Unpacking ASProtect v of 2.xx (cutting sections, the restoration of the [skramblernogo] code, [dekompiljatsija] VM, the restoration of import, [inlajn] of [patch])

Victim: FontExpert of 2005 Version: 7.0 Release 1. To take is possible here: <u>the http://www.proximasoftware.com/</u> If on the site there will be new version, and by you will be necessary precisely this - you will report to me, and I will lay out somewhere for the running off.

Author:PE_Kill e the mail:PE_Kill@mail.ru

Introduction

As occasion for writing of this article served the complete absence of information, about the restoration of the [skramblernogo] code and [dekompiljatsii] "virtual machine". I began to write article, simultaneously becoming acquainted with work and equipment Of aSProtect'a. Therefore after writing of article I much rethought, saw many errors and learned considerably more than he wrote in this article. This article more greatly resembles not the management on the removal Of aSProtect'[a], but the diary of the cracker, in which in the order it is said, as I broke concrete program.

Here there are no some, very necessary scripts. For example script for the passage to OEP. This is done intended, since most likely this is last article with more or less complex [iskhodnikami]. To whom will be wanted to work unknown how much at [dekompiljatorom], to and then lay out him to [pablik], so that in following [bilde] all this would cover? It is necessary to be investigated with all itself. I having been, for example, steamed, twentieth time hands to reach OEP, became to study OllyScript and to search for into ASProtect'e of regularity. Already in a day other day I wrote OEP Of finder. So that there is nothing complex of - dare, investigate!

Note: Most likely the addresses of the chosen sections of memory coincide will not be! Even in me on the middle of article was changed the address of the [skramblernogo] code. In this program there is no table of initialization; therefore in this article nothing it it is discussed. The table of initialization exists only in the programs, written to **Delphi**, although it can be, also, to Borland Of c++. Information on the restoration of the table of initialization you can take from the article **Of sergSh** "unpacking Of aSProtect 2.13 based on the example Of icolover.exe", which lies at the division **of rar- article** on the site www.cracklab.ru.

Necessary tools:

OllyDbg 1.10, with [plaginami], for the concealment of the presence of diagnostic routine.

OllyScript by Of sHaG v 0.92 or ODbgScript by Of epsylon3 v 1.41 (better it) PETools by Of nEOx of v1.5 RC6 of - is compulsory of this version! WinHex or another hexadecimal editor. ResFixer v of 1.0 beta 1 by seeQ of - or another utility for [rebilda] of the resources PEiD of v0.93 and v0.94 ImpREC 1.6 for restoring the import. Necessary knowledge: Knowledge PE Of format of - is compulsory Knowledge of the assembler of - superficially Necessary habits: It is necessary to have at least initial habits of unpacking.

Determination of version and the search for original entrance point.

 <i>, , ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,</i>						
🥵 PEiD v0.94		×				
File: C:\PROGRA~1\	ONTEX~1\FONTEX~1.EXE					
Entrypoint: 0000100	GenOEP X tion:	>				
File Offset: 0000100	Found OEP: 0048C197	>				
Linker Info: 7,10	tem: Win32 GUI	>				
	ОК					
ASProtect 2.1× SKE -> Alexey Sologovnikov						
Multi Scan Task Viewer Options About Exit						
🔲 Stay on top		->				

And so, here, which tells us PEiD of v0.94

I will say immediately according to the experience that version 0.94 equally defines AsProtect 2.0x, AsProtect of 2.1xxx and AsProtect 2.2 as AsProtect 2.1x SKE.

But here version 0.93 of these 3[kh] of versions can recognize only AsProtect 2.0x, about the remaining versions he speaks Nothing of found *.

Let us look, which to us will say PEiD of v0.93

艦 PEiD v0.93							
File: c:\PROGRA~1\FONTE	~1\FONTEX~1.EXE						
Entrypoint: 00001000	EP Section:						
File Offset: 00001000	First Bytes: 68,01,40,69						
Linker Info: 7,10	Subsystem: Win32 GUI >						
Nothing found *							
Multi Scan Task Viewer Options About Exit							
Stay on top	>> ->						

It means AsProtect 2.0x it drops off. How to determine, with which of the remained two versions we do deal? This we will explain only further, and we thus far load program into OllyDbg and we see the standard beginning Of asProtect'a of any version (except the early).

00401000	68 01406900	PUSH FONTEX"1.00694001	
00401005	E8 0100000	CALL FONTEX"1.0040100B	
0040100A	C3	RETN	
0040100B	Č3	RETN	
0040100C	93	XCHG EAX.EBX	
0040100D	336A 28	XOR EBP.DWORD PTR DS:[EDX+28]	
00401010	9E	SAHF	
00401011	ŚD	POP EBP	
00401012	93	XCHG EAX.EBX	
00401013 -	- 7F E5	JG SHORT FONTEX"1.00400FFA	
00401015	69BC17 07F2080	IMUL EDI, DWORD PTR DS: [EDI+EDX+908F207]	
00401020	129F DC982A65	ADC BL,BYTE PTR DS:[EDI+652A98DC]	
00401026	E7 86	OUT 86,EAX	I/O command
00401028	B8 76346193	MOV EAX,93613476	
0040102D	7B 49	JPO SHORT FONTEX"1.00401078	
0040102F \	/ 7D 79	JGE SHORT FONTEX"1.004010AA	
00401031	6D	INS DWORD PTR ES:[EDI],DX	I/O command
00401032	7B 0B	JPO SHORT FONTEX"1.0040103F	

Now let us open the options of diagnostic routine and will remove all [galochki] in the supplementary sheet Of exceptions.

Debugging options							
Commands Disasm CPU Registers Stack Analysis 1 Analysis 2 Analysis 3							
Security Debug Events Exceptions Trace SFX Strings Addresses							
Ignore memory access violations in KERNEL32							
Ignore (pass to program) following exceptions:							
INT3 breaks							
🔲 Single-step break							
Memory access violation							
Integer division by 0							
Invalid or privileged instruction							
All FPU exceptions							
Ignore also following custom exceptions or ranges:							
Add last exception							
Add range							
TOK Undo Cancel							

"Why?" - you will ask. Protector with the work generates exceptions, for the difficulty of its fixing. I do not know, can, when it only thought, this it interfere withd coma that fixing its code, but today such Of antiTracing can frighten perhaps that entirely green novice. Well that zh, who harms us, that us will help. Usually the number of exceptions from the moment of the load of program into the diagnostic routine and before complete unpacking one and also (although there are the exceptions (in the sense there is the unequal number of exceptions)). After last exception occurs the even more insignificant correction of data, and then leap to the original entrance point, i.e., in the place, from which the program began to be performed to the protection by protector. This point is our purpose. Costing on the address of original entrance point into the program we let us be able to throw out the unpacked dump of program from the memory to the disk.

And so	we s	tart	our	prograi	n (1	F9).	They	stop	ped i	n this	place	
00C4B530	0 C700) 7FECC	512 M 54261 S	OV DWORD PTP	C DS: 1	EAX],	1205E07F 1#4+65421	16031 04				
00C4B53D	2006	ро ног)	P420.5	ND DH,AL	33:10	LDFTEO	1#4703421	16431,44				
00C4B53F	67:6	4:8F06	000 P	OP EDI OP DWORD PTF	FS:I	[0]						
00C4B546 00C4B549	5 83C4 9 3344	04	A A	DD ESP,4 OR FAX.DWORD		SSIF	SP+281					
00C4B54E		79	Ř	CR EAX,79		00.02	0, .201		Shift o	constant ou	t of range	131
Below i	n th	e li	ne o	f state	we	see	:		-			

Access violation when writing to [00000000] - use Shift+F7/F8/F9 to pass exception to program

OllyDbg reports that occurred the exception with a attempt at the writing into the memory with address 00000000 and proposes to harvest Shift+F7/F8/F9 in order to transmit this exception into

the program for the subsequent working. This is those exceptions themselves, which are necessary to us. Now it is possible to more precisely determine the version of protector. In the version Of asProtect 2.2, as far as to me known, generally there are no exceptions. Apparently the author introduced large changes in charger. Therefore AsProtect 2.2 drops off. In order to reach the original entrance point necessary to press Shift+F9 to that moment, when program is neglected. If after sequential exception the window appears:



, then it is necessary to place what or of [plagin] for OllyDbg the hiding diagnostic routine from the detection (IsDebugPresent, Hide Of debugger).

We memorize the number of occurred exceptions and it is reloaded program (Ctrl+F2). We start [pogrammu] (F9), we are interrupted on the exception and harvest Shift+F9 as many times, as they counted exceptions past time minus one (this and understandably, otherwise program again it will be neglected). In me this number is equal to 35. They must stop where that here:

00	C2A76A	C700 1FADD944	MOV DWORD PTR DS:[EAX],44D9AD1F
00	IC2A770	A2 FAFEFD10	MOV BYTE PTR DS:[10FDFÉFA],AL
00	C2A775	AC	LODS BYTE PTR DS:[ESI]
00	IC2A776	67:64:8F06 000	POP DWORD PTR FS:[0]
00	C2H77C	8304 04	HUU ESP,4
00	U2H77F	804451 07	LEH EHX,DWURD PIK DS:LECX+EDX#2+7]
00	C2H703	01 04990300	MOULEON DWORD PTP DS.[C29004]
00	102H104	0000	MOULEON DWORD FTR DS. (CS9004)

But do not entangle! These places: 2. Us are necessary the second.

But how to us to now find original entrance point? Simplest way of - this to place the point of stop in the section of the code. Since after last exception in the section of the code of more than anything it is written, the following turning to it will be when the code, located in this section it will begin be carried out. But it will be carried out it will begin certainly from our original entrance point (further OEP) into the program.

It is discovered the map of memory (Alt+M) and we see:

003F00001	00001000				Priv	RWE	RWE	
00400000	00001000	FONTEX"1		PE header	Imag	R	RWE	
00401000	0015C000	FONTEX"1		code	Imag	R	RWE	
0055D000	00056000	FONTEX"1		data	Imag	R	RWE	
005B3000	00010000	FONTEX"1			Imag	R	RWE	
005C3000	000D1000	FONTEX"1	.rsrc	resources	Imag	R	RWE	
00694000	00039000	FONTEX"1	.data	imports,rel	Imag	R	RWE	
006CD000	00001000	FONTEX"1	.adata		Imag	R	RWE	
006D0000	00006000				Map	RE	RE	
00790000	00002000				Map	RE	RE	
007A0000	00103000				Map	R	R	
aaanaaaal	00000000				Main	D E		

We see that the program is loaded with address 00400000. The first region with address 00400000 with size of 1000 (PE of header) is PE by the title of our victim (read the description PE Of format'a). The following region with address 00401000 with size of 15C000 is the section of the code, it is here to it to us and it

is necessary to place [brejkpoint] on the access to the memory. We place [brejpoint] (further [brjak]):

POTIC] (T ~ T	CIICI							
00400000	00001000	FONTEX"1		PE he	ader	Imag R		RWE	
00401000	0015C000	FONTEX"1		code		T		DUE	1
0055D000	00056000	FONTEX"1		data	Actua	alize			
005B3000	00010000	FONTEX"1							
005C3000	000D1000	FONTEX"1	.rsrc	reso	View	in Disassei	mbler		E
00694000	00039000	FONTEX"1	.data	impo					
006CD000	00001000	FONTEX"1	.adata		Dump) in CPU			
006D0000	00004000				- '				
00790000	00002000				Dump)			
007A0000	00103000								,
008B0000	00071000				Searc	:n			, (
00880000	00001000				Coord	h novt			1
00C30000	00033000				Searc	mext			· · ·
00070000	00004000								
00088000	00004000				Soth	reak-on-a	cocc		E
0000A0000	00028000				Decip	i cak-ui i-ai	10633		1
00CDC000	00004000								
00D70000	00001000				Set n	emory bre	aknoin	tion acce	224
00080000	00001000				Boon	ioniory bro	-mpoint	e on deed	
annaaaal	00001000								

We harvest Shift+F9 and... we are interrupted clearly not on OEP. Well not terribly, simply AsProtect stole in the program the first several bytes of the code and carried out their itself, but in their place it entered zero. It means above that address, where we now are located they must be zero or the command ADD BYTE PTR DS:[EAX], AL of - so [dizassembliruet] zero disassembler. We dolok above and... that after features?

0.0 - 0		00 0 . 0	0.110.000	0110.0	0.2002	200.0
20485966 20485966	55		POP	EDI		
204859BD	58		POP	EBX		
204859BE	Č9		LEAV	E		
004859BF	C3		RETN			
20485900	3D	00100000	CMP	EAX.1000		
20485905	~ 73	ØE	JNB	SHORT FOR	<pre>ITEX 1.004</pre>	I859D5
204859C7	F70)8	NEG	EAX		
204859C9	030	24	ADD	EAX,ESP		
204859CB	830	CØ 04	ADD	EAX,4		
304859CE	859	30	TEST	DWORD P1	FR DS:DEA>	(],EAX
304859D0	94		XCHG	EAX,ESP		
004859D1	8B9	30	MOV	EAX,ĎWORD) PTR DS:[EAX3
204859D3	50		PUSH	EAX		
204859D4	C3		RETN			

Above is located instruction RET, of zero it does not smell. I will say on the secret that this not OEP, but altogether only one of the numerous functions of program. How I did learn, that this not OEP? This will arrive with the experience. Try to look as it appears BY OEP in the programs, written in the different languages of programming and you will understand everything. Then another question asserts itself: "If this not OEP, this first turning to the section of the code and this is one of the numerous functions of program, then from where was caused this function?". This question to answer is not complicated. Once this is function, then with its call the address of recovery will be brought in into the stack. We look, what do we have in the stack:

0012FF20	00E22236
0012FF24	00F502B2
0012FF28	00CDE288
0012FF2C	3D83D8BF
0012FF30	0012FFE0
0012FF34	0012FF6C

We see that the address of the recovery of 00E22236. Council. Never mind, which is located in the third column of the window of stack. Here - nothing, but is - complete absurdity. Now let us pass to this address, we harvest Ctrl+G, enter 00E22236, Enter.[Okazyvaemsja] here:

-		10.0	1010	
	00E2222F	6H UU	PUSH 0	
	00E22231 ^	E9 ACE6FFFF	JMP 00E208E2	
	00E22236	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
	00E22239 V	EB 01	JMP SHORT 00E2223C	
	00E2223B	9A 8D340781 DE	CALL FAR CODE:8107348D	Far call
	00E22242	3D 010A8D74	CMP EAX,748D0A01	
	00E22247	24 03	AND AL.3	
	00E22249	8D76 FD	LEA ESI.DWORD PTR DS:[ESI-3]	
	00E2224C	893E	MOV DWORD PTR DS:[ESI].EDI	
	00E2224E	56	PUSH ESI	
	00E2224F	FF15 8CD25500	CALL DWORD PTR DS:[55D28C]	kernel32.GetVersionExA
	OOFOOTE	0045 40	MOUL FOW DWODD, DTD, DO, SEOT, 103	

We see that to 00E2224F is caused BY API the function Of getVersionExA. We recall, that this function one of the first is caused in the programs, written to Visual C. if this program is written on, then we are located almost on OEP. The result is that program begins they will be carried out not in the section of the code, but in this region to memory and [brjak] to the access to the memory on the last exception must be placed precisely on this region, and not in the section of the code. Let us consider this. In order to determine the beginning of this region we press the key For home and it is exerted on the address of 00E20000, this is a beginning of region. Let us memorize this number and it is reloaded program. We reach the last exception, it is discovered the map of memory and we place [brjak] on the access to the memory to that region itself:



If I do not make mistakes, then by command PUSH 60 does begin the program, compiled by what that (I do not remember) however, with version s. I that it is obtained? But is obtained this is what - Of asProtect now it does not steal several bytes with OEP (to the

first call of call), but takes away completely all beginning of program into itself and memories are carried out in region chosen under this code. Well this is not terrible! What does prevent us from [sdampit] program and separately this region memory, and then tightening this region of in the form new section to the dump? Let us look only below and we see:



We see that the function of 00E90000 it is caused to of 00E203CA and 00E203F1. Moreover in the first case into the stack is placed 00E208B0 (similarly to the address of recovery), but the secondly into the stack generally nothing it is placed. Really this is is one additional trick Of asProtect'a? Let us verify. We harvest by the right button of mouse and we select Search of for->All of commands. In the appeared window we write call of 00E90000 we harvest Enter. The imposing list was obtained:

00E203CA CALL 00E90000	(Initial CPU selection)
00E203F1 CALL 00E90000	
00E205B8 CALL 00E90000	
00E205C2 CALL 00E90000	
00E205ED CALL 00E90000	
00E205F4 CALL 00E90000	
00E20604 CALL 00E90000	
00E20614 CALL 00E90000	
00E20826 CALL 00E90000	
00E208AB CALL 00E90000	
00E208DD CALL 00E90000	
00E208EF CALL 00E90000	
00E208F9 CALL 00E90000	
00E2092A CALL 00E90000	
00E209EA CALL 00E90000	
00E20A27 CALL 00E90000	
00E20A32 CALL 00E90000	
00E20A9F CALL 00E90000	
00E20ABD CALL 00E90000	
00E20ACF CALL 00E90000	
AMEZABSDI CALL AMESAAAA	

In all in me it was located 112 calls of these functions. It means, to [sdampit] this region will not come out O. "but why!" you say: "It is possible to [sdampit] and this function!". But here this will not come out, since this is that "virtual machine itself" (further VM), about which everything they heard, but that it from itself represents and as to fight they do not know with it. To [sdampit] it is possible, but [gimorno] is not-pretty. [Gimorno], because it is already disposed to the current addresses yes and the size in it impressing, and is not-pretty because notpretty... And one additional problem consists in the fact that checking registration also occurs in VM.

[Dekompiljatsija] of virtual machine.

However, what such "virtual machine"? AsProtect with the packing of program moves away from it some instructions and they will replace with their call VM. In this context of understanding - this is VM the region memory, in which are carried out the specific actions. The result of these actions will be the same as with the fulfillment of original instructions. But, in contrast to the original instructions, in VM all is carried out more tangled. Maltsters it tried as it is possible to more strongly hide the mechanism of work VM. As a result with the dump of program remain many calls VM, which now is not. Therefore dump is obtained nonworking. To fight with this is possible by several methods. Best - to restore all stolen instructions, to thus remove dependence on VM. For restoring the instructions there are two methods:

1) to completely study work VM, to understand the operating principles with its tables, to understand the size of utilized data and to as a result write analog, but not for fulfilling the stolen instructions, but for their restoration.

