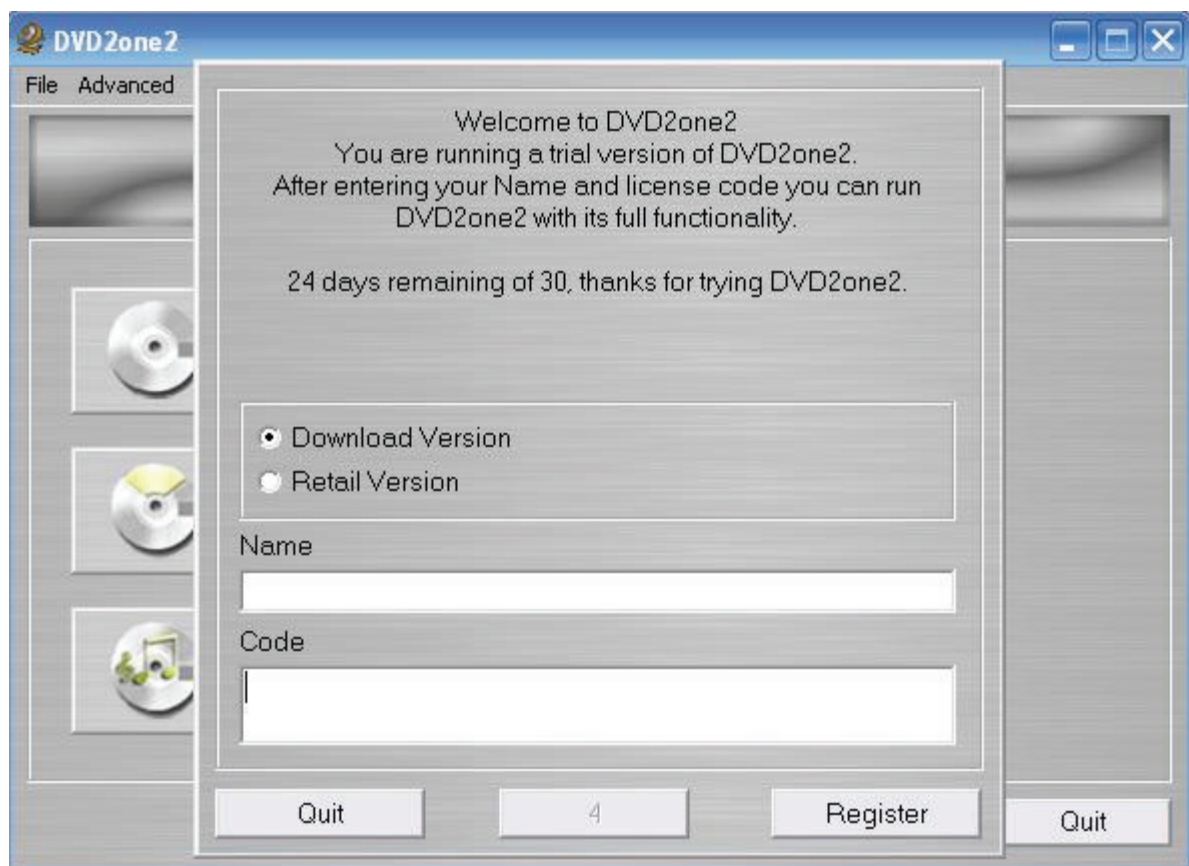
```
0089B0C1  C680 7A7C0100 04      MOV BYTE PTR DS:[EAX+17C7A],4
0089B0C8  C780 8B7C0100 8BD8891D  MOV DWORD PTR DS:[EAX+17C8B],1D89D88B
0089B0D2  66:C780 8F7C0100 1884    MOV WORD PTR DS:[EAX+17C8F],8418
```

Like before, but we move original code back.

```
0089B0D8  05 8B7C0100      ADD EAX,17C8B
0089B0E0  A3 00B58900      MOV DWORD PTR DS:[89B500],EAX
0089B0E5  58              POP EAX
0089B0E6  FF25 00B58900    JMP DWORD PTR DS:[89B500]
```

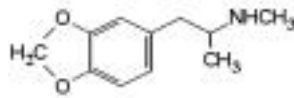
Now we restore the register, then jmp to our original destination.

We then save it, and run it, then.....



SUCCESS!!





## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s

Okay it was an extremely long part 1 of this tutorial. I have yet another surprise left. I am going to tell you how to patch Asprotect v2.2 – 2.3 SKE. Yes I will finally reveal my secret. It is almost same basic idea as 2.1 but now we get into nice allocations. Instead of the usual Asprotect allocations, and the simple CALL we can return from, we are now hit up with crazy stuff, where you eventually hit the call, but from a very strange location, I will go over my generic approach, which I will assume that the author will change once this tutorial is published, as he did with shoooo when he attacked the activation. So now the tutorial title changes from synopsis on Asprotect, to “ATTACK on ASPROTECT SKE REVISITED” and it is dedicated to shoooo, as he is my dog (woof woof). So when the latest build does eventually meet the community, I would ask if someone can be so kind as to donate a copy, with key, as I can continue my study into Asprotect. Or if Asprotect author hits me up with a job offer ☺ because I am sure that the author would read this to make his protection that much better. So when we hit up the next part of this tutorial, the target is going to be “Asprotect v2.3 SKE beta 426” (latest build as of this writing that I know of), soon to be Asprotect SKE v2.4 after this tutorial I am sure.