2) to find such places in the nucleus VM, where, after stopping at [brjake], it is possible to accurately determine the type of the emulated instruction and directly its operands. For example, after stopping in similar you [mete], on what that to signs to understand, that this is the instruction of cmp, to neglect program further, to again stop and to understand that the register of eax is compared with which the, and after stopping where the further - to understand with which it is compared.

The first method is completely long, is labor-consuming and in principle no one not necessary. The second more promising and the spent time depends only on the power of observation and on the level of meditation. Certainly, I selected the second method O. as a result I it wrote script for restoring the stolen instructions.

I will not describe in detail as I it searched for each control point for the script, since, although this and more rapid method, not on so many in order for two passages to understand the logic of that, to what Of [soldovnikov] it dedicated so much time. Script is applied to this article (**RebuildVM.osc**) and I tried as it is possible to in more detail describe each line of the code. To whom is not interesting the theory of - that it can simply neglect script and it on the automaton will restore the stolen instructions, well and all rest I please to pass with me into the peace of the captivating step-by-step laying out of the code \bigcirc .

Let us approach! Let us place indicator on the first call VM and will press Ctrl+ * or by the right button of [myshi]->New of origin of here, thus, after establishing location counter (eip) to

the necessary address. Now we harvest F7 (Step of into) and fall into the function of 00E90000. We see the heap of rubbish and instructions of jmp. How I did understand, where the rubbish? Beginning of the function:

00E90000 JMP SHORT of 00E90005 00E90003 INT 20 00E90005 PUSH EDI 00E90006 PUSHFD 00E90007 JMP SHORT of 00E9000C 00E9000A INT 20 00E9000C LEA EDI, DWORD PTR SS:[EBP+ESI+405AC8] 00E90013 SUB EDI, ESI 00E90015 SUB ESP, 20 00E90018 LEA EDI, DWORD PTR DS:[ECX+EDX * 2+5F]

We see to 00E90000 that is carried out the leap through the command INT 20 to 00E90005. Then into the stack is placed register EDI. Remains register EFL (state of all flags). Again the leap through the command INT 20 to 00E9000[S]. in EDI is placed dword with the address EBP+ESI+405AC8. From EDI is read ESI. ESP decreases by 20h. In EDI is placed dword with the address ECX+EDX * 2+5F. Note that for the recording of dword'[a] onto the register EDI to 00E90018 is used not one value of the previous calculations. Result simply is rerecorded! I.e. all this can be replaced with the following code:

PUSH EDI PUSHFD SUB ESP, 20 LEA EDI, DWORD PTR DS:[ECX+EDX * 2+5F]

Even then, if you look further, then you will see, that EDI again is rerecorded, so that the command LEA EDI, DWORD PTR DS:[ECX+EDX * 2+5F] is not also necessary. Now, if we want, then it is possible to completely clean rubbish from this function and to ascertain that the useful code in it entirely a little, in essence of - this calculation of the address of the passage into the following region of memory. Passage into this region is accomplished with the aid of the command CALL EXX, where instead of EXX there can be any register.

Interesting us passage:					
00E9012F	26:EB 02	JMP SHORT 00E90134	Superfluous prefix		
00E90132	- CU 20 908008 3552440	INT 20 LEG EDT DWORD PTR DS+FERV+ECV+44E23E1			
00E9013B	8DBC21 FC84C50	LEA EDI.DWORD PTR DS:[ECX+C584FC]			
00E90142	28F9	SUB EDI,ECX			
00E90144	FFD7	CALL EDI			
00E90146	68 848C4EF9	PUSH F94E8C84			
00E9014B	337C24 28	XOR EDI,DWORD PTR SS:[ESP+28]			
00E9014F	337C24 08	XOR EDI,DWORD PTR SS:[ESP+8]			
00E90153	SF	POP EDI			
00E90154	2078 CC	AND BYTE PTR DS:[EAX-34],BH			
00E90157	00C3	ADD BL,AL			
00E90159	0000	ADD BYTE PTR DS:[EAX],AL			
00E9015B	0000	ADD BYTE PTR DS:[EAX],AL			
00E9015D	0000	ADD BYTE PTR DS:[EAX],AL			
00E9015F	0000	ADD BYTE PTR DS:[EAX],AL			
00E90161	0000	ADD BYTE PTR DS:[EAX],AL			
00E90163	0000	ADD BYTE PTR DS:[EAX],AL			
00E90165	0000	ADD BYTE PTR DS:[EAX],AL			
00E90167	0000	ADD BYTE PTR DS:[EAX].AL			

Upon transfer into the following region of memory we fall into the function, which achieves identification of the stolen instruction, and it is more concrete specific, function sorts out the coded values in its table of hash and compares with hash of the current function, if they coincided, then the primary initialization of function is produced and it passes directly for emulation. This function also not large - of the order of OACh (172) of bytes.



CALL EDX to 00C5856F is occupied by sample and decoding of hash from the table of hash. Immediately the comparison of obtained hash with hash of the current function is produced after it. If they are not equal, then is checked not last this hash in the table. If the latter, then reveals "Error: 111". But here function to 00C5858B will be carried out only if hash it coincided. Here this function is a emulator of the stolen instructions! Let us place [brjak] on this function and will neglect program. They interrupted. We harvest F7 and fall directly into the emulator. This function most tangled, since many different actions here are produced. Only we see below such instructions



CALL EDX obtains the type of the stolen instruction: AL= 0 (it is stolen call) AL= 1 (it is stolen jmp) AL= 2 (it is stolen jcc (one of 16 [dzhampov])) AL= e (they are stolen cmp+jcc) Yes, ASProtect can emulate immediately several instructions. Respectively now we see that here VM branches out also depending on what type emulation, will be produced passage to the appropriate branch of emulation. Let us place [brjak] immediately after call edx.[Zapuskaem] program. They interrupted, we see that in AL 0, means it is emulated call (you remember into the stack it was placed the address of recovery?). Means to us it is necessary first conditional [dzhamp]. We harvest two times F8 and we here.



[Dotrassirujte] to the instruction Of jmp to **00C583F1** and will carry out it. Let us prove to be here.

	<u> </u>	
0C584AB	8855 FC	MOV EDX, DWORD PTR SS:[EBP-4]
0C584AE	83EA 04	SUB EDX,4
0C584B1	8902	MOV DWORD PTR DS:[EDX],EAX
ØC584B3	8B45 08	MOV EAX.DWORD PTR SS:[ÉBP+8]
0C584B6	E8 D5FAFFFF	CALL 00C57F90
0C584BB	FF75 FC	PUSH DWORD PTR SS:[EBP-4]
0C584BE	FF75 10	PUSH DWORD PTR SS:[EBP+10]
0058401	FF75 ØC	PUSH DWORD PTR SS:[EBP+C]
0058404	8845 F8	MOV EAX.DWORD PTR SS:[EBP-8]
0058407	FF60 20	JMP DWORD PTR DS: [EAX+20]
0C584CA .	V EB 01	JMP SHORT 00C584CD

Jmp to 00C584C7 carries out the leap into another region of memory, but there it occurs nothing interesting, only is normalized stack, the address of recovery is placed into the stack and leap to the function of program is carried out. I.e. if in the program to the protection it was as follows:

Call of xxxxxxxx

Ø

The **ASProtect** substitutes this by the call VM, where, if we strongly strongly simplify it is carried out: **Push** the address of the recovery **Jmp** of xxxxxxx

Where instead of xxxxxxx there can be both the address in the section of the code and the address in the chosen region of memory.

In my script I substitute call VM by precisely such instructions. The address, which the instruction **of push** places into the stack necessary to change by the address, with which will be placed the stolen code after dump. With emulation **of call** they were dismantled. That I can say apropos emulation **of jmp**, and nothing. This one and the same, with one reservation alone, that, if it is caused BY VM for emulation **of call**, then before **call VM** goes **push** of xxxxxxx. But if **jmp**, then nothing it is placed into the stack.

Let us return to the function, which determines the type of the stolen instruction, for this sufficient several times to harvest key "-" on the digital keyboard. Let us move indicator for following conditional [dzhamp].



And let us press Enter. Let us prove to be here:

20U583F9	3360	XUK EHX,EHX
AAC583EB	8943 04	MOU AL BYTE PTR DS:[FBX+4]
BOCCODEE	ODEE EO	MOU EDY DWODD DTD CC. [EDD-0]
SOCOOLE!	0000 00	TION EDA, DWORD FIR SSILEDF-01
30C58401	8B5482 40	MOV_EDX.DWORD_PTR_DS:[EDX+EAX*4+40]
200552405	88C6	MOLL FOX FST
300004001	5555	
20US8407	FFU2	UHLL EUX
30C584091	8BD8	IMOV EBX.EAX
aarsedael	8840 10	MOULECX DWORD PTR SS.FERP+101
20000400	0000	MOULEDVEDV
2005840E	8603	NUV EDA, EBA
30C58410	8B45 F8	MOV EAX.DWORD PTR SS:[EBP-8]
20052413	F8 D4FBFFFF	COLL 00C57EEC
200000410	0400	
20C58418	8400	IESI HL,HL
20C5841A	74 17	JE SHORT 00C58433
aac5841c	- FB 01	IMP SHORT 00C5841E
2000004101	· LD 01	

Call of edx to 00C58407 determines the type of the stifling of jcc. After its fulfillment of eax contains the type of the stifling of jcc.

Number of [dzhampa]	[Dzhamp]	[Opkod]
0	Jo	70
1	Jno	71
2	Jb	72
е	Jnb	73
4	Je	74
shch	Jnz	75
1	Jpe	7A
"	Jpo	7B
8	Js	78
9	Jns	79
А	Jbe	76
В	Ja	77
С	Jl	7C
D	Jge	7D
Е	Jle	7E
F	Jg	7F

As you see entire **jcc** they go practically on the growth, with exception of two.

But **call** to **00C58413** performs complex logical operations, leading the values of conditional flags to one - 0 or 1, and is placed it in **ZF**. I.e., converts any (one of 16) of [dzhampov] in **jz**. Depending on the state of flag **ZF** is carried out one of two passages. Both they lead to the functions of the formation of the address of passage, the first - if condition is satisfied, the second - if no. But passage is further accomplished there, where during emulation **of call**.

I.e. if in the program to the protection it was as follows:

Jcc of xxxxxxx

That ASPR \$\$RTASPR - automated system of planned calculations substitutes this by the call VM, where, if we simplify, it is carried out:

Call Of getStolenJmp Call Of convertFlags Jz xxxxxxx Jmp of xxxxxxx Well here, with emulation **of jcc** also they were dismantled. Remained most complex - emulation **of cmp+jcc**.

Again let us return to the function, which determines the type of the stolen instruction and let us place indicator on following conditional [dzhamp].



You will memorize this address, since for us it is necessary to here still return. **Call** to **00C5844E** emulates the instruction **of cmp**. We place on it indicator and harvest Enter. Now we proved to be directly in this function. We see below this code only:



Call of edx to **00C58260** determines the first operand for the comparison. If into **eax** after the fulfillment of function the number from 0 to ", then the first operand of - one of eight registers:

Number of the register	Register
0	Eax
1	Ecx
2	Edx
е	Ebx
4	Esp
shch	Ebp
1	Esi
"	Edi

Otherwise this is constant and its value is calculated only below. We see below the same code:





Entire the same, only for the second operand. Even we see below this:



Call of edx to **00C582DC** determines the type of that emulated **of cmp**. In **eax**, after the fulfillment of this [kella] and command **and eax**, **7F**, is located the number, which designates the type of emulation:

Number of Mask of the comparison

the	
comparison	
0	Cmp of dword of ptr [???????]???????
1	Cmp???????, dword of ptr [???????]
2	Cmp of byte of ptr [???????]??
е	<pre>Cmp??, byte of ptr [???????]</pre>
4	Cmp???????????????

Where instead of questions can stand either register or constant. Jmp to 00C582E6 accomplishes a passage to the instructions, which initialize one or another type emulations.

For example, during emulation of cmp of dword of ptr [of eax], ebx it is necessary to first obtain dword to eax, to and then compare it from ebx, that also make these instructions.

We look still below

00058339	E8 DECCFEFF	CALL 00C4501C
00C5833E	8B5424 10	MOV EDX,DWORD PTR SS:[ESP+10]
00058342	8BC5	MOV EAX,EBP
00058344	E8 CBFEFFFF	CALL 00C58214
00058349	83C4 14	ADD ESP.14
00C5834C	50	POP EBP
00C5834D	SF	POP EDI
00C5834E	SE	POP ESI
00C5834E	58	POP FBX
AAC5835A	čã	RETN

Call to **00C58344** answers directly for the comparison of operands. Before its fulfillment into **eax** lies the value of the first operand, into **edx** of - of the second. All, with emulation **of cmp** were dismantled, now let us look how is emulated **jcc** after it. You do remember, I did request to memorize address, before the entrance into this function? Let us return to it.



Jmp to 00C58456 is altogether only debris instruction for the trapping of disassembler. We place on it indicator, harvest Enter even we see that the instructions recognized correctly.

00C58459	3300	XOR EAX,EAX
00C5845B	8A43 04	MOV AL, BYTE PTR DS:[EBX+4]
00C5845E	8855 F8	MOV EDX.DWORD PTR SS:[EBP-8]
00C58461	8B5482 40	MOV EDX DWORD PTR DS:[EDX+EAX*4+40]
00C58465	8BC6	MOV EAX,ESI
00C58467	FFD2	CALL EDX
00C58469	8BD8	MOV EBX.EAX
00C5846B	8B4D 10	MOV ECX DWORD PTR SS:[EBP+10]
00C5846E	8BD3	MOV EDX.EBX
00C58470	8B45 F8	MOV EAX, DWORD PTR SS:[EBP-8]
00C58473	E8 74FBFFFF	CALL 00C57FEC
00C58478	84CØ	TEST AL.AL
00C5847A	~ 74 17	JE SHORT 00C58493
00C5847C	✓ EB Ø1	JMP SHORT 00C5847F
00C5847E	- E9 8B45F88B	JMP SCBDCAØE

However, and that we do see? Yes nothing else but previous emulation **of jcc**, only with another address. I was shocked! Why two times to write one and the same? I think that you will be dismantled themselves.

Well here we dismantled cursor VM. As you see nothing complex. To there remained only write script, that I already made. But! Script is not a little stable. In it there are several [bagov], which I will not correct. This script was my first script for OllyScript, I studied his possibilities and commands. Therefore he is not very optimized and it is terribly realized. Known [bagi]: Incorrect algorithm of the restoration of that emulated **of call**. Necessary to correct **push** only if with obtaining of the type of the stolen instruction it will be accurately known that this **call**. But script restores push always at the given moment, if it costs before **call VM**, and this not correctly and there were already errors.

Script itself searches for the empty place for the at the end current region of memory under the restored instructions and it is not always correct. Sometimes simply does not be sufficient vacant place and the part of instructions is not restored. I treated this by the fact that with the load of program intercepted **VirtualAllocExA** and looked in the stack the size of the inquired memory, if it coincided with the size of necessary to me region memory, then increased it by 1000h.

For the fastening of knowledge, you [perezapustite] program and place [brjaki] on the obtained control points. It will be useful to with its own eyes see work VM.

Now it is possible to restore all stolen instructions and to [sdampit] this region memory and to hitch it as the new section to the dump. But for the fact that to restore the stolen instructions, to script it is necessary to indicate the base address of section, which will be added to the dump. But for this by first [sdampim] our program. And here here again problem. ASProtect, besides the theft of instructions, is occupied even and by the theft of the calls API of functions, substituting call API, by the call of its function, which is located in the chosen region memory. Therefore, if we to [sdampim] program, these calls will indicate in anywhere. It means it is necessary to first restore all which is connected with the import.

Restoration of the import

In this version Of aSProtect'[a] the import is protected considerably stronger. Protection appears just as in the previous versions, and it here works differently.

For those, who are not familiar with the old protection. Pass into the section of the code. For this we harvest Ctrl+G, we write with 00401000 and harvest Enter. Now we harvest by the right button of [myshi]-> Of search of for->All of intermodular of calls (to find all calls of functions).

even works according to another principle. Earlier than ASProtect instead of the address API in IAT substituted the address of its adapter and for API of function was caused adapter on API function. Now ASProtect takes function from IAT, searches for all its calls and substitutes them by the calls of its universal function, and then is moved away the address API of function from IAT. Thus! Now in order to restore this function, it is necessary to learn, which for API function causes this adapter and to restore the address of this API in IAT, to and then restore its call. It is here with this exactly of problem in this version. Theory is one and the same. It is necessary to learn beginning and end of the table of import. Then as that to learn that for API causes the function of - adapter. To find this function in IAT (if there is), if no, then write it into the end IAT. To change the call of adapter to the call API of function. In the old versions Of aSProtect'[a] (from 1.33 to 2.00) the adapter worked as follows:

Generally the this was not the adapter, but the function, which, it formed. Those, who are familiar with this concept, as Delay Of import of - know, this is it and there is. With turning of program to this function, ASProtect calculated the address of function necessary TO API and formed new adapter to it, then money-changers the address of the call of its function to the address of adapter. In more detail read the appropriate articles. I will say only that for the formation of the adapter Of aSProtect used API the function Of virtualAlloc. After placing on it [brjak] and, after neglecting program, we jingled to VirtualAlloc and in the stack, on the specific displacement was visible the address of stolen API.

Now everything otherwise. Now ASProtect uses immediately two forms of adapters. They appear equally - the call of one and the same function in the chosen memory. But, in one case Of aSProtect continues to as before cause VirtualAlloc, to form adapter and to [propatchivat] the place of call, and in other of - it connects its new VM for emulation of the call API of functions. For this he does not use VirualAlloc and not [propatchivat] the call of adapter. But how to us to now learn the address of [ukradenoj] API of function? Nothing complex. It suffices to a little [potrassirovat] and we will see in the stack address API. But there is a universal method, with the installation of [brjaka] on API the function, which uses ASProtect for determining the address API. I thought that ASProtect cannot but use not one API of function for the work its VM. Let us place [brjak] on the access to the memory in the section of the code in the library of kernel32.dll.