### 3. ATTACK ON ASPROTECT SKE REVISITED

We start out here:

Address	Hex dump	Disassembly
00401000	68 01006200	PUSH ASProtec.0062D001
00401005	E8 01000000	CALL ASProtec.0040100B
0040100A	C3	RET
0040100B	C3	RET
0040100C	B9	DB B9
0040100D	...	...

You know this place ☺

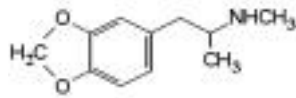
Because you made it this far, I can only assume you know what to do now. So I will only start from the POPAD, since we can all reach this ( ☺ ) (HINT: There is only 4 blocks here) not the 5. Once you reach the POPAD the fun really begins. We will do a few extra things as far as patching to fix the newly revised CRC check. This check is not too hard, although it did take me some time to actually figure it out, and also develop a “Generic” approach. So my way specifically deals with byte searches. Once found we note these offsets, and then we will create a test.

#### Asprotect SKE v2.3 Build 426 Blocks:

```
BLOCK 1.  
SUB ECX,3CB346D4  
XOR ECX,13E1F07D  
XOR ECX,0C27AB72
```

```
BLOCK 2.  
ADD EDX,32CD05A3  
ADD EDX,581540A0  
ADD EDX,167D7459
```





## Synopsis on Asprotect Patching by MaDMAn\_H3rCuL3s

```
BLOCK 3.  
XOR EAX,383A966  
SUB EAX,79064A7  
SUB EAX,774FED54
```

```
BLOCK 4.  
XOR EDI,5569C48D  
ADD EDI,69F9D642  
SUB EDI,37B48653
```

### Asprotect SKE v2.3 Build 426 Reversed Blocks:

```
BLOCK 4:  
ADD EDI,37B48653  
SUB EDI,69F9D642  
XOR EDI,5569C48D
```

```
BLOCK 3:  
ADD EAX,774FED54  
ADD EAX,79064A7  
XOR EAX,383A966
```

```
BLOCK 2:  
SUB EDX,167D7459  
SUB EDX,581540A0  
SUB EDX,32CD05A3
```

```
BLOCK 1:  
XOR ECX,0C27AB72  
XOR ECX,13E1F07D  
ADD ECX,3CB346D4
```

0062D5A4	53	ADD EBX,EDI
0062D5AC	53	PUSH EBX
0062D5AD	68 00800000	PUSH 8000
0062D5B2	6A 00	PUSH 0
0062D5B4	56	PUSH ESI

VirtualAlloc: Before Patch

0062D5A4	53	PUSH EBX
0062D5AD	68 00E06800	PUSH ASProtect.0068E000
0062D5B2	C3	RETN
0062D5B3	90	NOP
0062D5B4	56	PUSH ESI

VirtualAlloc: what we want it to do

AND THEN.....

0062D5A4	005E 00	MOV EBX,EDWORD PTR DS:[ESI+0]
0062D5A9	03DF	ADD EBX,EDI
0062D5AC	53	PUSH EBX
0062D5AD	68 00E06800	PUSH ASProtect.0068E000
0062D5B2	C3	RETN
0062D5B3	0056 FF	ADD BYTE PTR DS:[ESI-1],DL
0062D5B6	95	XCHG EAX,EBP
0062D5B7	F4	HLT
0062D5B8	0300	ADD EAX,EDWORD PTR DS:[EAX]
0062D5BA	0068 00	ADD BYTE PTR DS:[EAX],CH
0062D5BB	0000	ADD BYTE PTR DS:[EAX],AL

FINISHED with the encryption part.

### Asprotect v2.3 SKE Build 426 Encryption Patch:

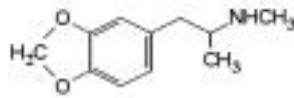
0062D5AF 62 0D CC 9B

b.i>

These 4 bytes above will fix the encryption up to the First VirtualAlloc.







## Synopsis on Asprotect Patching by MaDMA\_n\_H3rCuL3s

Address	Hex dump	Disassembly
0068E000	C705 AFD56200 8000006A	MOV DWORD PTR DS:[62D5AF],6A000080
0068E00A	893D 00E26800	MOV DWORD PTR DS:[68E200],EDI
0068E010	C787 F3F00400 6828E068	MOV DWORD PTR DS:[EDI+4F0F3],68E02868
0068E01A	66:C787 F7F00400 00C3	MOV WORD PTR DS:[EDI+4F0F7],0C300
0068E023	- E9 87F5F9FF	JMP ASProtect.0062D5AF
0068E028	50	PUSH EAX
0068E029	A1 00E26800	MOV EAX,DWORD PTR DS:[68E200]
0068E02E	C780 F3F00400 68008000	MOV DWORD PTR DS:[EAX+4F0F3],800068
0068E038	66:C780 F7F00400 006A	MOV WORD PTR DS:[EAX+4F0F7],6A00
0068E041	C780 C1F50400 6865E068	MOV DWORD PTR DS:[EAX+4F5C1],68E06568
0068E04B	66:C780 C5F50400 00C3	MOV WORD PTR DS:[EAX+4F5C5],0C300
0068E054	05 F3F00400	ADD EAX,4F0F3
0068E059	A3 50E26800	MOV DWORD PTR DS:[68E250],EAX
0068E05E	58	POP EAX
0068E05F	- FF25 50E26800	JMP DWORD PTR DS:[68E250]
0068E065	90	NOP
0068E066	90	NOP

This is the code up till the POPAD.

Now we must search for my “Generic” patch approach.

So first thing we do is find what I dub the “Homebase” code. I originally posted this on the ARTeam forum in December of last year, and it has since changed quite a bit. Now my “HomeBase” is now two homebases. There was one instance of the searched code, but now in the latest build we have 2 instances. This is what we will use as our test and also, at this part in code, most of our other code is decrypted in memory, so we can patch anything. So now go into the base address that EDI held at the first VirtualAlloc redirection and then search for this binary string:

HOMEBASE:	
60 89 E0 9C 5A 55 89 E5	

Address	Hex dump	Disassembly
0072303C	60	PUSHAD
0072303D	89E0	MOV EAX,ESP
0072303F	9C	PUSHFD
00723040	5A	POP EDX
00723041	55	PUSH EBP
00723042	89E5	MOV EBP,ESP
00723044	83C5 24	ADD EBP,24
00723047	31C9	XOR ECX,ECX
00723049	64:8B09	MOV ECX,DWORD PTR FS:[ECX]
0072304C	81EC B80B0000	SUB ESP,0BB8
00723052	FF75 08	PUSH DWORD PTR SS:[EBP+8]
00723055	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
00723058	52	PUSH EDX
00723059	51	PUSH ECX
0072305A	50	PUSH EAX
0072305B	FF75 04	PUSH DWORD PTR SS:[EBP+4]
0072305E	E8 F5FEFFFF	CALL 00722F58
00723063	81C4 DC0B0000	ADD ESP,0BDC
00723069	C2 0C00	RETN 0C

Should be here.

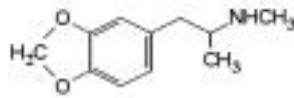
This is the first found instance which will work for us and our purposes here. Now we can easily find the CRC check from this point. One way is to follow this CALL here before the RETN 0C, following it brings you to this place:

Address	Hex dump	Disassembly
00722F58	55	PUSH EBP
00722F59	8BEC	MOV EBP,ESP
00722F5B	81C4 40FFFFFF	ADD ESP,-0C0
00722F61	53	PUSH EBX
00722F62	56	PUSH ESI
00722F63	57	PUSH EDI
00722F64	8B5D 0C	MOV EBX,DWORD PTR SS:[EBP+C]
00722F67	837D 18 00	CMP DWORD PTR SS:[EBP+18],0
00722F6A	74 08	IF SHORT 00722F75

Now scroll down a bit to see this call:







## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s

00723005	68 34307200	PUSH 723034	ASCII "196",CR,LF
0072300A	E8 4127FEFF	CALL 00705750	
0072300F	E8 B001FDFF	CALL 006F31C4	
00723014	8D95 43FFFFFF	LEA EDX,DWORD PTR SS:[EBP-BD]	
0072301A	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
0072301D	E8 8AFDFFFF	CALL 00722DAC	
00723022	5F	POP EDI	006F0000
00723023	5E	POP ESI	006F0000
00723024	5B	POP EBX	006F0000
00723025	8BE5	MOV ESP,EBP	
00723027	5D	POP EBP	006F0000
00723028	C2 1800	RET 18	

Now enter this call:

Address	Hex dump	Disassembly
00722DAC	55	PUSH EBP
00722DAD	8BEC	MOV EBP,ESP
00722DAF	83C4 F4	ADD ESP,-0C
00722DB2	53	PUSH EBX
00722DB3	8B42 30	MOV EAX,DWORD PTR DS:[EDX+30]
00722DB6	83E8 20	SUB EAX,20
00722DB9	83E8 04	SUB EAX,4

Now again scroll down to reveal our place.

00722DFF	FF75 F8	PUSH DWORD PTR SS:[EBP-8]
00722E02	9D	POPFD
00722E03	8B65 F4	MOV ESP,DWORD PTR SS:[EBP-C]
00722E06	61	POPAD
00722E07	C3	RET
00722E08	5B	POP EBX
00722E09	8BE5	MOV ESP,EBP
00722E0B	5D	POP EBP
00722E0C	C3	RET

And there it is.