5D C2 0400 RETN 4 7C801DA7 This API the function Of loadLibraryA, with the aid of it ASProtect [podgruzhaet] or, if it is already loaded, is obtained the base DLL, in which is located stolen API the function. We look into the window of the registers:

7C801DA5 7C801DA6

ĞΒ

	-		
Regi	isters (FF	PU)	
EAX ECX EDX	00CA132C 00C70D16 00C70D16	ASCII	"KERNEL32.dll"
EBX ESP EBP	00000007 0012FD24 0012FD2C		
ESI EDI EIP	0012FD6F 7C801D82	ASCII kerne	"GetACP" 132.7C801D82

To [uuu]... This is already very interesting. Give [dotrassiruem] to the instruction **RETN 4**, that would load necessary DLL, and then let us look, where indicator to the name of function will move. And so, we remove [brjak] from the section of the code:

7C:	раскир		'	132.70801098
7C	Сору		۲	7C80E2F0 • SS•[FBP+8]
20	Binary		×	DS:[<&ntdllstrompi>]
źġ	Assemble	Space		
20	Label	:		132.7C801DAA
20	Comment	;		
70	Breakpoint		×	Toggle
701	Run trace		۲	Conditional
70	Follow	Enter		Conditional log
	New origin here	Ctrl+Gray *		Run to selection
Adı	Go to		۲	Memory, on access
00! 00!	Follow in Dump		۲	Memory, on write
00! 00!	Search for		•	Remove memory breakpoint

we harvest Ctrl+F9 and we at the end function. We look into the window of the registers:

Registers (FPU)				
EAX	70800000	kernel32.7C800000		
ECX	708018F6	kerne132.70801BF6		
EBX	00000007			
ESP	0012FD30			
EBP	0012FE8C			
FDT	00L70D04	OSCII "GetOCP"		
EID	200121007	keypel22 7001007		
EIF	CS0IDH7	Kernel32.7C80IDH7		

It is excellent! In eax is located the base of the necessary of dll, while into edi indicator to the name of the stolen function. Who does not know - this two parameters, the necessary for API functions Of getProcAddress, which returns the address API of function. The experiments can be ended on this, since more than ASProtect API functions does not cause (only stolen). Yes to us in the principle of more than anything and it is must. Now it is possible to write script on the restoration of adapters. But there are difficulties. To us it is not possible to allow the fulfillment of the stolen function, otherwise we will lose control over the program. If we interrupt work VM, then the balance of stack will be disrupted and during the restoration of the large number API of functions it simply will fall through and program will collapse. Therefore before the fulfillment of adapter we will preserve the register of esp, and after the restoration of adapter - respectively restore the register of esp. To calculate the address API of function we will be with the aid of the function Of getProcAddress. Cause it we will be with the aid of the injection of the code. Everything else, as in the previous versions Of aSProtect'[a]. by the way, the it turned out that this method works also for the previous versions. Therefore we will restore all adapters by one script. Script, as you surmised, I already wrote (IAT_Recover.osc) it itself finds the adapters Of aSProtect'[a] it restores them, but it should indicate beginning and end IAT.

I fairly often hear, that in many appear the problems with the determination of beginning and end IAT, and also about the fact that in IAT much rubbish is, that among it also are encountered API of function. Give let us try to solve this problem. For this let us pass into the section of the code and will find any call API of function.

00401000	55	PUSH EBP	
00401001	8BEC	MOV EBP,ESP	
00401003	51	PUSH ECX	
00401004	51	PUSH ECX	
00401005	56	PUSH ESI	
00401006	33F6	XOR ESI,ESI	
00401008	E8 F3EFA400	CALL 00É50000	
0040100D	C46A 07	LES EBP, FWORD PTR DS:[EDX+7]	Modification of segment r
00401010	8D4D F8	LEA ECX DWORD PTR SS:[EBP-8]	-
00401013	51	PUSH ECX	
00401014	68 04100000	PUSH 1004	
00401019	50	PUSH EHX	
0040101A	FF15 84D25500	CALL DWORD PTR DS:[55D284]	kernel32.GetLocaleInfoA
00401020	8500	TEST EAX,EAX	
00401022	74 1F	JE SHURI FUNIEX 1.00401043	

We see that is caused BY API the function **Of getLocaleInfoA**, whose address lies to **0055D284**. This address (**0055D284**) is one of the addresses IAT. Let us determine the beginning of the section, in which is located this address. For this in the command line we write **with d of 0055D284** and harvest Enter. We see:

Address	Hex dump)				ASCII
Address 00550284 00550294 00550294 00550204 00550204 00550204 00550204 00550204 00550304 00550304 00550324 00550344 00550344	Hex dump 7E D4 80 CF C6 80 C7 A0 80 93 8D 83 E0 C6 80 39 9A 80 90 72 C4 DF 06 86 8A 2B 86 C4 CE 80 C4 CE 80 FD 79 91 16 1E 80	0 7C 05 A4 3 7C 65 A0 3 7C 52 70 4 7C 52 70 5 7C 9 68 6 7C 11 03 6 7C 7C 11 6 7C 7C 36 6 7C 7C 36 7C 28 228 37 3 7C 28 28 3 7C 28 22 7C 30 7C 28 3 7C 28 22 3 7C 28 22 3 7C 22 24	80 7C 51 80 7C 58 80 7C 2F 80 7C 2F 81 7C 29 80 7C 29 80 7C 94 81 7C 89 81 7C 89 81 7C 89 81 7C 29 83 7C 29 83 7C 29 83 7C 29 83 7C 25 81 7C 59	28 81 7C F C7 80 7C 2 2C 82 7C 8 FE 80 7C 9 C7 80 7C 9 C7 80 7C 3 EE 81 7C 5 8F 83 7C 3 94 83 7C 8 29 81 7C 1 09 91 7C 2 1E 80 7C E B8 80 7C E	1 BA 80 7C 3 CC 81 7CC 9 E3 83 7CC 8 93 83 7CC 1 03 91 7CC 6 AA 80 7CC 6 E8 81 7CC 6 E8 81 7CC 6 011 81 7CC 6 018 81 7CC 6 018 81 7CC 5 17 80 7C	ASCII * 44,44,41,0(5;8) + 44,84,80,5,81,9 0,81,44,5,91,41,9 9,41,44,5,91,41,9 9,41,44,5,91,41,9 9,41,44,5,91,40,14 * 44,165,11,01,19 + 4,11,61,90,13 + 4,11,51,04,01,19 + 4,11,11,11,11 + 4,11,11 + 4,11,
0055D364 0055D374 0055D384 0055D384	81 9A 80 40 7A 93 E6 2B 81 F5 9B 80	0 7C B3 9E 3 7C A1 97 1 7C F0 78 3 7C AF 2B	80 7C 3D 83 7C 2D 82 7C 53 81 7C 50	04 91 7C D 2C 82 7C 2 34 81 7C B 97 80 7C 0	4 05 91 7C A E8 81 7C 1 E2 81 7C 5 10 90 7C	6ЪA¦ ЮA¦=♦C¦⁵♣ @z9¦6ЧΓ¦-,B¦★ш ц+6¦ЁхB¦S46¦∭⊤ ïЫΩ!*+5!РЧΩ!≛▶
000000074			01 10100	7 00 10 0		

Commanc d 55D284

D address – Dump at address

This is IAT. To novice certainly this window not about which will say, this understanding will arrive with the time (experience). Now let us make a window of dump active (sufficient to call to it by mouse) and let us press the key **For home**. Now we in the beginning section with the directory of import.

Add:	ress	He	x du	amp														ASCII	
005	50000	C3	CA	DE	77	F0	6B	DC	77	4A	CF	DD	77	1B	76	DC	77	H≏ wëk <u>a</u> wJ≓ w	ŧψ
005	5D010	53	77	DC	77	34	C5	DE	77	1B	D1	DE	77	E7	EB	DC	77	Sw_w4+ w++ w	чы
005	5D020	F4	EΑ	DC	77	1B	C4	DE	77	E5	ED	DC	77	83	78	DC	77	Tъ <u>∎</u> w← wxs <u>∎</u> w	Γx
005	5D030	BB	D5	DE	77	10	CC	DE	77	23	C1	DE	77	63	DЗ	DF	77	╗╒∎₩⋫╠∎₩#∸∎₩	сш
+h·	- + 1	20	~ i 1	na	÷	+	fr	om	· +	-ha	<u> </u>	~ d	d٣	~ ~	a	of	· 1		

It leaves, that begins it from the address of 0055D000. Let us memorize and [perezagruzim] program. Thus far ASProtect it did not begin its work, let us look, which is located in th section e interesting us.

Address	Hex dump								ASCII	
0055D000 0055D010	24 43 B3 2F F4 54	2A 99 1 73 20 8	9 A2 00 7 10 19	9 9E 65 8 BE 70	C9 C3	D7 D1	BA 3 6F 9	C 06 0 6E	\$C #Щ∔в.Юе╔╟╫ ∠@Ts_3 ∗↑ ₅!_Ы⊟п	
0055D020	1D 49 49	09 23 5	B 09 C9	22 22	0E 93	<u>98</u>	93 9	ĕ šč	#II.#[-+""#9ШS	,
0055D030	8F 74 ED	42 4B 2	5 90 08	3 70 44	BE 2D	É4	23 2	6 AC	RtaBK%bapDP-0#	ŧ
0055D050 0055D060	E1 0D 5E 98 A2 8D	E4 F6 0 C8 C3 A	B 90 E7 0 8F 80	7 CC DF 9 FB D5	61 E5 87 20	25 92	56 E 12 5	7 BE 2 36	∣с.^ф9∂Рч∦≡ах%∪ Шенч⊦аПАлка.Т≰	
000000070	<u>àn an nr</u>	60 40 8	<u>a zh ca</u>			- áz	<u>26 C</u>	<u>7 66</u>	In a contract of the o	
Comman	d 00550	000			-					

This is none other than the rubbish, not necessary not To aSProtect'[u], not to us. Let us place indicator on number 24 to **0055D000** and let us twist the window of dump downward to the end itself by mouse for [skrol]. Now let us stop up the key **For shift** and will call by mouse to quite last [chiso] of this section. Well here was isolated entire section. Now we harvest by the right button of mouse we select:

Address	He	(du	amp														ASCII
00582F60 00582F70 00582F90 00582F90 00582F80 00582F80 00582F80 00582F00 00582F00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	000000000000000000000000000000000000000	00 00 00 00 Ba	00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	
00302FF0	99	00	1	Bir	hary									E	dit		Ctrl+E
				La	bel						:			F	ill w	ith I	00's
				- D	1	:_	L .						- N		-: 11	all i	cc!-

Well here. Now entire section is oppressed by zero, and it means, there is no rubbish greater:

	Address	Hex	dump)													ASCII			
	0055D000	00 0	90 00 90 00	00	00	00	00	00	00	00	00	00	00	00	00	00	•••••	• • • • • • •		
	0055D020	00 0	10 00	00	00	00	00	00	00	00	00	00	00	00	00	00				
	00550030	00 0	90 00 30 00	00	00	00	00	00 00	00 00	00 00	00 00	00 00	00 00	00 00	00 00	00	•••••	•••••		
	00550050	00 0	õõ õõ	00	00	00	00	ŏŏ.	ŏŏ.	ŏŏ.	00	ŏŏ.	ŏŏ.	ĕĕ.	ŏŏ.	õõ				
	00550060	00 0	10 00 10 00) 00) 00	00 00	00 00	00 00	00 00	00 00	00 00	00 00	00 00	00 00	00 ЙЙ	00 ЙЙ	00 00				
	00550080	00 0	0 00	00	00	00	00	õõ.	õõ.	ÖÖ.	00	õõ.	õõ.	õõ.	õõ.	õõ				
	00550090	00 0	10 06 10 06	1 00	00	00	00	00	00	00	00	00	00 00	00 00	00 00	00				
	0055D0B0	00 0	0 00	00	00	00	00	00	<u>00</u>	00	00	00	<u>00</u>	<u>00</u>	<u>00</u>	00				
	00550000	00 0	10 00	00	99	99	99	66	66	66	99	66	99	99	66	99	•••••			
Now we rea	ich OEF	, ai	nd	10	эĸ	,	wh	at	C	lo	We	i e	nav	ve	1:	n	this	sect	10	pn.
	Address	Hex	dump														ASCII			

We see that the section was filled up with the addresses of the imported functions. Let us look end IAT. It is here here easy to be mistaken. Therefore I always search for the empty place at the end for section and for itself I consider that this is a end IAT.

Address	Hex	du	IMP														ASCII
0055EA80	33	12	00	07	33	57	0E	26	00	BA	DC	66	AC	90	B2	E5	3\$.•3₩#&4 _fmP
0055EA90	6C	FØ.	8D	89	B1	01	F6	ĈĎ	39	7E	75	Â9	6B	12	80	26	LEH0@09=9"unk\$
0055EAA0	E8 :	2Ē	44	20	2B	B 3	DĒ	18	2C	10	18	19	75	ЙЙ	Â1	8Ē	w/D-+1∎t.▶t∔u.
0055EAB0	58	78	33	ĒĨ	47	26	38	31	ÃĐ.	Bī.	Đ9.	D9	23	68	DS.	SC.	Xx3c6&81+#+
0055EAC0	17	4Č -	ĒŽ.	D4	4B	-99	4A	ĒĨ	29	ЗĀ.	BF.	īċ	16	28	42	6Ē	≠Lŭ ^t KШJe):n∟=(
0055E0D0	Ê8 I	ÉŘ.	ŻŔ.	5É	89	2Å.	26	<u>й</u> З.	DÉ.	ΞĒ.	B2	2Ă.	18	65	śñ.	<u>92</u>	ui9n (1#8#∎?₩#†e
0055E8E0	2ñ	6Š.	ŔŘ.	ĂB.	ΠŔ.	9Ë	$\overline{2}\overline{B}$	ĂŠ.	10	34	19	ŘΫ.	Ξêč.	ĞĔ.	й4	9Ē	-et al 8tr, 44%bo
0055E8E0	49	ĂŽ-	6Ř.	27	FC	1A	йñ	ЙЙ	ĥŽ.	йŔ	Âñ.	ăй	ñй	й'n	йŔ	ñй	Tek N+
0055EB00	00	ññ.	ññ	ñà.	00	Â0	ññ.	ññ.	ñ0	ññ	ññ.	ññ	ññ	ññ	ññ.	Ö Ø	
0055EB10	ññ	ăă -	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	йй	
0055EB20	ăă I	ăă -	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	йй	
0055EB30	ăă I	ăă -	ăă.	ññ.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	ăă.	йй	
0055EB40	ññ	ăă -	ññ	<u>aa</u>	ññ	ññ	ññ	ññ	ññ	ññ	ññ	ññ	ññ	ññ	ññ	ññ	
0055EB50	ãã l	ăă -	ãã.	ăă	ăй	ăă.	ăă.	ăă	ăă.	ăă.	ăă.	ăă	ăă	ăă.	ăă.	ăă	
00000ED00	88	88 88	20	20	20	20	20	20	20	20	20	20	20	20	20	20	

We will consider that began IAT of 0055D000, and the end of 0055EB00. If we did not drive in this section by zero, then she would be entire filled with rubbish (except the addresses API of functions) and we did not find vacant place.

Apropos adapters. As you know programs written to **Delphi** they have adapters on API of the function of the form

jmp of dword of ptr [of xxxxxxxx],

and on SI of - of call of dword of ptr [of xxxxxxx]. It is natural that also [opkody] in these instructions are different: in the first case FF25, in the second of - FF15. Therefore to script necessary to indicate what form passages we restore. You do remember the first obtained call API?

It means nevertheless FF15. Therefore when script will ask to use [obkod] "of call" (FF15) necessary to harvest yes. Well that zh, let us neglect script and will wait for the end of work. I recommend the decreasing of the window Of ollyDbg to the minimum,

since with the work of script, [Olli] in the line of state he writes:

Too long (recursive?) SEH chain

And in this case very slowly it works, and here if window was minimized, then everything is normal and script very rapidly works.

We start script and on the first a question answer **0055D000**, on the second **of 0055EB00** well and to third **yes**. Through several minutes we see that the script finished its work.



200h = 512 functions are restored! Yes, I tried to optimize script, that he so rapidly worked. Let us look, did add the script of the address of functions in IAT.



We see that no. Means all addresses it it found in already existing IAT. If it added function, then respectively would increase size IAT.

Here now we can [sdampit] program to the disk.

We start PETools and [dampim] program.

📕 d:\windows\system32\taskmgr.exe	00000724	ŧ 01000000	00025000	
🔄 e:\!rekill\programs\!!!trial\fontex~1\fontex	90000400		- 002CE000	
📓 d:\windows\system32\calc.exe		p Full	0001F000	
<pre> @e:\!rekill\programs\dumpers\petools\petools </pre>	C Dum	p Partial	00053000	
	Dum	p Region		-

To before restore import, give to [otrezhim] in dump excess sections. Indeed now file is completely unpacked, import let us soon restore, with VM will be finished - why to us now the section Of aSProtect'[a]? But it cannot be thus far cut off, since ASProtect steals even and resources. If we will cut off its section, then let us be deprived of the part of resources. Give them let us restore.

I use **ResFixer by of seeQ**. We start this remarkable utility and we select our dump. We see that the utility counted all resources of our dump. Now let us twist window downward. We see:

Res type	File Offset	RVA 2 Data	Size	Sec N²	Sec name	
BITMAP	001C6A18	0026AF98	00000144	04	.rsrc	
ICON	001C6A28	0029DDBC	000002E8	05	.data	
ICON	001C6A38	0029DC94	00000128	05	.data	
ICON	001C6A48	0029D3EC	000008A8	05	.data	
ICON	001C6A58	0029CE84	00000568	05	.data	
ICON	001C6A68	0029C1DC	00000CA8	05	.data	
LCOM	00100470	002000774	000000200	05	data	

Everything which is marked by red, ASProtect of [sper] to its section and we should return this in the place. We select as Rebuild Of method:

- Rebuild method:
O Method 1 (Cut and paste)
 Method 2 (Full reconstruct)

And we harvest Rebuild. We prescribe any, pleasing itself name and preserve the restored section to the disk.

Now let us open our dump with the aid of **PETools** and will open Directory Of editor (we harvest to the button Of directories). We look RVA of the directory of the resources:

Le sussiti à sussitie l'			
	RVA	Size	
Export Directory	00000000	00000000	 Н
Import Directory	002CE000	0000012C	 Н
Resource Directory	001C3000	000D0D88	 Н

Now let us open dump and the restored section of resources into WinHex'[e]. in the dump let us pass to the address of 001C3000, for this we harvest Alt+G, enter 001C3000 and harvest Enter.

Dumped_.exe rsrc.bin

Offset	0	1	2	3	4	5	6	7	8	9	A	в	С	D	Е	F	
001C3000	00	00	00	00	00	00	00	00	00	00	00	00	01	00	0C	00	
001c3010	E8	4B	00	80	78	00	00	80	01	00	00	00	90	00	00	80	иК.ЂхЂ.