I know it isn't much to look at, but believe me, this is it. When the POPAD is executed, the registers will tell you the story, as you will see below this. So now let's redirect from the POPAD before this, and have it write to the PUSHAD we just found a second ago. So we will redirect from the PUSHAD, and then do another search for the next set of code. We search for PUSH 0 (68 00 00 00 00 68)

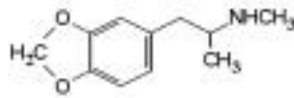
Binary Search:
68 00 00 00 00 68

Address	Hex dump	Disassembly
0072C334	68 00000000	PUSH 0
0072C339	68 34C37200	PUSH 72C334
0072C33E	68 9C07E100	PUSH 0E1079C
0072C343	E8 F46CFFFF	CALL 0072303C
0072C348	4F 00000000	CALL 0072C334

Here it is...

Almost all of the time you will find a lot of these, the best way to figure out which one is the right one, is to simply find the one that isn't normal code, this one as you have now seen is





## Synopsis on Asprotect Patching by MaDMan\_H3rCuL3s

obfuscated. Or if you feel lazy (which I do almost every hour of the day) set a BP on every instance. (NOTE: in older versions, possibly v2.2 I believe it was the eighth instance of it). If you ever become consumed by a feeling of “loser-ness” just set a BP like a mention, if EAX = your executable (as seen in the below picture) then you got the right one.

Now look in EAX.

```
EAX 0012FA1D ASCII " :\\Program Files\\ASProtect SKE 2.3 Beta426\\ASProtect_patch.exe"
ECX 01730000
EDX 000FAD3A
EBX 000B68FF
ESP 0012F9F8
EBP 0012FB44
ESI 0007031C
EDI 5B49E465
EIP 0072C334
```

Renamed of course.

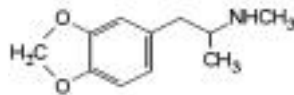
Now all you wimps can actually stop here. This is the filename check, so you can actually just from this point, have it point to your .BAK file. Since when you usually apply a patch there is that option (make backup) if you know the backup name, then you can just use that. But for all you hardcore reversers, let's move on. We now know we need to redirect from this point. I know there is two actual searched binary searched pieces of code. (The PUSHAD). What we were mainly concerned about was getting to this point. So in our patched area we will add this offset to it. But we now have everything we need to actually patch to the CRC check. We found Homepage, and the backup push (the backup meaning the PUSH 0, where we can make a .bak file instead). So our direction of movement will be:

1. Homepage.
2. Backup PUSH (look above)
3. Homepage (create a test loop)
4. Patch CRC

So we have patch to the first Homepage, then the Backup PUSH, now we go back to homepage, and create our loop. To do this we must understand what's going on, so we have valid data to compare to. The choice really is yours whether to loop it, or just patch and reverse it. By this I mean (e.g. Patch: 00401050, move up 6 bytes, 0040104A, move up 6 bytes, 00401046... etc)

Instead we will loop it, so we can minimize our patch code, and learn more things. So we know the CRC is at that POPAD. If you actually count how many times the POPAD executes before the actual CRC check, it's 4 times. So we do this, Hit Homepage, increment our counter, hit homepage, inc our counter, etc... till our counter equals 4, then we patch it. Sounds simple? Let's see ☺





## Synopsis on Asprotect Patching by MaDMAn\_H3rCuL3s

0068E116	- FF25 50E26800	JMP DWORD PTR DS:[68E250]	START HOMEBASE
0068E11C	60	PUSHAD	EAX = ImageBase
0068E11D	A1 00E26800	MOV EAX,DWORD PTR DS:[68E200]	Increment the Counter
0068E122	FE05 60E26800	INC BYTE PTR DS:[68E260]	Move the counter to CL
0068E128	8A0D 60E26800	MOV CL,BYTE PTR DS:[68E260]	IS it the 4th time?
0068E12E	80F9 04	CMP CL,4	Jmp if it is
0068E131	74 17	JE SHORT ASProtect.0068E14A	Return back
0068E133	05 42300300	ADD EAX,33042	kernel32.MapViewOfFile
0068E138	A3 50E26800	MOV DWORD PTR DS:[68E250],EAX	
0068E13D	61	POPAD	
0068E13E	60	PUSHAD	
0068E13F	89E0	MOV EAX,ESP	
0068E141	9C	PUSHFD	
0068E142	5A	POP EDX	
0068E143	55	PUSH EBP	
0068E144	- FF25 50E26800	JMP DWORD PTR DS:[68E250]	JMP BACK
0068E14A	C780 3C300300 6089E09C	MOV DWORD PTR DS:[EAX+3303C],9CE08960	
0068E154	66:C780 40300300 5A55	MOV WORD PTR DS:[EAX+33040],555A	
0068E15D	C780 072E0300 6881E168	MOV DWORD PTR DS:[EAX+32E07],68E18168	
0068E167	66:C780 0B2E0300 00C3	MOV WORD PTR DS:[EAX+32E0B],0C300	
0068E170	05 3C300300	ADD EAX,3303C	
0068E175	A3 50E26800	MOV DWORD PTR DS:[68E250],EAX	kernel32.MapViewOfFile
0068E17A	61	POPAD	
0068E17B	- FF25 50E26800	JMP DWORD PTR DS:[68E250]	
0068E181	90	NOP	
0068E182	90	NOP	