It is now discovered supplementary sheet with the directory of resources, we harvest Ctrl+A (to isolate everything), Ctrl+Shift+C (to copy as the sequence of bytes) and we pass to the supplementary sheet with our dump. We place cursor on **00** to **001C3000** and harvest Ctrl+B (to put the sequence of bytes), we agree with all questions and preserve our dump. Let us look to [ikonku] of our dump, if it did not disappear, then everything is normal.

↑Имя	Тип	і Разме	ер Дата		Атриб
\$ []		<dir></dir>	03.03.2	2006 02:47	
🚰 Dumped	exe	2 940 92	28 03.03.2	2006 01:46	-a
₽ Dumped_r	exe	2 940 92	28 03.03.2	2006 02:47	-a
	bin	856 00	64 03.03.2	2006 02:26	-a

We see that [ikonka] on the spot. Let us again open file into PETools and will open Section Of editor (we harvest to the button Of sections) and let us remove both sections after the section of rsrc.

	00056000	0015D0(Kill section (from file)		E0000040	
	00010000	0016300	Fill section		E0000040	
.rsrc	000D1000	001C300			E0000040	
.data	00039000	00294060	00037000	002,77000	E0000040	
.adata	00001000	002CD000	00001000	002CD000	E0000040	

After this, we should correct some parameters in PE title, for this let us shut the current window and will open Image Of optional Of header Of editor (we harvest to the button Of optional Of header). We harvest to all [voprosiki] in this window:

П		1					ł.				
ł	Image Optional Header Editor										
	-Image Optional Head	der Inf <u>ormation</u>	s [HEX]				1				
-	Magic	010B	Major SubSystem Version	0004							
	Major Linker Version	07	Minor Subsystem Version	0000							
	Minor Linker Version	0A	Win32 Version Value	00000000							
	Size Of Code	0015C000	Size Of Image	002CE000	?	-	ł				
	Size Of Init Data	00137000	Size Of Headers	00001000	?	-	-				
:	Size of UnInit Data	00000000	Checksum	00000000	?	-					
	Entry Point	00001000	D Subsystem	0002							

We shut **PETools**.

Now we start **ImpREC** for restoring the import and we select our process. In the field **OEP** it is possible to introduce although that (main thing in the limits of file) I entered 1000, since present OEP in the file we as yet do not have. In the field **RVA** we introduce the address of the table of import minus **Of imageBase** (address of load) of our program, i.e., **of 0055D000-00400000=0015D000**. In the field **Of size** we introduce **0055EB00-0055D000=1b00**. We press **GetImports**, then **ShowInvalid**.

Imported Functions Found		
rva:0015D2F4 ptr:00C47290		
rva:0015D2F8 mod:kernel32.dll ord:0242 name:LoadLibraryA		
rva:0015D2FC mod:kernel32.dll ord:025E name:MoveFileA		
Let us try to use special [plagin] for the recognition of t	the	old
adapters Of aSProtect'[a]. in me of such [plaginov] much, &	out	
correctly works only one.		
Imported Europtic IN VPUOLI 013		

Imported Function	Kiypton 0.5	
rva:00150-004000-47200	ASProtect 1.2x	
	ASProtect 1.23 rc4	
rva:0015 Disassemble / HexView	ASProtect 1.22	

We see that the function recognized - this **GetProcAddress**. We again harvest **ShowInvalid**.

Imported Functions Found	
rva:0015D9CC ptr:00000001	
🚊 ? FThunk:0015D9D8 NbFunc:2 (decimal:2) valid:NO	
🚊 ? FThunk:0015D9F8 NbFunc:6A (decimal:106) valid:NO	
rva:0015D9F8 ptr:54415025	-

Now, if we twist window to the bottom itself, we will see, that there is not one identified address API of function. Therefore boldly we harvest:

rva:0015D9	
rva:0015D.9	Cut thunk(s)
rva:0015D9	Delete thunk(s)
178.001303	

All unrecognized addresses were removed. This it indicates only that all [vosstanovlenye] with the aid of the script adapters already have a address in IAT. But here if one address was added in IAT, then in the very to the bottom of window we would see identified API of function.

Now we harvest FixDump and we select our dump. It is finished with the import.

Is discovered dump with the aid of PETools and is discovered Section Of editor (we harvest to the button Of sections). We should add the new section, into which we will place th code e stolen By aSProtect'[om].

2	abric		•			
s	ections Ec	litor	Load section from disk			×
Г	Sections In	formations [HEX]	Save section to disk			
	Name	Virtual Size	Add section	Offset	Characteristics	
		0015C000 (Kill section (from header)	p1000	E0000040	
		00056000 (5D000	E0000040	
		00010000 (Kill section (from file)	<u>β</u> 3000	E0000040	
	.rsrc	000D1000 (Fill section	23000	E0000040	
	.mackt	00003000 (This section		E0000060	

PETools requests to introduce some important parameters: the name of section, the virtual and real size of section. But so he asks, we do want to put section from the file, to fill with its zero or to put the description of section only into the title, but not to put it in the file. As the name it is possible to introduce everything, anything. In order to determine the size of section, let us open into OllyDbg the map of memory and will look the size of region memory with the stolen code.

Memory map							х
Address	Size	Owner	Section	Contains	Туре	Access	
00E20000	00001000				Priv	RWE	
00E30000	00005000	0			Priv	RWE	
00E40000	00000000				Priv	RWE	

It means we introduce into PETools as the virtual and real size of the section of - 00005000 we select the flag Of fill of with of 0x00 so that the editor would create both the description of section in PE title and section itself in the file.

Add New Section 🛛							
ESection Header	[HEX]	Section Data				
Section Name	Section Name .scram 💌			Load from binary file Fill with 0x00			
Size of Raw Da	Size of Raw Data		Ŏ Add header	only			
Size of Virtual Data		00005000	Add	Cancel			

We harvest Add and look VA of new section. .mackt 00003000 00294000 00003000 00294000 E0000060 .scram 00005000 (00297000) 00005000 00297000 E00000E0

VA of this section will be new base for the stolen code. Specifically, this value must be introduced into the script, which restores the stolen instructions.

[Perezapuskaem] program we reach OEP. All, it is possible to start script on the restoration of the stolen instructions. We start script, await thus far it it [samonastroitsja] and it will inquire the new base of the code. We after which introduce VA of new section in the dump plus Of imageBase, i.e., of 00297000+00400000=00697000.

00E3030D		6A 60		PUSH 60	
00E3030F		68 608	CBD31	PUSH_31BD8C6C	
00E30314		66:9C		PUSHFW	
30E30316		57		PUSH EDI	
30E30317		ØBFB		OR EDI,EBX	
30E30319		8107 1	CDD73BD	ADD EDI,BD73DD1C	
00E3031F		8D7C2		LEG FOT BUODD DTD OG	
30E30323		8D7C2	OllyScrip	t - Input request	×
30E30327		52			
30E30328	\sim	EB 01			
30E3032A		F2:	Enter ne	w base of this code	
30E3032B		53			
30E3032C	\sim	EB 01	006970	00	
30E3032E		6955	1		}
30E30335		CA 23			1
30E30338	\sim	EB 02	L OK		Cancel I
30E3033A		CD 20		·	Cancor
30E3033C		68 68			
0F30341		65 EB-			

We harvest Enter and await, until script restores instructions. All, script worked out. We harvest * on the digital keyboard in order to move for the instantaneous value of eip. But now let us press the key For end and let us prove to be the at the end chosen memory. We harvest Pages Up, until zero, are changed into the code. Beginning from the address of 00E346ED it begins the imposing list of conditional [dzhampov] and commands of comparison. This is the restored instructions. Here are some fragments.

	* * *
00E346DD 0F84 00CSFFFF 00E346E3 E9 43D1FFFF 00E346E4 3BF3 3BF3 00E346EA 0F84 2FC9FFFF 00E346F0 E9 C3D5FFFF 00E346F0 E9 C3D5FFFF 00E346F0 A ABF3 00E346F0 A ABF3	JE 00E30C83 JMP 00E3182B CMP ESI,EBX JE 00E3101F JMP 00E31CB8 .IGF 00E32485 * * *
00E34838 ^ E9 E9CEFFFF 00E34830 813D 90265C00 0 00E34840 ^ 0F85 EAC7FFF 00E34840 ^ E9 46CEFFFF 00E34852 ^ 09402FFFF 00E34850 813D 84825800 0 00E34860 ^ E9 9005FFFF	UMP 00E31726 CMP DWORD PTR DS:[5C2690],0 UMP 00E31037 UMP 00E31698 UE 00E31AE9 UMP 00E31AE9 CMP DWORD PTR DS:[5BB2B4],1 UMP 00E31676 UMP 00E31E02
	* * *
00E348CB ^ E9 34D2FFFF 00E348D0 ^ 0F84 54D4FFFF 00E348D0 ^ 0F84 54D4FFFF 00E348E5 ^ 0F85 14BDFFFF 00E348E8 ^ E9 3FC6FFFF 00E348E8 ^ E9 3FC6FFFF 00E348F8 ^ 0F84 F7CDFFFF 00E348F8 ^ 0F84 F7CDFFFF 00E348F8 ^ 0F85 A5BFFFFFF 00E3498F8 ^ 0F85 A5BFFFFF 00E34906 0000	UMP 00E31804 (CHP DWORD PTR DS:[58B2B4],1 UF 00E31034 UHP 00E31996 UH2 00E305FF UF 00E3062F UHP 00E316ED UHP 00E316ED UHP 00E31686 UHP 00E318BA ADD BYTE PTR DS:[EAX],AL

Now it is possible to [sdampit] region memory with the stolen code. We start PETools, we select our process even we indicate that we should [sdampit] region memory.

I DALL					T D	Terrer Cine	•	1
I		Dump Region						l
irekill\programs\dumpers		Dump Partial	98		00400000	00053000	Ţ	
🔚 e:\!rekill\pro	grams\!!!trial\f	Dump Full		3C	00400000	002CE000		
a twingowst	systemszynspan	ICICXC	000001	51	01000000	00037000		1

In the appeared window we search for our region and harvest Dump.

Address	Size	Protect	State	Туре	
00E30000	00005000	EXECUTE READ/	COMMIT	PRIVATE	
00E35000	0000B000	NO ACCESS	FREE	NONE	1

Now we repeat all operations, that also with the directory of resources. Is discovered dump and [sdamplenyj] region into WinHex'[e]. in the dump we pass to **00297000** (beginning of new section). We pass to the supplementary sheet with the [sdamplenym]

region Of ctrl+A, Ctrl+Shift+C. We pass to the supplementary sheet with the dump Of ctrl+B. We preserve dump. It is finished with the scrambler and VM (I I hope). Is now reparable OEP. OEP will be equal to address OEP in the chosen memory minus the old base of the stifling code plus the new base of the stifling code. OEP = of 00E3030D-00E30000+00297000=0029730D. Is discovered our dump with the aid of PETools and is discovered Image Of optional Of header Of editor (we harvest to the button Of optional Of header). In the field Of entry Of point we change 00001000 by 0029730D and preserve changes. In principle the program is unpacked. But always there are its "but!".

Let us open our dump into OllyDbg and will wait for, until the analysis of the code ends. We see that the now stolen code is located through another address. It turns out that we did everything correctly.

0069730D	6A 60	PUSH 60
0069730F	68 6C8CBD31	PUSH 31BD8C6C
00697314	66:90	PUSHFW
00697316	57	PUSH EDI
00697317	ØBFB	OR EDI,EBX
00697319	81C7 1CDD73BD	ADD EDI,BD73DD1C
0069731FL	8N7C24 39	I FO FNI NWARD PTR SS+FFSP+391
	_	

We start program F9. Program fell, and OllyDbg in [storoke] of state it writes:

Access violation when executing [00E30662] - use Shift+F7/F8/F9 to pass exception to program

To [ugu]. Error, with the starting of the code to **00E30662**. Interesting address. Indeed the same the address of the memory, where there was the stifling code! Similarly program where that still causes it. By [perezapustim] program let us pass into the section of the code (Ctrl+G, 00401000, Enter). Let us try to look the address **of 00E30662** as the constant. For this we harvest by the right button of [myshi]->Search **of for->Constant**. In the appeared window we write:

	Enter constant to search for 🔀
	Hexadecimal 00E30662
	Signed 14878306
	Unsigned 14878306
	Entire block
	OK Cancel
And [zhmjom] OK. Yes!]	was rights!
00485D66 L. C3 00485D67 CC	RETN INT3
00485060 3- E9 00485060 F8	DB F8

It is now necessary to recount displacement relative to new section and to change the address of [dzhampa]. New address = of 00E30662-00E30000+00297000 = 00297662. Now it is possible to change jmp of 00E30662 for jmp 00297662, but you do not hurry. I will say immediately that such [dzhampov] in this program THERE ARE VERY many. Therefore I wrote script on the restoration of such [dzhampov]. He is called JMP_VM_REDIRECT.osc. This script must be disposed to your program. Variables: **RegionVM_Start** of - the beginning of the chosen region memory, in which there was the stolen code. In me is equal **00E30000**.

RegionVM_End of - respectively the end of this region. In me is equal **00E30000+00005000 = of 00E35000**

RegionMain_Start of - the beginning of new section with the
[sdamlenym] stolen code plus Of imageBase.
I have - of 00297000+00400000=00697000.

We enter values in the script and start it in our dump. We see in second:



21h = 33 [dzhampa] are adapted to the new displacement. Now we pass into the section of the code, we place indicator on the very first command and it is moved by mouse for the sulky the window of the code into the bottom itself. We press Shift and will call by mouse on the last command of this window. We see that entire section of the code was isolated. Let us call by the right button of mouse and we select:

005238B0	. FF15 8CD6550	Searchillor	li SetActiveWi
005238B6	> 8B06	Find references to	•
00523888	· SBLE		
005238BH	. FF50 60	View	►
005238BD	. SBUE		
005238BF	. E8_1DFAFFFF	Copy to executable	Selection
005238C4	. 395E 54		
00523807	.~ 74 09	 Analysis 	

We shut the appeared window, OllyDbg it asks, we do want to preserve changes. We harvest yes and OllyDbg proposes to introduce the name of file. It is possible to preserve into the same dump, but I always preserve into the new file, so more easily to make a recoil, in the case of error. Is discovered into OllyDbg the preserved dump even we await, until the analysis of the code ends. We start program even we see that it again fell in this place:

0048E437 > 50 0048E438 . FF15 18D3550 CALL DWORD PTR DS: [<&kernel32.GetStdHandle 0048E438 . 1045 83FFF7 ADC BYTE PTR SS: [EBP+74FFFF83],AH 0048E444 . 3F AAS

This still that after command is such after the call API of function? For the first time similar I see. Let us roll up for the moment OllyDbg and will neglect one additional copy Of ollyDbg, in which let us open the protected program. Let us reach OEP and will pass to the address of 0048E438 and will install location counter (eip) to this address (Ctrl+ *). Now let us isolate several bytes after the call API of function establish [brjak] to the access to the memory.

0048E426 v	EB ØF	JMP SHORT FONTEX"1.0048E437	Comment	3	сто
0048E428 0048E428	83C8 FF EB 79	OR EAX, FFFFFFFF JMP_SHORT_FONTEX"1.0048E4A6	Breakpoint	►.	Toggle
0048E42D 0048E42F	8BC3 48	MOV EAX,EBX DEC EAX	Run trace	+	Conditional
0048E430 0048E432 0048E434 0048E437	F7D8 1BC0 83C0 F5 50	NEG EAX SBB EAX,EAX ADD EAX,-08 PUSH EAX	New origin here Go to	Ctrl+Gray *	Conditional log Run to selection
0048E43D 0048E43D	E8 C31B9C00 04 10 A5	CALL 00E50000 ADD AL,10 MOUS DWORD PTR ES:[EDI].DWO	Follow in Dump	+	Memory, on access
0048E440 0048E443 ~	83FF FF 74 3F	CMP EDI,-1 JE SHORT FONTEX"1.0048E484	Search for	•	Memory, on write

Now let us neglect program. They interrupted here:

We see that this where that in the chosen region memory and to us this does not approach. Again we harvest F9 and we here:

 00482440
 83FF
 FF
 CMP
 EDI,-1

 00482443
 ~ 74
 3F
 JE
 SHORT
 FONTEX"1.00482484

 00482445
 ~ 57
 PUSH
 EDI

But this already the code being been located almost immediately after the call API of function, you will memorize its [opkody] (83FFFF). Open diagnostic routine with the dump. Now let us place indicator on the address of 0048E43E and will press Ctrl+E.

Hexadecimal editor was opened. We see:

0048E438 0048E43E	÷	FF15 18D35500 10A5 83FFFF74	ADC BYTE PT	PTR DS:[<&kernel32.GetStdHan R SS:[EBP+74FFFF83],AH	
0048E444 0048E445 0048E446	÷	3F 57 FF15 5CD25500	Edit code a	t 0048E43E	×
0048E44C 0048E44E 0048E450	÷	85C0 74 34 25 FF000000	ASCII	ъъъът	
0048E455 0048E458 0048E458	:.	83F8 02 893E 75 06	UNICODE	БББ	_
0048E450 0048E460 0048E462	;~	804E 04 40 EB 09 83F8 03	HEX +00	18 45 83 FF FF 74	
0048E465	÷	804E 04 08	1		

We see that the bytes, on which we dwelled, be present, but here to them is even what that bytes and here their ASProtect stole and carried out somewhere earlier, after fulfillment API of function, also, to the return to the section of the code. Now at their place the debris bytes, which do not give to disassembler it is normal to recognize commands. Give we change them for 90 90 ([opkody] of nop).

HEX +02 90 90 83 FF FF 74

We harvest Enter even we see that now the instructions recognized normally.

0048E437	> 50	9	PUSH EAX	DevType
0048E438	. FF	15 18D35500	CALL DWORD PTR DS: [<&kernel32.GetStdHan	GetStdHandle
0048E43E	- 98)	NOP	
0048E43F	- 98)	NOP	
0048E440	83	SFF FF	CMP EDI1	
0048E443	~ 74	4 3F	JE SHORT DUMPED"2.0048E484	

But how to us to return the stolen two bytes? Let us look to the logic of the work of program.

First is caused the function Of getStdHandle, which returns to eax of [khendl]. Then they go ([jot]) unknowns ([aja]) to us the command (a) then of edi it is compared with -1. By the way the number - 1 (0xFFFFFFF) is the constant (INVALID_HANDLE_VALUE), which returns with the functions, which work with [khendlami] as the result of the unsuccessful attempt at the discovery [khenda]. It is interesting that after the call [API] of [khendl] it will be into eax, and to the validity is checked edi. Not about which he does not speak? So that finally it will be convinced of my theory let us look that it lies at the register of eax and edi in our dump, and in the protected program.