We see what we need to do. (NOTE: it's a good idea to set the counter to 00, since its default is 90)

### Check EAX

```
EAX 7C80B780 kernel32.MapViewOfFile
ECX 0012F9C4
EDX 7C90EB94 ntdll.KiFastSystemCallRet
EBX 00000076
```

EAX = OUR API ☺

### Check the Stack

Address	Value	Comment
0012F9C0	7C80B780	kernel32.MapViewOfFile
0012F9C4	01750000	
0012F9C8	00000074	
0012F9CC	00000004	
0012F9D0	00000000	

Let's make it easier to read.

Address	Value	Comment
EBP-34	7C80B780	kernel32.MapViewOfFile
EBP-30	01750000	
EBP-2C	00000074	
EBP-28	00000004	
EBP-24	00000000	
EBP-20	00000000	

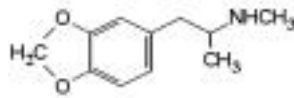
So [EBP-28] is the magic number here.

We see that we must patch EBP-28 from 04 to 01 in order to be able to fix the CRC check. So in our redirected space we must patch this to 01, then again redirect to our homebase.

0068E17B	- FF25 50E26800	JMP DWORD PTR DS:[68E250]	
0068E181	C645 D8 01	MOV BYTE PTR SS:[EBP-28],1	FIX CRC CHECK
0068E185	90	NOP	

There, now let's add the homebase code.





## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s

0068E17B	- FF25 50E26800	JMP DWORD PTR DS:[68E250]	
0068E17C	C645 D8 01	MOV BYTE PTR SS:[EBP-20],1	FIX CRC CHECK
0068E185	60	PUSHAD	
0068E186	A1 00E26800	MOV EAX,DWORD PTR DS:[68E200]	
0068E18B	C780 072E0300 C35B8BE5	MOV DWORD PTR DS:[EAX+32E07],E58B5BC3	---- Restore Code
0068E195	66:C780 0B2E0300 5DC3	MOV WORD PTR DS:[EAX+32E0B],0C35D	---- Restore Code
0068E19E	C780 3C300300 68C2E168	MOV DWORD PTR DS:[EAX+3303C],68E1C268	---- HomeBase Code
0068E1A8	66:C780 40300300 00C3	MOV WORD PTR DS:[EAX+33040],0C300	---- HomeBase Code
0068E1B1	05 072E0300	ADD EAX,32E07	
0068E1B6	A3 50E26800	MOV DWORD PTR DS:[68E250],EAX	
0068E1BB	61	POPAD	
0068E1BC	- FF25 50E26800	JMP DWORD PTR DS:[68E250]	
0068E1C2	90	NOP	
0068E1C3	90	NOP	

Now look in EAX:

```
EAX 01760000 ASCII "MZP"
ECX 0012F968
```

EAX = OUR MAPPED IMAGE ☺

So now like last part of this tutorial, we will patch the bytes we used to get this far. We have 4 bytes for our VirtualAlloc Patch, then one byte in the header for our size addition. So we have 5 bytes to fix, then restore the code for our HomeBase, then we have a running executable.

0068E1BC	- FF25 50E26800	JMP DWORD PTR DS:[68E250]
0068E1C2	C680 99030000 00	MOV BYTE PTR DS:[EAX+399],0
0068E1C9	90	NOP

Header FIX.

VirtualAlloc fix:		
0062D5AF	C2 36 CB 56	Ä6ËV
BINARY: C2 36 CB 56		

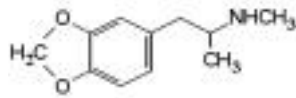
So now we fix it like so:

0068E1C9	C780 AF6D0D00 C236CB56	MOV DWORD PTR DS:[EAX+D6DAF],56CB36C2
0068E1D3	60	PUSHAD

And that's the dword we used to get past most of the encryption.

Then add the code we used to get here back, and we save all changes, and then run it....





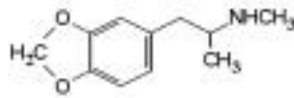
## Synopsis on Asprotect Patching by MaDMAN\_H3rCuL3s



And it RUNS!