In the dump: In that protected:



EAX	FFFFFFF.
ECX	7C90FB71
EDX	00000007
EBX.	00E3030D
ESP	0012FF5C
EBP	0012FF58
ESI	014C4B23
EDI	FFFFFFF

As we see in the protected program the register of edi so it contains the result of the work API of function as eax. I know only one command, capable of copying value from eax into edi, and which occupies the size of - of 2 bytes. This is **mov of edi, eax**. We enter it instead of two **nop'[ov]**, for this we harvest gap, we introduce necessary command and we harvest Enter. We preserve and is discovered the preserved dump. We start program even we see that it again fell. In the window of the code nothing it is reflected, but it is here in the line of the state:

Access violation when executing [00E90000] - use Shift+F7/F8/F9 to pass exception to program

It is familiar, not so whether? Error, with a attempt at the fulfillment of the code to **00E90000**, only now this is already accurate not the address of our restored code. Means ASProtect where that still it stole the piece of the code and for us one must find it. For the beginning let us find the call of this piece. We search for just as past time as the constant in the section of the code.

00401000	55	DB	55	
00401001 00401002 00401003	8B EC 51	DB DB DB	⁸⁸ Enter constant t	o search for 🔀
00401004 00401005 00401006	51 56 33	DB	Hexadecimal	00E 90000
00401007 00401008 00401009	F6 FF 15	DB DB DB	Signed	15269888
0040100A 0040100C 0040100C	88D2 55 00	MÕV DB	Unsigned	15269888
0040100E 0040100F	6A 07		🔲 Entire block	
00401011 00401012 00401012	4D F8 51		OK	Cancel
	11	122	1.0	

And we again find \odot . only now this not [dzhamp]:

The required address is placed in the stack, and then the command of retn extracts it it from there and accomplishes to it a passage. Let us look, what do we have here in the protected program. The same. Therefore in the protected program we place location counter (eip) on this address and harvest two times F8. Now we in the chosen region memory, to **00E90000**. We see that this is the entirely small piece of the code and we can decrease it still more, if we clean from the rubbish and even let us be able to return it in the place.

00E90000		68 FFFFFFFF	PUSH -1
30E90005		50	PUSH EAX
30E90006		6A 00	PUSH 0
30E90008		58	POP EAX
30E90009		64:8B00	MOV EAX,DWORD PTR FS:[EAX]
30E9000C		50	PUSH EAX
30E9000D		81C8 A4FBBEBA	OR EAX,BABEFBA4
30E90013		F2:	PREFIX REPNE:
30E90014	\sim	EB 01	JMP SHORT 00E90017
30E90016		9A 8B44240C 64	CALL FAR 8964:0C24448B
30E9001D		25 00000000	AND EAX,0
30E90022		896C24 ØC	MOV DWORD PTR SS:[ESP+C],EBP
30E90026	\sim	EB 01	JMP SHORT 00E90029
30E90028	-	E9 C1D5ED2E	JMP 2FD6D5EE
30E9002D	\sim	EB 01	JMP SHORT 00E90030
30E9002F		F3:	PREFIX REP:
30E90030		BD 40BC4800	MOV EBP,48BC40
30E90035	\sim	EB 01	JMP SHORT 00E90038
30E90037		9A 03ED8D6C 0C	CALL FAR 0C0C:6C8DED03
30E9003E		2BE9	SUB EBP,ECX
30E90040		F2:	PREFIX REPNE:
30E90041	\sim	EB 01	JMP SHORT 00E90044
30E90043		F0:50	LOCK PUSH EAX
30E90045		C3	RETN
30E90046		0000	ADD BYTE PTR DS:[EAX],AL

But that not to fan even without that large article we will not this make. Let us look, where it is possible to insert this code. The first, that occurs of - this the section of the code (indeed there it and it was earlier). In the diagnostic routine with the dump in the window of the code let us press the key **For end**. Now we at the end the section of the code, see continuous zero. Let us twist window upward, until we meet the first nontrivial bytes.



It means insert the code we will be to **0055C796**. In the window with the protected program let us isolate the code

and let us copy it into the buffer of exchange.

00E9000C 50 00E9000D 91C9 04EBBEBO	PUSH EAX			
00E90013 F2:	PREFIX REPNE:	Backup	•	1
00E90014 V EB 01 00E90016 90 8B44240C 64:	JMP_SHORT_00E90017	Copy	•	
00E9001D 25 00000000	AND EAX.0	Сору	•	
00E90022 896C24 0C	MOV DWORD PTR SS:[ESP+C],EE	Binary	•	Edit
00E90026 ~ EB 01	JMP SHORT ØØE90029	Accomble	Co o co	Fill with ook
00E90028 - E9 CIDSED2E	MP CLOPT OFFORD	Assemble	Space	Fill With OUS
00E9002D × EB 01 00E9002F F3:	PREFIX REP:	Label	;	Fill with NOPs
00E90030 BD 40BC4800	MOV EBP,48BC40	C		
00E90035 V EB 01	JMP SHORT 00E90038	Comment	;	Distance
00E90037 9A_03ED8D6C_0C	CALL FAR 0C0C:6C8DED03	Breakpoint	•	Binary copy
ИИЕЗИИЗЕТ УВЕЗ	I SUB EBPLELIX	prograpoine		

Let us pass into the diagnostic routine with the dump let us press Ctrl+E, Shift+Insert, Enter. Now this code will be placed to 0055C796. It is naturally necessary to change reference to this code. We pass to the address of 004843F8 and instead of 00E90000 let us enter 0055C796:

004843F8 68 96C75500 PUSH DUMPED"3.0055C796 004843FD L. C3 RETN

We again separate entire section of the code and we preserve changes. Is discovered dump. F9. In the line of the state: Access violation when executing [ODEFODOD] - use Shift+F7/F8/F9 to pass exception to program

Search for the constant of 00EF0000.

In the protected program we pass to the address **of 0044CEEB** and it is passed into the chosen memory. This time the code of completely solid size, and, after twisting window downward, we see:

OFFOODD.		20 0E0/EE00	
00EF03FB		68 8E06EF00	FUSH BEFBERE
30EF0400		ES FBFBØDØØ	CALL ØØEDØØØØ
ACCEDIOE.			DOD FOT
00EF 0405			FUF EDI
30EF0406	×	E9 61020000	JMP ØØEFØ66C
30EF040B		5F	POP EDI
30EF040C	×	E9 F4FFFFFF	JMP 00EF0405
30EF0411	Iv.	E9 20020000	JMP 00EF0640
00FF0416	U.	F9 60020000	JMP 00FE0678
AGEFG418	·	C9 0000000	LEOUE
		22 10000000	
JUEFU41C	V	E9 4HU2UUUU	JNP 00EF066B
30EF0421		381E	CMP BYTE PTR DS:[ESI].BL
10FF0423		8945 F8	MOU DWORD PTR SS: FERP-81.FAX
00000402		E7	DUCU EDI
00EF0426		57	PUSH EDI
30EF0427		E8 D4FB0D00	CALL ØØFDØØØØ
OFFO IGO			

Yes \otimes this again VM. To this code it is first necessary to apply script, on [rebildu] VM, and already then to copy into its dump. But, as we remember, to script it is necessary to indicate the new base of the restorable code. [Petomu] first let us be determined, where it will be placed. Let us arise to the beginning of the code and let us call two times by mouse at the intersection of the first column and current line.

\$ ==>	✓CEB 01	JMP SHORT 00EF0003
\$+2	F3:	PREFIX REP:
\$+3	♦FF7424 08	PUSH DWORD PTR SS:[ESP+8]
E 1 7	1 22.00	L DUCUEN

Now the addresses of commands show in the form displacement. Let us twist window, until the code ends.

\$+75E \$+761	8304 00	PUSH EBX
\$+762	^LE9 D7FCFFFF	JMP 00EF043E
\$+767	0000	ADD BYTE PTR DS:[EAX],AL
\$+769	0000	ADD BYTE PTR DS:[EAX],AL
5+76B	0000	I ONN RVTE PTR NS•FFOX1 OI

Leaves that this code it occupies **0x00000766h** of bytes. Let us pass into the diagnostic routine with the dump. Do remember where we they did put the last stolen code? We pass to the address of 0055C796 and let us twist downward, until zero begin:

				2
0550705 0550706		E9 F2	DB F2	
055C7D7	•~	EB 01	JMP SHORT DUMPED"4.0055C7DA	
055C7D9		F0	DB F0	
055C7DA	>	50	PUSH EAX	
055C7DB		C3	RETN	
055C7DC		00	DB 00	
055C7DD		00	DB 00	

Now let us place cursor on 0055C7DC and will call two times by mouse at the intersection of the first column and the display line. We see that also here the addresses became displacement.



ÑÖ.

Ø

\$+821 \$+823

Let us press the key For end and we at the end the section of the code. We look at the displacement of the last byte: DB 00 DB 00 DB 00 ăй

[Khekh]. Place is sufficient, still and it remains. It means, this code we will place to 0055C7DC. In the diagnostic routine with the protected program we start script for [rebilda] VM, and we write to a question about the new base of the code:

00EF0000	✓ EB 01 F3:	JMP_SHORT_00EF0003		
00EF0003	FF7424	OllyScript - Input request		×
00EF0009	52 035424	Enter new base of this code		
00EF000E	BA 1E9E	Enter new base of this code		- 1
00EF0013	88 467D	0055C7DC		
00EF001C 00EF0020	805424 83EA 27	ОК	Cancel	
00EF0023 00EF0027	8D540A 2BD1			
00EF0029	64:EB 0			

We harvest **OK** and await, until script works out. We harvest * and we again on the beginning of the stifling code, as you already know script it throws down the restored instructions at the end of the code. Let us look, they did not exceed the permissible size of the code.

\$+767 \$+76D \$+772 \$+774 \$+774 \$+779	~ ~ ~ ~	0F85 0EFFFFF E9 46FFFFFF 3BF8 0F8D 66FFFFFF E9 4DFFFFFF	JNZ JMP CMP JGE JMP	00EF067B 00EF06B8 EDI,EAX 00EF06E0 00EF06E0	
\$+77A \$+77F	l^	E9 4DFFFFFF 0000	JMP ADD	00EF06CC BYTE PTR	DS:[EAX],AL
	_				

We see that a total of several instructions were added and the code still gets in into the section of the code. We separate entire code and we copy into the buffer of exchange. We pass into the diagnostic routine with the dump, we separate from **0055C7DC** the bytes of the long**of 77F**bytes and we harvest by the right button of the mouse:



pass to 0044CEEB and we correct 00EF0000 to 0055C7DC

We preserve dump, and it is discovered it in the diagnostic routine. F9. They fell here:

			0055CC2C 0055CC2D 0055CC2E 0055CC2F 0055CC30 0055CC30 0055CC31	1E BF 80 00 00 00	DB DB DB DB DB DB DB	1E BF 80 00 00 00		
Line	of	the	state					

Access violation when reading [00C33A29] - use Shift+F7/F8/F9 to pass exception to program

What after...? Judging by the line of state in this place it occurs a attempt at reading to **00C33A29**. We harvest Ctrl+A and immediately gap. We see:

0055CC2B	381E	CMP BYTE PTR DS:[ESI],BL	
0055CC2D	BF 8000000	MOV EDI,80	
00550032 -	0F84 05030000	JE DU62AE~1.0055CF3D	
0055CC38	56	PUSH ESI	
00550039	FF15 D4D25500	CALL DWORD PTR DS:[<&kernel32.lstrlen>]	kernel32.lstrlenA
0055CC3F ~	EB 01	JMP_SHORT_DU62AE"1.0055CC42	
BL =00			

DS:[00C33A29]=???

It turns out that program compares from [nuljom] something, which is stale to **00C33A29**, when there was ASProtect. And you [zamete], that if this something is not equal to zero, then program calculates the length of line, which is been located to **00C33A29**. Nothing it does resemble? Greatly it reminds me of testing the registration code. Give let us try to substitute the address **of 00C33A29** to the address, which it will indicate what or line. Let us open the map of the memory:

		-				
RN3BRRRR [00003000				Priv RW	
00400000	00001000	DU62AE~1			17 10	
00401000	0015C000	DU62AE~1		Actualize		
0055D000	00056000	DU62AE~1				
005B3000	00010000	DU62AE~1		Dump in CPU		
005C30001	000D1000	DU62AE~1	.rsrc	•		
00694000	00003000	DU62AE~1	. mack	Dump		
00697000	00005000	DU629E~1	. sona			
00600000	00005000	0000C///2 1		Search		
00740000	000000000					
001000001	00002000			C		

In the appeared window we write:

HEX+04 29 3A C3 00

And we harvest Enter. Was opened the window of dump even it shows that the address is found.



We see that it is located in the section with the resources to **005B804C**. In the command line we write **with d of 005B804C** and we see.

Address	Hex	: du	IMP														ASC	II				
005B804C	29	3A	<u>C</u> 3	00	1E	00	00	00	17	00	00	00	10	DA	55	00) ; j	.▲.	<u> 1</u>	••••	чr	-
0058805C	10	ин аа	55 00	00	04 02	DН	55 00	00	63	09	55 00	00	00 04	00 00	00 00	00	Pr∟ 8	• • r	0.4	· · · ·	÷.	
005B807C	ŏô	ŏŏ	ŏŏ	ŏŏ	ŏŏ	ŏŏ	ŏŏ	ŏŏ	ŏŏ	ŏŏ	ŏŏ	ŏŏ	ŏó	ŏŏ	ŏŏ	ŏŏ						
005B808C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	÷	• : •	•••		••	
005B809C	02 00	00 00	00 00	00 00	00	00 00	00 00	00 00	04 00	00 00	00 00	00 00	00 02	00 00	00 00	00	e	: *:			ė.	
005B80BC	0ī	00	00	00	04	00	00	00	00	00	00	<u>00</u>	00	00	00	00	0					▼
												7 -										_
: Comman	d d	00(5B8	304)	С						-											

From the address of 005B807C go zero. Let us there and enter our key \odot .A to 005B804C let us correct indicator from 00C33A29 to 005B807C.

Address	Hex du	amp														ASCII	
0058804C 0058805C 0058806C 0058807C 0058808C 0058809C 0058809C 0058809C	7C 80 10 DA 01 00 46 72 00 00 02 00 00 00 01 00	58 50 65 00 00 00 00	00 00 00 00 00 00 00	1E 04 02 57 00 10 00 04	00 DA 00 00 00 00	00 55 00 72 00 00 00	00 00 00 00 00 00 00	17 F8 00 00 00 00	00 D9 00 00 00 00	00 55 00 00 00 00	00 00 79 00 00 00	1C 00 00 00 00 00 02	DA 00 00 00 00 00	55 00 00 00 00 00	00 00 00 00 00 00	AEL ▲ ♥ ▶rU. ◆rU. [⊙] J U. ♥ ♥ FreeWare key ♥ ♥ ♥	
Comman	ic d 00	6B8	040	0						-] [Dia	ddre	ess	— C	Dump at addres	s

Now we separate all this matter and we preserve changes. Is discovered the preserved dump. F9. They fell. In the line of the state:

Access violation when executing [00EE0000] - use Shift+F7/F8/F9 to pass exception to program Search for the constant **of 00EE0000**:

CIIC	COID	canc	OF 0			
004	44CED2		2931A900	JMP	00EE0000	
00.	440002	40			10	

We pass in the protected program to the address **of 0044CED2** and it is passed to **00EE0000**:

0022000	, .	
00EE0000	803D 88DD5800	(CMP BYTE PTR DS:[5BDD88],0
00EE0007	0F95C0	SETNE AL
00EE000A	C3	RETN
1 41 41 1 1 41 41 41 11	12121212	

We see that this entirely [kazjavka]. We insert into the section of the code and we guide adapter.

0055CF5B 0055CF62 0055CF65 0055CF66	803D 88DD5B0 0F95C0 C3 00	CTP BYTE PTR DS:[5BDD88],0 SETNE AL RETN DB 00					

0044CED2 0044CED7	✓ E9 84001100 4C E0	JMP DU62A2~1.0055CF5B DB 4C DB 50					

We preserve changes. Is discovered the preserved dump. F9. Program is loaded..., is loaded..., is loaded. It appeared in [taskbare] and bang... it again fell here:



Again what the [fignja] after the call API of function. Let us try to use the same measures. Instead of the first two bytes after call API let us enter 90 90.

÷	00	ab	0110	<u> </u>	20		••								
	0046	4A69	I. F	F15	D4D25	50	CALL	DWORD	PTR.	DS: D	K&ker	nel32.	lstrlen>]	lstrlenA	
	0046	4A6F	9	0			NOP								
	0046	4870	- 9	0			NOP								
	0046	4871	4	3			INC I	EBX							
	0046	4872	8	DØ41	1B		LEA I	EAX.DWO	DRD F	PTR D	S: LEB	X+EBX3			
	0046	4875	128	ЗСЙ	́ ØЗ		ADD I	EAX.3							
	0046	4978	lż ä	ЗĒЙ.	ĒČ		AND I	FAX. FEF	FFFF	ю. — Э.					
	0046	497B	Ē	8 40	aAFA2A	юİ	CALL	DU6286	Ύ1. P	ю́485	900				

Well here, I so knew. Two-byte instruction is again stolen. But judging from the fact, that after **nop'[ov]** go inc of ebx, the most likely this **mov of ebx, eax**. Let us verify. [Perezapustim] the protected program. Let us pass to the address **of 00464A69** and will install location counter (eip) to this address. We place [brjak] on the instruction after the call API of adapter to the access:

00101000	50		5		
00464A69 00464A6E	E8 92B59E00 66:B8 DB43	CALL 00E50000 MOV AX,43DB	Breakpoint	۱.	Toggle
00464A72 00464A75	8D041B 83C0 03	LEA EAX,DWORD ADD EAX,3	Run trace	•	Conditional
00464A78 00464A7B	83E0 FC E8 400F0200	AND EAX, FFFFFF	New origin here Ctrl+G	rav *	Conditional log
00464H80 00464A82	8804 FF75 08	PUSH DWORD PTF	Go to	•	Run to selection
00464A86 00464A86 00464A89	53 FF75 C0 50	PUSH DWORD PTF	Follow in Dump	+	Memory, on access

We start program. For the first time we are again interrupted not there. Again F9 and it is, where must.