Overall patched code:			
0068E000	C705 AFD56200 8000006A	MOV DWORD PTR DS:[62D5AF],6A000080	
0068E00A	893D 00E26800	MOV DWORD PTR DS:[68E200],EDI	
0068E010	C787 F3F00400 6828E068	MOV DWORD PTR DS:[EDI+4F0F3],68E02868	
0068E01A	66:C787 F7F00400 00C3	MOV WORD PTR DS:[EDI+4F0F7],0C300	
0068E023	- E9 85F5F9FF	JMP ASProtect.0062D5AD	
0068E028	50	PUSH EAX	
0068E029	A1 00E26800	MOV EAX,DWORD PTR DS:[68E200]	
0068E02E	C780 F3F00400 68008000	MOV DWORD PTR DS:[EAX+4F0F3],800068	
0068E038	66:C780 F7F00400 006A	MOV WORD PTR DS:[EAX+4F0F7],6A00	
0068E041	C780 C1F50400 6865E068	MOV DWORD PTR DS:[EAX+4F5C1],68E06568	
0068E04B	66:C780 C5F50400 00C3	MOV WORD PTR DS:[EAX+4F5C5],0C300	
0068E054	05 F3F00400	ADD EAX,4F0F3	
0068E059	A3 50E26800	MOV DWORD PTR DS:[68E250],EAX	
0068E05E	58	POP EAX	
0068E05F	- FF25 50E26800	JMP DWORD PTR DS:[68E250]	
0068E065	50	PUSH EAX	
0068E066	A1 00E26800	MOV EAX,DWORD PTR DS:[68E200]	
0068E06B	C780 C1F50400 617508B8	MOV DWORD PTR DS:[EAX+4F5C1],B8087561	
0068E075	66:C780 C5F50400 0100	MOV WORD PTR DS:[EAX+4F5C5],1	
0068E07E	C780 3C300300 68A2E068	MOV DWORD PTR DS:[EAX+3303C],68E0A268	
0068E088	66:C780 40300300 00C3	MOV WORD PTR DS:[EAX+33040],0C300	
0068E091	05 C1F50400	ADD EAX,4F5C1	
0068E096	A3 50E26800	MOV DWORD PTR DS:[68E250],EAX	