We preserve changes. Is discovered the preserved dump. F9. And... On the miracle!!! Program was neglected! We go in **Help->About Of** fontExpert... and program falls here here:

ronchaper c and	program rarro	HCLC H				
201 996 996 996 996 996	C0005 0000 C0007 0001 C0009 0012 C0008 002400 C000E 0000	ADD BYTE PT ADD BYTE PT ADD BYTE PT ADD BYTE PT ADD BYTE PT ADD BYTE PT	R DS:[EAX],AL R DS:[ECX],AL R DS:[EDX],DL R DS:[EAX+EAX],AH R DS:[EAX],AL			
No-load conditi	on, the same a	lmost 0	0EC0000. Pro	ogram wo	ould fall	as
earlier, but th	is time to 00E	C0000 ra	andomly prov	ved to b	e the cod	le,
and several ins	tructions even	were ca	arried out.	We sear	ch for	
00EC0000 as the	constant:					

0042AAD2	≻ ∍ E9 2955A900	JMP 00EC0000
0042AAD7	B4	DB B4
0042AAD8	8F.	DB 8F
		\sim \cdot \cdot

Well here, I directly prophet O. pass in the protected program to the address of 0042AAD2 and it is passed into the chosen memory. We see that the code not small and is present VM.

LEA EAX,DWORD PTR SS:[EBP-10] 00EC0000 50 PUSH EAX PUSH ØECØ13A 68 3A01EC00 E8 F2FF0A00 5E V E9 BD010000 00EC0009 00EC000E 00EC000F POP ESI Size of the code: 68 DD01EC00 E8 01FE0A00 PUSH ØECØ1DD +1FA CALL 00F70 RETN C3 0000 +1EE+200 ADD BYTE PTR DS:[EAX].AL Plus will be added the restored instructions. Now the code we this matter will not clearly insert into the section. Let us look, how much vacant place in th section e added by us. 0F85 A5BFFFFI E9 B4D1FFFF 0000 069B8FB ADD BYTE PTR DS:[EAX],AL 0069B906 0000 0000 0000 0069B90C * * * 0000 0000 0000 0000 HUD BYTE FIN DSILEAN,IAL ADD BYTE PTR DSILEAXI,AL +6EE \$+6F0 \$+6F2 \$+6F4 +6F6 ōōōō \$+6F8 0000

Place will be sufficient, still and it will remain. We start script for [rebilda] VM:

00EC0003		8D4 50	15	FØ LEA EAX,DWORD PTR SS:[EBP-10] PUSH FAX	
00EC0004 00EC0009		68 E8	3f Fá	OllyScript - Input request	×
00EC000F 00EC0014	č	E9 E9	B[15	Enter new base of this code	
00EC0019 00EC001E	ř	E9 51	85	00698906	
00EC001F 00EC0021 00EC0022		50 50 88	C:	OK Cancel	1 I
00EC0027 00EC002A		C10 8B0)Ø 24		

We harvest Enter and await, until script works out. We harvest * and we on the beginning of region with the stolen code. We separate the code, we copy into the buffer of exchange, troubleshooter with the dump is discovered and we put the code to 0069B906. Now we change [dzhamp] to 0042AAD2 from 00EC0000 to 0069B906.

We preserve changes. Is discovered the preserved dump. F9. Program was neglected! We go in **Help->About Of fontExpert**... This time generally system error was thrown out. Well nothing. Also it is possible much interesting to extract from the errors of system:

\mathbf{J}	
FontExpert 2005	
Подпись ошибки AppName: du72ae~1.exe AppVer: 7.0.0.1 ModName: unknown ModVer: 0.0.0.0 Offset: 00f00100	

Сведения об отчете

We see that the error occurred to **00F00100**, if we are relied on entire obtained by us experience, then possible to assume that program turned to **00F00000**, and by it transported so, that it it could reach [azh] the hundredth displacement. Let us verify theory. We search for **00F00000** as constant. No, unfortunately this time I was mistaken. Then we enter on - to other. To [perezagruzim] program let us place [brjak] on last restored [dzhamp].

 We start program. We go in Help->About Of fontExpert... They interrupted on our [brjake]. We begin to locate and we fall here:

 We start program. We go in Help->About Of fontExpert... They interrupted on our [brjake]. We begin to locate and we fall here:

 00440F39F
 B8

 00440F44
 B8

 00440F74
 B8

 005
 DB

 005
 DB

 005
 DB

 005
 DB

 006
 DB

 007
 B8

 008
 B8

 008
 B8

 00440F78
 F

 008
 B8

 008
 B8

 008
 B8

 008
 B8

 008
 B8

 008
 B8

recognized:

We go	o in	the	0044CF9F 0044CFA4 0044CFA9 0044CFA9 0044CFA9 functio	B8 0C005500 E8 4F740300 51 8365 F0 00 On On F7.	MOV EAX, DU72AE"1. CALL DU72AE"1.004 PUSH ECX OND DWORD PTR SS. We see the	0055000C 843F8 FFRP-101 0 already	restore	d adapter:
			004843F8 004843FD 004843FE	68 96C75500 C3 E8 7B892F31	PUSH DU72AE"1.005 RETN CALL 3177CD7E	50796		
We ha	arves	st tv	vo times	s F8.				
			00550796 00550798 00550790 00550795 00550795 00550795	68 FFFFFFF 50 6A 00 58 64:8800 50	PUSH -1 PUSH EAX PUSH Ø POP EAX MOV EAX,DWORD PTR PUSH EAX	FS:[EAX]		
This	is t	he r	restored	d previous	sly code. We	locate	, until v	we leave

the function. They left here:

0044CFA9	51	PUSH ECX
0044CFAA	8365 FØ ØØ	AND DWORD PTR SS:[EBP-10],0
0044CFAE V	E9 01000000	JMP_DU72AE"1.0044CFB4
0044CFB3	19E9	SBB ECX,EBP

[Dotrassiruem] to the instruction **of jmp** let us carry out it. Burn here:

00440EB4 - E9 4730AB00 JMP 00F00000 0044CFB9 9E SAHF 0044CFB9 1858 06 SBB BYTE PTR DS:[EAX+6],BL

[Oppaaa]... But indeed we searched for the constant of 00F00000! Yes, I and itself was strongly astonished. It leaves, it does not be worthwhile to separately entrust to the mechanism of search into OllyDbg. Let us pass to this address in the protected program and will visit into the chosen region memory. The code of small and again is present VM.

F= 0000		
\$+7C	68 6B00F000	PUSH 0F0006B
\$+81	E8 7AFF0F00	CALL 01000000
\$+86	C3	RETN
\$+87	8B4D F4	MOV ECX, DWORD PTR SS: [EBP-C]
\$+8A	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]
\$+8D	64:890D 000000	(MOV DWORD PTR FS:[0],ECX
\$+94	C9	LEAVE
\$+95	~ E9 ECFFFFFF	JMP 00F00086
\$+9A	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]
\$+9D	8365 FC 00	AND DWORD PTR SS:[EBP-4],0
\$+A1	8D45 F0	LEA EAX,DWORD PTR SS:[EBP-10]
\$+A4	^ E9 D2FFFFFF	JMP 00F0007B
\$+A9	0000	ADD BYTE PTR DS:[EAX],AL
210D	0000	LODD DUTE DTD DC-FEOVIL OL
- oarl	ior To ho	located this code in me will be to

All, as it is earlier. To be located this code in me will be to **0069BB06**, it means:

00F00000 00F00005		68 B82AF0C3	PUSH_C3F02AE PUSHEM	38	
00F00007 00F00008		OllyScript - I	Input request		\ge
00F00009 00F0000B	~	Enter new ba	ase of this code		
00F00013 00F00016		00698806			1
00F0001C 00F0001E 00F00022	v	ОК		Cancel	
00F00024 00F00025		0005			3

Script worked out. We copy the code in the buffer of exchange and put to 0069BB06. We change [dzhamp] to 0044CFB4 from 00F00000 to 0069BB06. We preserve changes. Is discovered the preserved dump. F9. Program was neglected! We go in Help->About Of fontExpert...

	oxima		
E.	FontExpert Version: 7.1 Copyright [@] License Ke FreeWar	2005 00 Release 1 > 1999-2005 Proxima Software y: e kev	OK.
	Web Site:	http://www.ProximaSoftware.com Since 1991 on the market of font tools.	

[UraAAAaa]! Program earned. First test of - closing program. I noted that with this ASProtect'[om] in essence two problems. These are the window Of about and the correct completion of the work of program. We shut program. Well, I so thought - program it fell here:

7C9012AF	83C9 FF	OR ECX,FFFFFFFF
7C9012B2	3300	XOR EAX,EAX
7C9012B4	F2:AE	REPNE SCAS BYTE PTR ES:[EDI]
7C9012B6	F7D1	NOT ECX
7C9012B8	81F9 FFFF0000	CMP ECX.0FFFF
7C9012BE	√ 76 05	JBE SHORT ntdll.7C9012C5
7C9012C0	B9 FFFF0000	MOV ECX,0FFFF
7C9012C5	66:894A 02	MOV WORD PTR DS:[EDX+2],CX
7C9012C9	49	DEC ECX
7C9012CA	66:890A	MOV WORD PTR DS:[EDX],CX
7C9012CD	5F	POP EDI
7C9012CE	C2 0800	RETN 8

This is the system library of ntdll. But how to us to find where does fall? Let us look, into the window of stack, there must be the address of recovery from the function into the program.

1	0012EEB0 0012F1DC		
	0012EEB4 7C9135A4	RETURN to ntdll.7C9135A4 from nto	<u> - </u>
	0012EEB8 0012EEC0		
1	0012EEBC FFFFFFFF		
	0012EEC0 00000000		
	00120004 FFFFFFF		
	0012EEC0 0012EEEC	RETURN to advanti32 77DEC168 from	nto
	0012FED01 0012FEF0	Meronal to advaptoz. Trbectob from	HVC .
	0012EED4 FFFFFFF		
	0012EED8 00000103		
	0012EEDC 00000001		
	0012EEE0 0012F024		
	0012EEE4 005543F0	DUE013~1.005543F0	
	0012EEE8 00000000		
	0012EEEC 0012F080	DETUDN to DUE010"1 OGEODOED (man	a at I
	0012EEF0 0053D65D	RETURN to DUEUIS 1.0055D65D From	adv
1	0012EEF8 FFFFFFF	110011.10902210	I I
L			
Let us look, that	t we have to	0053D85D:	
00530852	FE30 PUSH	NUORD PTR DS: FEAX1	Subke

What for [fignja] with API by functions is today created? I see for the first time so that after the call API of function would be conditional jump back to the call API! This smells a little by already infinite cycle! I make the assumption that two bytes, after the call API of function are stolen By aSProtect'[om], and instead of them is inserted this insidious [dzhamp]. Here here simple so to surmise, that after command it is stifling it will not succeed. Let us try to [pomeditirovat]. Let us open diagnostic routine with the protected program, let us reach OEP and will pass to the address of 0053D857, let us install to it location counter (eip) and let us place [brjak] on several bytes after call API to the access to the memory. We harvest F9. For the first time not there, let us pass, but they landed by the here second in the section of the code.

0053085F FFB5 DCFEFFFF PUSH DWORD PTR SS:[EBP-124] 00550865 E8 96279100 CALL 00550000

Thus, now let us open diagnostic routine with our dump and also will pass to the address of 0053D857. Let us carry out the call API of function (F8). But now let us compare registers in the protected program and in our dump.

Protected program	Our dump
EAX 00000006 ECX 7C91056D EDX 00030000 EBX 3D83D88F ESP 0012FF64 ESP 0012FF64 ESI 00000006 EDI 00E3030D	EAX 0000000 ECX 0012FF80 EDX 7C90EB94 r EBX 7FF06000 ESP 0012FFC4 EBP 0012FFF0 ESI FFFFFFFF EDI 7C910738 r

The function Of regDeleteKeyA returns result to the register of eax, and in the protected program the values of eax and esi are equal. But the in the manner that stolen command has already been carried out the, it can be assumed that stolen command it is **mov of esi, eax**. We substitute in the dump conditional [dzhamp] by **mov of esi, eax**, we preserve changes and is discovered the preserved dump. We start program and attempt to shut. Program fell here:

004C93FE . 6A 0:	1 PUSH 1		
004C9400 . FF12	68 MOULECY	WORD PTR DS:[EDX]	,
DS: [586E90001=222	100 201		=
B0.1000290001=:::			

We see that the program attempts to fulfill the function, whose address is located into edx. But this address indicates generally beyond the limits of memory. Let us look, which occurs in this place in the protected program.

-		
004C93FC 8B11	MOV EDX,DWORD PTR DS:[ECX]	
004C93FE 6A 01	PUSH 1	
004C9400 FF12	CALL DWORD PTR DS:[EDX]	FONTEX~1.00508D30
00400400 0045 00	MOULEON DUODD DTD DO FEOT.COT	
DS:[00586E90]=00508D30	(FONTEX"1.00508D30)	

It cannot be! The number into edx is also the very, but is shifted by two discharges to the right! I searched for here this error very for long. And it was already it solved, that this is mysticism and maltsters were accepted to itself to the service of sorcerer. But I solidly understood that never he is worthwhile to despair. How I did localize this [bag]? If you [potrassiruete] a little program, then you will understand because of what Of [ares] it is shifted by several discharges. In reality it is not shifted. The address falls as follows in the register of edx:

004C93FC 8B11 MOV EDX, DWORD PTR DS: [ECX]

Ecx is equal **003C84F7** most likely precisely this indicator to what that by means it decreases by one, indicating no longer the beginning of address. I decided to search for the place, where to

003C84F8 will be brought in th address e necessary for the work of program. In the command line we write **with d of 003C84F8**, harvest Enter and we see th address e interesting us. Let us place on it hardware of [brjak] to the record, with the size of dword (4 bytes). To place is necessary precisely hardware of [brjak], since after reloading of program it will still act.

Address	Hex dump					ASCII	▲	0012F
003C84F8 003C8508 003C8518 003C8518 003C8528 003C8528 003C8548 003C8548	90 6E 52 00 00 00 00 04 18 57 00 00 00 00 00 00 00 00 00	Backup Copy Binary	00 00 00 • •	90 00 00 10 00 00 10 00 00 10 00 00 10 00 00 10 00 00	00 00 00 00 00 00 00 00 00 00 00 00	PnX.\$+₩.8. 0. ++₩.x+₩.		0012F 0012F 0012F 0012F 0012F 0012F 0012F 0012F
00308578	00 00 00	Breakpoint Securb for	`	Memory	r, on ac 	1855 ita		0012F
003C8598	00 00 00	Search for	in Dump	memory	, on wr			0012F
003C85A8 003C85B8	00 00 00		unoanip ••••••••••••••••••••••••••••••••••••	Hardwa	are, on a	access 🔹 🕨		0012F
003C85C8 003C85D8	00 00 00 00	30.00		Hardwa	are, on v	vrite 🕨 🕨	Byte	e
003C85E8 003C85F8	00 00 00 🗸	Hex	•	Hardwa	are, on e	execution	Wor	d
003C8608 003C8618	00 00 00 00 00 00	Text	•	0 00 00	00 00 00 00		Dwo	ord
[Perezapuskaem] progr	am we ha	arvest	F9. Tł	ney	stopped	her	e:
70 70 70 70	91152A 89 091152A 89 091152C 89 091152F 57	950 04 902 941 04	MOV DWORL MOV DWORD MOV DWORD PUSH EDI) PTR DS:L) PTR DS:C) PTR DS:C	EHX+4J EDXJ,E ECX+4J	,EDX AX ,EAX		
We look into t	he wind	ow of th	ne dump	:				
Addres 003C84 003C85 003C85 003C85 003C85	Hex dump 4F8 78 01 30 508 60 60 60 518 60 60 60 528 60 60 60 528 60 60 60 538 60 60 60) 2 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	ASCII 00 x0< 00 00 00 00		
No, this is wh	at that.	another	addre	ess. Ag	gain	F9, the	ey s	topped
here:								
00 00 00	3507E5A 3507E5F 3507E65	E8 5C670100 C707 086C580 8D5E 54	MOV DWORD	013~1.0051) PTR DS:[)WORD PTR	ESBB EDIJ,D DS:[ES	UE013~1.005 I+54]	8600:	
We look into t	he wind	ow of th	ie dump	:				_
Addres 003C8 003C8 003C8 003C8	ss Hex dump 4F8 90 6E 58 508 00 00 00 518 00 00 00	0 3 00 78 01 30 3 00 00 00 00 00 3 00 00 00 00 00	; 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00	ASCII 00 PnX.x0<. 00		_
Yes, this is o	ur addr	ess. We	look i	nto th	ne wi	indow o	f th	.e
registers:			_					
			Registers EAX 0012F	(FPU) 6CC				



We see that into esi lies the normal unspoiled indicator. Let us remove hardware of [brjak] from the indicator so that it would not interfere.

•



But now we will locate the code on F8 and follow the register of esi. We see this place:

00507EDDI		89BE	A400000	(MOV.	DWORD	PTR [DS:EE	SI+A4	J.EDI				
00507EE3		FF15	28D4550	CAL	L DWORD) PTR	DS: D	<&ker	ne132.	LGe	tModu	(LeHan)	dleA
00507EE9	•	F9		STC									
00507EEA	•	4E		DEC	ESI								
00507EEB	•	02F1		ADD	DH,CL								
00507EED	•	68 74	46E5800	PUSI	H DÚE01	13~1.0	<i>3</i> 0586	E74		[CPr	ocNar	eOrOr	dinal
00507EF2	•	50		PUSI	H EAX					hM	odule	2	
00507EF3	•	FF15	F4D2550	CALI	L_DWORD) PTR	DS: D	<&ker	nel32.	(LGe	tProc	Addre	ss
					1	- T- T		<u> </u>					

This that still such after the call **Of getModuleHandleA**? Here, that tells me my scant knowledge of [asma]: STC advances flag [S]F in one, then DEC ESI decreases the indicator by th address by one e interesting us!!! But are further generally no one not necessary operations. Is similar this the again stolen instructions. As we earlier substitute two bytes after the call API of function by 9090. To [khe]! But indeed the following instructions did not recognize! It leaves, that ASProtect of [sper] of here 4 bytes. Now let us look to the logic of the work of program. The function **Of getModuleHandleA** calculates [khendel] of necessary module, unknown 4 bytes further go, into the stack is placed the indicator to the name of function, then into the stack is placed [khendel] obtained from **GetModuleHandleA**. However, that they could make these 4 bytes, indeed in principle they here were not necessary! Give [zabjom] by their **nop'[ami]** and the case is closed ©.

 00507EE3
 .
 FF15
 28D4550
 CALL
 DWORD
 PTR
 DS: [<&kernel32.(</th>
 GetModuleHandleA

 00507EE3
 90
 NOP
 We preserve changes and is discovered the preserved dump. We start program and attempt to shut.

Process terminated, exit code 0

All! We completely unpacked program! But there is one [bag]. I revealed him after [inlajn] of [patcha]. About it I will describe in chapter about [inlajn] [patche].

As you understood, in this program I did not in vain give so much attention to the stolen code after the calls API of functions. However, what this? But this nothing else but second VM Of aSProtect'[a]. I call its VM API. You do not relate to it disdainfully, since it emulates many commands. It can emulate even call of function after call API!

You do remember the place, where we did replace 4 bytes with nop'[ami]? It is so here there stolen:

Cmp of eax, edi je 00507EFF

It is strong? In this VM is much more powerful the mechanism of checking the integrity of the code. I began to write script on its restoration, but thus far it far to [reliza]. Most frequently ASProtect do not use this VM with the protection; therefore you can never not meet with it. So in the program can be encountered checkings to [raspakovannost] with the aid of the special macros. Therefore I consider that ASProtect it is necessary to [patchit], but not to unpack. So it is much safer. I never made [inlajn] of [patch] (in me it was another technology, but not [loader]), but in this version I it decided nevertheless to try.

We write [inlajn] of [patch].

After studying, a article Of alex'[a] about [inlajn] of [patch] Of aSProtect 2.0, I decided to make a little differently. Theory is the same: ASProtect has multilayer structure, i.e., with the starting it it unpacks its body in parts into chosen for this regions memory, which hampers its [propatchivanie]. In order to [propatchit] program, it is necessary to follow ASProtect'[om] into these regions of memory before the complete unpacking of program, to and then [propatchit] and program. But complexity consists also in the fact that the passages into the following region of memory are encoded and are deciphered dynamically, in proportion to the fulfillment of the code. Let us begin from the fact that we will not search for place for the arrangement of our [patcha]. Alex wrote that ASProtect cleans the code On the Border of sections, moreover several times. Let us consider this and will place the code between the end PE of title and the beginning of the first section. Let us attempt to trace, where occurs the first passage Of aSProtect'[a] into the chosen memory. We place [brjak] on VirtualAlloc of [zhmjom] F9, we are interrupted and we pass to the address of recovery.

006944A0	6A 00	PUSH 0	
006944A2	FF95 F0030000	CALL DWORD PTR SS:[EBP+3F0]	
006944A8	8985 CC010000	MOV DWORD PTR SS:[EBP+1CC],EAX	
006944AE	8B9D 00040000	MOV EBX,DWORD PTR SS:[EBP+400]	
006944B4	039D 0D040000	ADD EBX,DWORD PTR SS:[EBP+40D]	
006944BA	50	PUSH EAX	
006944BB	53	PUSH EBX	
006944BC	E8 04010000	CALL FONTEX"1.006945C5	
006944C1	6A 40	PUSH 40	
006944C3	68 00100000	PUSH 1000	
006944C8	FFB5_08040000	PUSH DWORD PTR SS:[EBP+408]	
006944CE	68 00	PUSH 0	
00694400	FF95 F0030000	CHLL DWORD PIR SS:LEBP+3F01	

Push 0 is the passage into the chosen region of memory. Thus far there 0, but if we a little [potrassirovat], then it is possible to find the place, where the address of passage is written instead of zero. Here is it:

0069457D 0069457E 00694581	5E 8B46 04 03C7	PUP ESI MOV EAX,DWORD PTR DS:[ESI+4] ADD EAX,EDI
00694583	8985 C7010000	MOV DWORD PTR SS:[EBP+1C7],EAX
00694589	8B55 5B	MOV EDX.DWORD PTR SS:[EBP+5B]
0069458C	8B85 C7010000	MOV EAX DWORD PTR SS:[EBP+1C7]
00694592	8942 ØC	MOV DWORD PTR DS: [EDX+C].EAX
00694595	8D9D 0D040000	LEA EBX, DWORD PTR SS: [EBP+40D]
0069459B	53	PUSH EBX

It is here necessary to enter [dzhamp] to our [patch]. But in the manner that this [dzhamp] will rub over the original code, then in [patche] it is necessary to first carry out the rubbed over code, to and then already [patchit] other addresses. It is reloaded program and we pass to address **00694583** and it is seen:

00694583	6F	OUTS DX,DWORD PTR ES:[EDI]	I/O command
00694584	A5	MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[E	
00694585	26:AB	STOS DWORD PTR ES:[EDI]	
00694587 -	E9 28E59602	JMP 03002AB4	
0069458C	BB E878E728	MOV EBX.28E778E8	
00694591	CE	INTO	

However, where be divided the instructions, which we did see here past time? But these are they and there is, only encoded are deciphered by gradually several decoders. Now there are two versions:

To enter so as Alex. To write [dekriptor], which will decipher this code and if it deciphers correctly, then write [kriptor] and encode by them our code, and then replace with them original. For the application of this method it is necessary to find all decoders of this code, to clean of the rubbish and to try to write its decoder. To find the code, which deciphers this place, to [perezagruzit] program. To pass to the address of decoder and if there rubbish, then find the decoder of this decoder, otherwise enter passage to our [patch]. And so until we find very first decoder. I selected the second version. We place [brjak] on address **00694583** to the record and start program.



F8 and we look that with the address interesting. Not it is [raskriptovan]? Again F9, F8. And so until instructions on address

00694583 are restored. After the fulfillment of [instuktsii] of mov in this place:

T		
00694368 0069436E	81C3 19AB5007 8918	ADD EBX,750AB19 MOV DWORD PTR DS:[EAX].EBX
00694370	BF 626C254B	MOV EDI,48256C62
00694375	81E8 B07E6C03	SUB EAX, 36C7EB0
0069437B	66:BA DCD5	MOV DX,0D5DC
0069437F	81C0 AC7E6C03	ADD EAX, 36C7EAC
00694385	- E9 10000000	JMP_FONTEX"1.0069439A
0069438A	C8 618647	ENTER 8661,47
0069438E	• 74_9D	JE_SHORT_FONTEX"1.0069432D
00694390	12E3	ADC AH,BL
NN6943921 ^	• EN 99	LUUPUNE SHURT FUNTEX 1.0069432D

We see that th instructions on address **00694583** e interesting us are completely [raskriptovany]:

-	CUCLY	LTABUTTACO	vanyj•	
	Address	Hex dump	Disassembly	Comment 🔺 🔺
	00694583	8985 C7010000	MOV DWORD PTR SS:[EBP+1C7],EAX	
	00694589	8855 58 0005 07010000	MOV EDX,DWORD PTR SS:[EBP+5B]	
	00694592	8942 ØC	MOV DWORD PTR DS:[EDX+C].EAX	
	00694595	8D9D 0D040000	LEA EBX, DWORD PTR SS: [EBP+40D]	
	0069459B	53	PUSH EBX	
	0069459C 0069459E	60 00		
	006945A0	6A 01	PUSH 1	
	006945A2	57	PUSH_EDI	
	006945A3	885E Ø8	MOV EBX,DWORD PTR DS:[ESI+8]	
	00694588	53	PUSH EBX	-
ł				
	Common	d 00694583	D address – Dump at a	ddress
	Comman	14 00034000	The database builty at a	

It leaves, that decoder with address **0069436E** latter and after it it is possible to [patchit] the code with address **00694583**. By [perezapustim] program let us pass to the address of the last decoder:



We see that the instructions of decoder are also [poshifrovany] and are deciphered by another decoder. On the whole we repeat all previous actions, until decoder deciphers. Then we pass to the address of the decoder, which deciphers this decoder even we look, that there. If bytes are there [poshifrovany], we again repeat all actions until we reach the decoder, which not is [pokriptovan] in the packed program. In me this chain of the decoders (in this sequence they they decipher each other) came out: **69415A - > 69420E - > 6942D 3-> 69436E - > 694583**.

It is now necessary to study the first decoder and to find in it such place, with passage of which the following decoder will be already [raskriptovan]. This is how the first decoder appears:



A little [potrassirovav] it it is possible to understand, that this is cycle, and that after working out it will pass to **00694187**. It means with this address necessary to place jmp on our [patch]. Since this jmp will rub over the command of sub of ecx, **2c0BD000** i.e. will have to carry out in our [patche], then to [propatchit] the following decoder and to pass to address **0069418D** for continuing the normal operation of program. I decided not to [patchit] this code statically (i.e. immediately in the packed file), but to [propatchit] only with the starting of program. Will look as begins the work Of aSProtect:



With address 00401000 it will bring in into the stack address 00694001. Through this address is located the body of [raspakovshchika]. If we will replace this address by the address of our [patcha], then ASProtect with the starting immediately will pass to our [patch]! Now let us calculate the address of our [patcha]. As I already spoke write him will be between the end PE of title and beginning of the first section. Let us open the map of memory even we see that PE the title begins with 00400000, and the beginning of the first section with 00401000:

	00400000 00001000 F 00401000 0015C000 F 0055D000 00056000 F 00553000 00010000 F 00553000 0001000 F 00694000 00039000 F 00650000 00001000 F 0066D0000 00001000 F	ONTEX"1 PE ONTEX"1 cod ONTEX"1 dat ONTEX"1 .rsrc res ONTEX"1 .data imp ONTEX"1 .adata	header Imag R le Imag R a Imag R ources Imag R orts,rel(Imag R Imag R Imag R Map R E	RWE RWE RWE RWE RWE RWE RWE RWE R	
In the co	mmand line we	drive into d	. 00400000 and	we see:	
	Address Hex dump		ASCI	I	
	00400000 40 54 90 00 00400020 00 00 00 00 00400020 00 00 00 00 00400030 00 00 00 00 00400030 00 00 00 00 00400050 69 73 20 70 00400050 69 73 20 70	0 03 00 00 00 00 04 00 0 0 00 00 00 00 40 00 0 0 00 00 00 00 00 00 00 0 00 00 00	90 00 FF FF 00 00 M2P, 90 00 00 00 00 00 9 90 00 00 00 00 00 90 00 28 01 00 00 91 4C CD 21 54 68 ∦♥ 20 63 61 6E 6F is r 5F 20 44 4F 53 20 t. be	● .(0. 1.1.=!10L=!Th program canno > run_in_DOS	
	Commanc d 400000		•		
This DE t	he title of in	the form he	🗕 xadecimal dum	n In OllyDbo	r i t
ia nogait	le to look PF	title in the	more conveni	p. III OIIyDDg ent idea For	thia
	vint out to di:	anostia rout	ine that this	DF +i+la	CIILD
iet us pe		Disassem		FE CICIC.	
		Special		PE beader	
	Oddress Hey dump	bocidi	· · · · · · · · · · · · · · · · · · ·		
	06400000 40 5A 90 00 0 00400010 B8 00 00 00 0 00400020 00 00 00 00 00 00400020 00 00 00 00 00 00400020 00 00 00 00 00 00400040 0E IF BA 0E 00400050 69 73 20 70	33 00 00 0 Appearai 30 00 00 00 00 00 00 30 00 00 00 00 00 00 00 30 00 00 00 00 00 00 00 30 00 00 00 00 00 00 00 30 08 00<	nce 00 00 00 00 00 00 00 28 01 00 00 4C CD 21 54 68 87∥8 63 61 6E 6E 6F is pr		
	Commanc d 400000		•		
Well here	, everything w	was converted	into the spe	cial structur	es:
	Address Hex dump	Data	Comment	▲	
	004000002 40 5A 00400002 9000 00400004 0300 00400006 0000 00400008 0400 00400008 0400 00400008 0400 00400008 0000	ASCII "M2" DW 0090 DW 0003 DW 0000 DW 0004 DW 0004	DOS EXE Signature DOS_PartPag = 90 (1 DOS_PageCnt = 3 DOS_ReloCnt = 0 DOS_HdrSize = 4 DOS_MinMem = 0	44.)	
As you ce	ertainly know t	that the last	structures P	E of title ar	e the
descripti	ons of section	ns. We search	for the desc	riptions of t	he
sections	(we twist wind	dow downward)	:		
	Address Hex dump	Data Du soco	Comment		
	00400220 00400222 00400224 00400228 00400228 22 61 64 6 00400270 00400200 00400229	DW 0000 DD E000040 ASCII ".adata" DD 00001000 DD 002CD000	NumberOfLineNumbers Characteristics = I SECTION VirtualSize = 1000 VirtualAddress = 2C	= 0 = 0 NITIALIZED_DA (4096.) DA00	
	004002F8 00000000 004002FC 008C1000 0040026C 008C1000 00400304 0000000 00400308 0000 00400308 0000 00400308 0000 00400308 0000	DD 00000000 DD 00108C00 DD 00000000 DD 00000000 DD 00000000 DW 0000 DD 0000 DD 0000 DD 0000	SizeOfRawData = 0 PointerToRawData = PointerToRelocation PointerToLineNumber NumberOfLineNumbers Characteristics = I	108C00 5 = 0 5 = 0 = 0 = 0 NITIALIZED_DA	
	00400311 00 00400312 00 00400312 00	DB 00 DB 00 DB 00			

We see that after the address of 0040030[S] the description of sections conclude and begin zero. I decided to write [patch] from address 00400350, suddenly he will be required to write any data (name of user for example O). Let us approach. The beginning Of aSProtect'[a] let us change thus:

00401000	68 50034000	PUSH FO0F53″1.00400350 🗲
00401005	E8 0100000	CALL F00F53~1.0040100B
0040100A	C3	RETN
0040100B	C3	RETN
0040100C	93	XCHG EAX,EBX

Now let us pass to address **00400350**. Here we should [propatchit] the first decoder. Let us write for this

MOV BYTE PTR DS:[694187], 0E9.

This command will prescribe with the address **00694187** [opkod] of the instruction **of jmp**. It is further necessary to enter after [dzhampa] the address, to which ASProtect will pass after the decoding of the second decoder. Thus far, that we do not have function, whose address must be entered; therefore let us enter for the moment any value **MOV DWORD PTR DS:[694188], 40404040** after this it is necessary to pass to that address, where ASProtect would pass, if we did not change its beginning. We recall what address it was after the command of push we at first and write **WITH JMP 00694001**. It is now necessary to write the function, to which will pass ASProtect after the decoding of the second decoder. You do remember that the command **of sub of ecx, 2c0BD000** will be rubbed over by [dzhampom]? Therefore let us carry out first it, and then let us return to the decoder. All this let us write after JMP **00694001**:

00400350		C605 87416900	E9
00400357		C705 88416900	4046
00400361	-	E9 9B3C2900	
00400366		81E9 00D00B2C	
0040036C	-	E9 1C3E2900	

and

MOV BYTE PTR DS:[694187],0E9 4040 MOV DWORD PTR DS:[694188],40404040 UMP F00F5311.00694001

JMP F00F53~1.00694001 SUB ECX,2C0BD000 JMP F00F53~1.0069418D

Now we know the address of the following function of our [patcha] (00400366) and we can enter him instead of 40404040 with address 00400357 but not all so simply. It is there necessary to enter displacement relative to address 00694187. You will look as it appears jmp 0069418D to 0040036[S]:

E9 1C3E2900. E9 - this [opkod] of [dzhampa], and **1C3E2900** (in the machine idea) = **of 00293E1C** (in the human) = **0069418D-0040036C-shch**. I.e. this nothing else but displacement. In order not to suffer and not to calculate these displacement let us pass to address **694187**, let us enter there **imp 00400366**

		51	
0069417B	81FA 9CF8FFFF	CMP EDX764	
00694181 ~	0F85 1900000	JNZ F00F53~1.006941A0	
00694187	81E9 00D00B2C	SUB ECX,2C0BD000	
0069418D	· E9 1F000000	JMP F00F53″1.006941B1	
00694192	· 7E DE	ILE CHOPT FOREFORT RECORDS	
00694194	2C_F5	Assemble at 00694187	X
00694196	SHFB		
00694198	1871 56		
00694198	07 CAOD E27201D9	jmp 00400366	▼
00694190	49 E2730109		
00694103	FF		
00694104	78 64		
00694106	F9 49FFFFFF	Fill with NOP's	Assemble Concel
006941AB	06		Assemble Cancel
- Contractions			1
let us l	look as [Olli] :	it assembied the co	mmand:
	00694181 ∨ 0⊦85 19	000000 JNZ FU0F53"1.	006941H0
	00694197 - F9 DOC1	DAFE	00400366

Well it is here and there is necessary displacement. Let us enter it instead of **40404040** (to enter necessary in "human" idea! **FFD6C1 DA** why thus? [Izuchi] the basics of assembler). Well here we already have a beginning of [patcha]:

0069418C 90 0069418D v E9 1F000000 NOP JMP F00F53~1.006941B1

00400350	C605 87416900 E9	MOV BYTE PTR DS:[694187],0E
00400357	C705 88416900 DAC1D6FF	MOV DWORD PTR DS:[694188],FI
00400361 -	E9 9B3C2900	JMP F00F53~1.00694001
00400366	81E9 00D00B2C	SUB ECX,2C0BD000
0040036C -	E9 1C3E2900	JMP FO0F53~1.0069418D

D6C1DA

After the second decoder will be completely decoded ASProtect it will fall into the function with address **00400366**, where the rubbed over command will be carried out and control will pass again to the decoder (well this only thus far we we will not find the place, where it is possible to [propatchit] the second decoder.). We place [brjak] on**0040036C**and start program. They interrupted, it means the second decoder it is deciphered. Let us look how it appears:



[Potrassirovav], we understand, that this again cycle and after finalizing control will pass to address **00694228**, it means instead of [dzhampa] with address**0040036C**necessary to write commands, [propatchivajushchie] instructions on address **00694228**. But after them to carry out the instruction **of mov si, cx**, to [propatchit] the third decoder and to return the secondly. To more shortly repeat all that the fact that they made with the previous decoder. In me this code came out:

00400350	C605 87416900 E9	MOV BYTE PTR DS:[694187],0E9
00400357	C705 88416900 DAC1D6FF	MOV DWORD PTR DS:[694188],FFD6C1DA
00400361 -	E9 9B3C2900	JMP_F00F53~1.00694001
00400366	81E9 00D00B2C	SUB ECX,2C0BD000
0040036C	C605 28426900 E9	MOV BYTE PTR DS:[694228],0E9
00400373	C705 29426900 55C1D6FF	MOV DWORD PTR DS:[694229],FFD6C155
0040037D -	E9 0B3E2900	JMP_F00F53~1.0069418D
00400382	66:8BF1	MOV SI,CX
00400385 -	E9 BE3E2900	JMP F00F53~1.00694248
ropost	over thing Me	$n \log \left(\frac{1}{2} \right)$