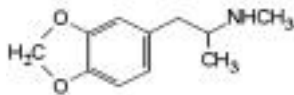




## Synopsis on Asprotect Patching by MaDMan\_H3rCuL3s

0068E09B	58	POP EAX
0068E09C	- FF25 50E26800	JMP DWORD PTR DS:[68E250]
0068E0A2	50	PUSH EAX
0068E0A3	A1 00E26800	MOV EAX,DWORD PTR DS:[68E200]
0068E0A8	C780 3C300300 6089E09C	MOV DWORD PTR DS:[EAX+3303C],9CE08960
0068E0B2	66:C780 40300300 5A55	MOV WORD PTR DS:[EAX+33040],555A
0068E0BB	C780 34C30300 68DFE068	MOV DWORD PTR DS:[EAX+3C334],68E0DF68
0068E0C5	66:C780 38C30300 00C3	MOV WORD PTR DS:[EAX+3C338],0C300
0068E0CE	05 3C300300	ADD EAX,3303C
0068E0D3	A3 50E26800	MOV DWORD PTR DS:[68E250],EAX
0068E0D8	58	POP EAX
0068E0D9	- FF25 50E26800	JMP DWORD PTR DS:[68E250]
0068E0DF	50	PUSH EAX
0068E0E0	A1 00E26800	MOV EAX,DWORD PTR DS:[68E200]
0068E0E5	C780 34C30300 68000000	MOV DWORD PTR DS:[EAX+3C334],68
0068E0EF	66:C780 38C30300 0068	MOV WORD PTR DS:[EAX+3C338],6800
0068E0F8	C780 3C300300 681CE168	MOV DWORD PTR DS:[EAX+3303C],68E11C68
0068E102	66:C780 40300300 00C3	MOV WORD PTR DS:[EAX+33040],0C300
0068E10B	05 34C30300	ADD EAX,3C334
0068E110	A3 50E26800	MOV DWORD PTR DS:[68E250],EAX
0068E115	58	POP EAX
0068E116	- FF25 50E26800	JMP DWORD PTR DS:[68E250]
0068E11C	60	PUSHAD ; START HOMEBASE
0068E11D	A1 00E26800	MOV EAX,DWORD PTR DS:[68E200] ; EAX = ImageBase
0068E122	FE05 60E26800	INC BYTE PTR DS:[68E260] ; Increment the Counter
0068E128	8A0D 60E26800	MOV CL,BYTE PTR DS:[68E260] ; Move the counter to CL
0068E12E	80F9 04	CMP CL,4 ; IS it the 4th time?
0068E131	74 17	JE SHORT ASProtec.0068E14A ; Jmp if it is
0068E133	05 42300300	ADD EAX,33042 ; Return back
0068E138	A3 50E26800	MOV DWORD PTR DS:[68E250],EAX
0068E13D	61	POPAD
0068E13E	60	PUSHAD ;  ----- missing code
0068E13F	89E0	MOV EAX,ESP ;
0068E141	9C	PUSHFD ;
0068E142	5A	POP EDX ;
0068E143	55	PUSH EBP ;  -----
0068E144	- FF25 50E26800	JMP DWORD PTR DS:[68E250] ; JMP BACK
0068E14A	C780 3C300300 6089E09C	MOV DWORD PTR DS:[EAX+3303C],9CE08960
0068E154	66:C780 40300300 5A55	MOV WORD PTR DS:[EAX+33040],555A
0068E15D	C780 072E0300 6881E168	MOV DWORD PTR DS:[EAX+32E07],68E18168
0068E167	66:C780 0B2E0300 00C3	MOV WORD PTR DS:[EAX+32E0B],0C300
0068E170	05 3C300300	ADD EAX,3303C
0068E175	A3 50E26800	MOV DWORD PTR DS:[68E250],EAX
0068E17A	61	POPAD
0068E17B	- FF25 50E26800	JMP DWORD PTR DS:[68E250]
0068E181	C645 D8 01	MOV BYTE PTR SS:[EBP-28],1 ; FIX CRC CHECK
0068E185	60	PUSHAD
0068E186	A1 00E26800	MOV EAX,DWORD PTR DS:[68E200]
0068E18B	C780 072E0300 C35B8BE5	MOV DWORD PTR DS:[EAX+32E07],E58B5BC3;  --- Restore Code
0068E195	66:C780 0B2E0300 5DC3	MOV WORD PTR DS:[EAX+32E0B],0C35D ;  --- Restore Code
0068E19E	C780 3C300300 68C2E168	MOV DWORD PTR DS:[EAX+3303C],68E1C268  --- HomeBase Code
0068E1A8	66:C780 40300300 00C3	MOV WORD PTR DS:[EAX+33040],0C300 ;  --- HomeBase Code
0068E1B1	05 072E0300	ADD EAX,32E07
0068E1B6	A3 50E26800	MOV DWORD PTR DS:[68E250],EAX
0068E1BB	61	POPAD
0068E1BC	- FF25 50E26800	JMP DWORD PTR DS:[68E250]
0068E1C2	C680 99030000 00	MOV BYTE PTR DS:[EAX+399],0
0068E1C9	C780 AF6D0D00 C236CB56	MOV DWORD PTR DS:[EAX+D6DAF],56CB36C2
0068E1D3	60	PUSHAD
0068E1D4	A1 00E26800	MOV EAX,DWORD PTR DS:[68E200]
0068E1D9	C780 3C300300 6089E09C	MOV DWORD PTR DS:[EAX+3303C],9CE08960
0068E1E3	66:C780 40300300 5A55	MOV WORD PTR DS:[EAX+33040],555A
0068E1EC	05 3C300300	ADD EAX,3303C
0068E1F1	A3 50E26800	MOV DWORD PTR DS:[68E250],EAX
0068E1F6	61	POPAD
0068E1F7	- FF25 50E26800	JMP DWORD PTR DS:[68E250]





## Synopsis on Asprotect Patching by MaDMan\_H3rCuL3s

BINARY:																																																																																																
C7	05	AF	D5	62	00	80	00	00	6A	89	3D	00	E2	68	00	C7	87	F3	F0	04	00	68	28	E0	68	66	C7	87	F7	F0	04	00	00	C3	E9	85	F5	F9	FF	50	A1	00	E2	68	00	C7	80	F3	F0	04	00	68	00	80	00	66	C7	80	F7	F0	04	00	00																																	
6A	C7	80	C1	F5	04	00	68	65	E0	68	66	C7	80	C5	F5	04	00	00	C3	05	F3	F0	04	00	A3	50	E2	68	00	58	FF	25	50	E2	68	00	50	A1	00	E2	68	00	C7	80	C1	F5	04	00	61	75	08	B8	66	C7	80	C5	F5	04	00	01	00	C7	80																																	
25	50	E2	68	00	50	A1	00	E2	68	00	C7	80	C1	F5	04	00	61	75	08	B8	66	C7	80	C5	F5	04	00	01	00	C7	80	3C	30	03	00	60	89	E0	9C	66	C7	80	40	30	03	00	5A	55	C7	80	34	C3	03	00	68	DF	E0	68	66	C7	80	38	C3	03	00	00	C3	05	3C	30	03	00	A3	50	E2	68	00	58	FF	25	50	E2	68	00	50											
3C	30	03	00	68	A2	E0	68	66	C7	80	40	30	03	00	00	C3	05	C1	F5	04	00	A3	50	E2	68	00	58	FF	25	50	E2	68	00	50	A1	00	E2	68	00	C7	80	34	C3	03	00	68	00	00	00	66	C7	80	38	C3	03	00	00	68	C7	80	3C	30	03	00	68	1C																														
68	00	50	A1	00	E2	68	00	C7	80	3C	30	03	00	60	89	E0	9C	66	C7	80	40	30	03	00	5A	55	C7	80	34	C3	03	00	68	DF	E0	68	66	C7	80	38	C3	03	00	00	C3	05	3C	30	03	00	A3	50	E2	68	00	58	FF	25	50	E2	68	00	50																																	
00	68	DF	E0	68	66	C7	80	38	C3	03	00	00	C3	05	3C	30	03	00	A3	50	E2	68	00	58	FF	25	50	E2	68	00	60	A1	00	E2	68	00	FE	05	60	E2	68	00	8A	0D	60	E2	68	00	80	F9	04	74	17	05	42	30	03	00	A3	50	E2	68	00	61	60	89																														
A1	00	E2	68	00	C7	80	34	C3	03	00	68	00	00	00	66	C7	80	38	C3	03	00	00	68	C7	80	3C	30	03	00	68	1C	E1	68	66	C7	80	40	30	03	00	00	C3	05	34	C3	03	00	A3	50	E2	68	00	58	FF	25	50	E2	68	00	60	A1	00	E2	68	00	FE	05	60	E2	68	00	8A	0D	60	E2	68	00	80	F9	04	74	17	05	42	30	03	00	A3	50	E2	68	00	61	60	89	
E1	68	66	C7	80	40	30	03	00	00	C3	05	34	C3	03	00	A3	50	E2	68	00	58	FF	25	50	E2	68	00	60	A1	00	E2	68	00	FE	05	60	E2	68	00	8A	0D	60	E2	68	00	80	F9	04	74	17	05	42	30	03	00	A3	50	E2	68	00	61	60	89																																	
68	00	FE	05	60	E2	68	00	8A	0D	60	E2	68	00	80	F9	04	74	17	05	42	30	03	00	A3	50	E2	68	00	61	60	89	E0	9C	5A	55	FF	25	50	E2	68	00	C7	80	3C	30	03	00	60	89	E0	9C	66	C7	80	40	30	03	00	5A	55	C7	80	07	2E	03	00	68	81	E1	68	66	C7	80	0B	2E	03	00	00	C3	05	3C	30	03	00	A3	50	E2	68	00	61	FF	25	50	E2	68	00
2E	03	00	68	81	E1	68	66	C7	80	0B	2E	03	00	00	C3	05	3C	30	03	00	A3	50	E2	68	00	61	FF	25	50	E2	68	00	C6	45	D8	01	60	A1	00	E2	68	00	C7	80	07	2E	03	00	C3	5B	8B	E5	66	C7	80	0B	2E	03	00	5D	C3	C7	80																																	
00	C6	45	D8	01	60	A1	00	E2	68	00	C7	80	07	2E	03	00	C3	5B	8B	E5	66	C7	80	0B	2E	03	00	5D	C3	C7	80	3C	30	03	00	68	C2	E1	68	66	C7	80	40	30	03	00	00	C3	05	07	2E	03	00	A3	50	E2	68	00	61	FF	25	50	E2	68	00																															
3C	30	03	00	68	C2	E1	68	66	C7	80	40	30	03	00	00	C3	05	07	2E	03	00	A3	50	E2	68	00	61	FF	25	50	E2	68	00	C6	80	99	03	00	00	00	C7	80	AF	6D	0D	00	C2	36	CB	56	60	A1	00	E2	68	00	C7	80	3C	30	03	00	60																																	
68	00	C6	80	99	03	00	00	00	C7	80	AF	6D	0D	00	C2	36	CB	56	60	A1	00	E2	68	00	C7	80	3C	30	03	00	60	89	E0	9C	66	C7	80	40	30	03	00	5A	55	05	3C	30	03	00	A3	50	E2	68	00	61	FF	25	50	E2	68	00																																				
89	E0	9C	66	C7	80	40	30	03	00	5A	55	05	3C	30	03	00	A3	50	E2	68	00	61	FF	25	50	E2	68	00																																																																				

If everyone asks nicely I just might do another tutorial on the latest Asprotect (since I see now that 426 is not latest anymore).

## 4. References

- [1] "Asprotect 2.11 Patching by JohnWho", JohnWho , <http://forum.accessroot.com>

## 5. Conclusions

**Don't use these concepts for making illegal operation, all the info here reported are only meant for studying and to help having a better knowledge of application code security techniques.**

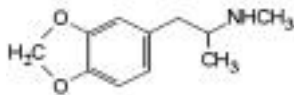
## 6. History

- Version 1.0 – First public release!

## 7. Greetings

Greetings go to just about anyone who is actively contributing to the knowledge scene, If you or anyone you know reading this has any sort of Asprotect keys they are willing to share (personal) this would be great and help continue my study into Asprotect. I want to say what's up to the following people:





### Synopsis on Asprotect Patching by MaDMan\_H3rCuL3s

ARTeam, fly, shoooo, heXer, unpack.cn, PEdiy forum, SECTiON-8, Like maybe one or two 0day groups, Anyone who has done any sort of chemical brain enhancement ( ☺ ), Anyone who makes their own chemical brain enhancers, especially the old HiVE dwellers (you know who you are), and of course.... YOU!



<http://cracking.accessroot.com>