Further we repeat everything. We place [brjak] on **00400385** F9 they interrupted. The third decoder is deciphered. We look, that there:

		± ,
006942D3	8F043B	POP DWORD PTR DS:[EBX+EDI]
006942D6	B5 21	MOV CH,21
006942D8	BA B2DA4D62	MOV EDX.624DDAB2
006942DD	83EF 01	SUB EDI.1
006942E0	81E2 755EE474	AND EDX. 74F45E75
006942E6	4F	DEC EDI
006942E7	4F	DEC EDT
006942F8	4F	DEC EDI
006942F9	BE E1603952	MOU EST. 52396CE1
006942FF	81FF FCF9FFFF	CMP_EDL_=614
006942F4	ØF85 SOFFFFF	IN7 E00E53"1 00694284
00694250	P0 62549500	
000042FH	DH 02340E0H	
006942FF	SE	FUP ESI

Entire also the very upon transfer to **006942FA** the fourth decoder will be completely decoded. I think and here you will manage themselves:

00400361 -	E9 9B3C2900	JMP_F00F53~1.00694001	
00400366	81E9 00D00B2C	SUB ECX,2C0BD000	
0040036C	C605 28426900 E9	MOV BYTÉ PTR DS:[694228],0E9	
00400373	C705 29426900 55C1D6FF	MOV DWORD PTR DS:[694229],FFD6C155	
0040037D -	E9 0B3E2900	JMP F00F53~1.0069418D	
00400382	66:8BF1	MOV SI,CX	
00400385	C605 FA426900 E9	MOV BYTE PTR DS:[6942FA],0E9	
0040038C	C705_FB426900_9CC0D6FF	MOV DWORD PTR DS:[6942FB],FFD6C09C	
00400396 -	E9 AD3E2900	JMP_F00F53~1.00694248	
0040039B	BA 62548EØA	MOV EDX,0A8E5462	
004003A0 -	E9 5A3F2900	JMP FO0F53~1.006942FF	
We pla	ace [briak] on	004003A0 and start program	The

We repeat. We place [brjak] on **004003A0** and start program. The fourth decoder is deciphered. We look that there:



But here here increasingly more complex. Cycle after finalizing will pass to the address of 006943A3 but as it is seen the there conditional jump, which it did not be desirable [ba] to rub over, since it it is sufficiently complicated to carry out in [patche] (in the plan of the size of [patcha]). Well it is good. Then it is possible to rub over the following command (although known will not be carried out the conditional jump or not). But here here that exactly and problem. The fact is that this last decoder will be decoded not to the end and deciphers not only the code, which to us is necessary, but also itself. All which is located below address of 006943A3 encoded it will decipher only during the work of this decoder. Then we make thus. Let us replace this passage to our, and in the function of our [patcha], to which will pass ASProtect after the interpretation of th code e interesting us we let us restore this conditional [dzhamp] let us return to it.

We place [brjak] on **004003C7** and start program. Everything! The interesting us code with address **00694581** is completely [raskriptovan], the thanks to you industrious decoders, [propatchivat] it we will be thus:

ТO

00694581 03C7 ADD EAX,EDI 00694583 8985 C7010000 MOV DWORD PTR SS:LEBP+1C7],EAX 00694583 8985 S5 MOV EDX,DWORD PTR SS:LEBP+1C7] 00694582 8885 C7010000 MOV EDX,DWORD PTR SS:LEBP+1C7] 00694582 8942 0C MOV DWORD PTR SS:LEBP+1C7] 00694592 8942 0C MOV DWORD PTR SS:LEBP+1C7] 00694595 8D9D 0D040000 LEA EBX,DWORD PTR SS:LEBP+40D] Afterward 00694580 FFD2 00694589 90 00694580 8B55 5B 00694582 8B85 780 0009000 PTR SS:LEBP+581 00694582 8B85 780 00694582 8942 0C 0009000 PTR SS:LEBP+1C71 00694582				10	
00694583 8985 C7010000 MOV DWORD PTR SsitEBP+1C7], EAX 00694582 8855 5B MOV EDX, DWORD PTR SsitEBP+15B] 00694582 8942 0C MOV EDX, DWORD PTR SsitEBP+1C7] 00694582 8942 0C MOV EDX, DWORD PTR SsitEBP+1C7] 00694592 8942 0C MOV EDX, DWORD PTR SsitEBP+1C7] 00694592 8942 0C MOV DWORD PTR SsitEBP+1C7] 00694592 8942 0C MOV DWORD PTR SsitEBP+1C7] 00694581 BA E7034000 MOU EDX, F00F53~1.004003E7 00694586 FFD2 CALL EDX 00694588 90 MOV EDX, DWORD PTR SsitEBP+5B] 00694582 8855 5B MOV EDX, DWORD PTR SsitEBP+5B] 00694582 8855 C7010000 MOV EAX, DWORD PTR SsitEBP+1C71 00694582 8855 C7010000 MOV EAX, DWORD PTR SsitEBP+1C71 00694582 8942 0C MOV DWORD PTR DsitEBP+1C71 00694582 8942 0C MOV DWORD PTR DsitEBP+1C71 00694592 8942 0C MOV DWORD PTR DsitEBP+1C71		00694581	03C7	ADD EAX,EDI	
00694589 8855 5B MOV EDX, DWORD PTR SS: EEBP+5B] 00694592 8942 0C MOV EAX, DWORD PTR SS: EEBP+1C7] 00694592 8942 0C MOV EAX, DWORD PTR SS: EEBP+1C7] 00694593 8090 00040000 LEA EEX, DWORD PTR SS: EEBP+1C7] 00694594 8090 00040000 Afterward Afterward 00694586 FFD2 CALL EDX NOP 00694588 90 MOU EDX, DWORD PTR SS: EEBP+5B] 00694589 8855 C7010000 MOV EAX, DWORD PTR SS: EEBP+5B] 00694592 8942 0C MOU DWORD PTR DS: EEDX+CJ, EAX MOU EAX, DWORD PTR SS: EEBP+1C7] 00694592 8942 0C MOU DWORD PTR DS: EEDX+CJ, EAX MOU EAX, DWORD PTR SS: EEBP+1C7] 00694592 8942 0C MOU DWORD PTR DS: EEDX+CJ, EAX MOU EAX, DWORD PTR DS: EEDX+CJ, EAX		00694583	8985 C7010000	MOV DWORD PTR SS:[EBP+1C7],EAX	
00694582 8885 C7010000 MOV EAX,DUORD PTR DS:LEBP+1C71 00694592 8942 0C MOV DWORD PTR DS:LEDX+CJ.EAX 00694595 809D 0D040000 LEA EBX,DWORD PTR SS:LEBP+40D1 Afterward Afterward 00694588 90 MOV EDX,F00F53*1.004003E7 00694588 90 MOV EDX,DWORD PTR SS:LEBP+46D1 00694589 8855 58 MOV EDX,DWORD PTR SS:LEBP+581 00694592 8942 0C MOV DWORD PTR DS:LEDX+CJ.EAX 00694592 8942 0C MOV DWORD PTR DS:LEDX+CJ.EAX		00694589	8B55 5B	MOV EDX, DWORD PTR SS: [EBP+5B]	
00694592 8942 0C INOV DWURD PTR DS:LED#CJ,EHX 00694595 8D9D 0D040000 LEA EBX,DWORD PTR SS:LEBP+40DJ Afterward 00694581 BA E7034000 00694583 BA E7034000 00694584 FFD2 00694589 90 00694580 8855 58 00694582 8855 C7010000 0009 EAX WOV EDX,DWORD PTR SS:LEBP+58J 00694582 8855 C7010000 000 EAX DU EDX,DWORD PTR SS:LEBP+1C7J 00694582 8942 0C MOV EDX,DWORD PTR DS:LEBP+1CA 00694582 8942 0C		0069458C	8885 C7010000	MOV EAX, DWORD PTR SS: [EBP+1C7]	
Od654595 BD90 OD400000 FLEH EBA, DWORD FIR SSILEDF+40D1 Afterward Afterward 00694581 BA E7034000 MOU EDX, F00F53~1.004003E7 00694586 FFD2 CALL EDX 00694589 90 NOF 00694589 8855 B 00694589 8855 MOV EDX, DWORD PTR SS:[EBP+5B] 00694582 8855 MOV EAX, DWORD PTR SS:[EBP+1C7] 00694592 8942 0C MOV DWORD PTR DS:[EDX+C], EAX irst of all your will block to [opkcdl] of the		00694592	8942 00	INUV DWURD FIR DS:LEDX+CJ,EHX	
Afterward Ø0694581 BA E7034000 MOU EDX,F00F53~1.004003E7 00694586 FFD2 CALL EDX 00694588 90 NOP 00694589 8855 58 MOV EDX,DWORD PTR SS:LEBP+5B1 00694582 8885 C7010000 MOV EDX,DWORD PTR SS:LEBP+1C71 00694592 8942 0C MOV DWORD PTR DS:LEDX+C1,EAX		00694595	8090 00040000	LEH EBA, DWORD FIR SS:LEBF+40DJ	
00694581 BA E7034000 MOU EDX,F00F53~1.004003E7 00694586 FFD2 CALL EDX 00694588 90 NOP 00694589 8855 ThOV EDX,DWORD PTR SS:[EBP+5B] 00694582 8855 MOV EDX,DWORD PTR SS:[EBP+5B] 00694582 8885 C7010000 00694582 8942 MOV EDX,DWORD PTR SS:[EBP+1C7] 00694592 8942 MOV DWORD PTR DS:[EDX+C],EAX				Aiterward	
00694586 FFD2 CALL EDX 00694588 90 NOP 00694589 8855 58 00694584 8855 C7010000 00694582 8855 C7010000 00694582 8855 C7010000 00694582 8942 0C MOV DWORD PTR SS:[EBP+1C7] 00694592 8942 0C MOV DWORD PTR DS:[EDX+C],EAX		00694581	BA E7034000	MOV EDX,F00F53~1.004003E7	
00694588 90 NOP 00694589 8855 58 MOV EDX,DWORD PTR SS:[EBP+5B] 00694580 8885 C7010000 MOV EAX,DWORD PTR SS:[EBP+1C7] 00694592 8942 0C MOV DWORD PTR DS:[EDX+C],EAX rst of all you will look to [onbod] of the		00694586	FFD2	CALL EDX	
00694589 8B55 5B MOV EDX,DWORD PTR SS:[EBP+5B] 00694580 8B85 C7010000 MOV EAX,DWORD PTR SS:[EBP+1C7] 00694592 8942 0C MOV DWORD PTR DS:[EDX+C],EAX		00694588	90	NOP	
0069458C 8B85 C7010000 MOU EAX, DWORD PTR SS: LEBP+1C7] 00694592 8942 0C MOU DWORD PTR DS: LEDX+CJ, EAX		00694589	8855 58	MOV EDX, DWORD PTR SS:[EBP+5B]	
rst of all you will look to [opkod] of the		0069458C	8885 C7010000	MOV EAX, DWORD PTR SS:[EBP+1C7]	
rst of all you will look to [opkod] of the		00694592]	8942 UC	MOV DWORD PTR DS:LEDX+C],EAX	
-1 SC UE ALL VUL WILL LUUN CU LUUNUUL UL CH	r	st of	all vou w	ill look to [opkod] of t	:he

mov of edx, 004003E7.

Why thus? F

You do see? BA of - of [opkod] of mov of edx, and further goes address without any displacement (in "machine" idea). Then we cause function with the address, which is located into edx. This method is good even and fact that with the fulfillment of the instruction of call in the stack there will be the address of the recovery, to which we can pass, after carrying out, the instruction of ret. Why then we earlier did not use this method? But you will look, how many bytes it occupies. Thus far we have each byte on the calculation. The register of edx I used because further it it is rerecorded By aSProtect'[om]. We finish writing [patch]:



You will memorize, that to before return to the body Of aSProtect'[a] in the register of eax is located the address of that isolated to [oboasti] memory, where will pass ASProtect and this key place, after which ASProtect will work only in the chosen memory. You will memorize this number! [Potrassiruem], until we see here this code.

0C610F3 0C610F8 0C610F8	68 00800000 68 00 50	PUSH 8000 PUSH 8000 PUSH 60 PUSH EAX	
0C610FB	FF95 7D294400	CALL DWORD PTR SS:[EBP+44297D]	VirtualFree
0C61101 0C61103 0C61106 0C61107 0C61108 0C61109	8DØE 8551 2C 44 07 50 C3	LEA ECX, DWORD PTR DS: [ESI] TEST DWORD PTR DS: [ECX+2C], EDX INC ESP POP ES PUSH EAX RETN	Modification c

We see that after the call Of virtualFree Of aSPr again somewhere it passes and (here [khitrjuga]!) instructions after the address of 00C610FB are encoded and are deciphered not long before the fulfillment. It means necessarily to cause our [patch] in such place, where these instructions are already deciphered. To me they were pleased to instruction, that they are carried out before the instruction of call.

But as to us them to [propatchit]? Indeed this region to memory is allotted dynamically. You do remember, I did request to memorize value from the register of eax? With the aid of this address we will be able to [propatchit] the interesting us instructions. Let us calculate the displacement of data of instructions relative to address from eax. **00C610F of 3-00C61000=F3**. Let us write our [patch].

004003E2 - E9 BC3F2900 JMP F00F53"1.00694383 004003E7 03C7 ADD EAX,EDI 004003E9 8985 C7010000 MOV DWORD PTR SS:[EBP+1C7],EAX 004003EF 83C2 22 ADD EDX,22 004003EF C680 F300000 MOV BYTE PTR DS:[EAX+F3],0B9 004003FF 66:C780 F80000 MOV DWORD PTR DS:[EAX+F4],EDX 004004FF 66:C780 F8000 MOV DWORD PTR DS:[EAX+F4],EDX 004004FF 66:C780 F8000 MOV DWORD PTR DS:[EAX+F3],0D1FF 004004FF 66:C780 F8000 MOV DWORD PTR DS:[EAX+F8],0D1FF POP EBX POP EBX PUSH 8000 PUSH 0 JMP EBX 00400409 00800000 68 00400400 0040040F 00400411 6A 00 FFE3

We place [brjak] on **00400411** and start program. We are interrupted and look at the deciphered instructions.



Passage is accomplished to the address, which is located into eax, and here into eax this value falls from **ebp+442C51**. I can certainly not rights, but I am not confident, that the number **of 442C51** is constant, and is not generated each time (or each) anew. Therefore in order not to risk, let us replace the instruction **of retn** by the passage into our [patch], but in [patche] let us restore everything in the place (since we will rub over instructions after the instruction **of retn**) and let us carry out the instruction of retn in its [patche]. We finish writing [patch].

004003E9	8985 C7010000	MOU DWORD PTR SS:[EBP+1C7],EAX	
004003F2	C680 F3000000	MOV BYTE PTR DS:[EAX+F3],0B9	
004003F9	8990 F4000000	MOV DWORD PTR DS:[EAX+F4],EDX	
004003FF	66:C780 F80000	MOV_WORD PTR DS:[EAX+F8],0D1FF	
00400408	C3	RETN	
00400409	83C1 1A	ADD ECX,1A	
0040040C	C643 07 BB	MOV BYTE PTR DS:[EBX+7],0BB	
00400410	894B 08	MOV DWORD PTR DS:[EBX+8],ECX	
00400413	66:C743 0C FFD	MOV WORD PTR DS:[EBX+C],0D3FF	
00400419	5B	POP EBX	
0040041A	68 00800000	PUSH 8000	
0040041F	6A 00	PUSH 0	
00400421	FFE3	JMP EBX	
00400423	58	POP EAX	
00400424	C740 F9 C30000	MOV DWORD PTR DS: [EAX-7],0C3	
0040042B	66:C740 FE 000	MOV WORD PTR DS:[EAX-2],0	
00400431	C3	RETN	

We place [brjak] on **00400431** and will neglect program. They interrupted, we harvest F8 and fall into the program.

00C6130B	0000	ADD BYTE PTR DS:[EAX] AL
00C6130D	8B9D 552A4400	MOV EBX, DWORD PTR SS:[EBP+442A55]
00C61313	ØBDB	OR EBX, EBX
00C61315 🗸	74 0A	JE SHORT 00C61321
00041017	ODAO	MOLLEON DWODD DID DE.FEDNI

Now we twist window downward in search of the instruction of retn

and at the sufficiently large removal we see: 00061590 8306 14 00061590 8306 14 00061590 8306 14 000 ESI,14 0 00C6159H 8946 10 00C6159D 83C6 14 00C615A0 8B95 D8304400 00C615A6 ~ E9 EBFEFFFF JMF UMP 00C61496 MOV EAX,DWORD PTR SS:[EBP+442A65] PUSH EAX ADD EAX,DWORD PTR SS:[EBP+4430D8] POP EBX OR EBX,EBX MOV DWORD PTR SS:[EBP+442F11],EAX POPAD 00C615AB 00C615B1 00C615B2 00C615B2 8B85 652A4400 50 0385 D8304400 ØBDB. 00C615B9 00C615BB 8985 112F4400 61 75 Ø8 00C615C2 ~ 68 01000000 C2 0C00 68 00000000 MOV EAX,1 00C615C4 00C615C9 PUSH 0 C3 00C615D2 8885 DC304400 MOV EAX, DWORD PTR SS: [EBP+4430DC]

, in addition I am not confident, that this code will be always located on one and the same displacement relative to that address, on which we left [patcha]. Therefore I decided to search for this section on the signature and to place passage to [patch]. And, attention! Beginning from this address necessary to restore all rubbed over by us commands, since beginning from this address (can and earlier, we indeed already was restored the code) ASProtect begins to observe its integrity and will not make it possible so simple to rummage in its code. We finish writing [patch].



Well here, almost everything. Program is practically completely unpacked. BUT! If now it are neglected, then it will fall down with the cry about the fact that the virus is discovered. Now for us one must find testing the integrity of file and somehow mix ASProtect'[u] us to reveal. I decided to separately not [zamorachivatsja] and made just as Alex. We press Ctrl+G, we write **MapViewOfFile**, harvest Enter and we on this function. To place [brjak] on it is impossible in what place, since ASProtect completely [dizassembliruet] entire function and it searches for interceptions. It is possible to place [brjak] on the memory with this address, but, in addition this [brjak] will frequently operate because of constant checkings. But iron [brjaki] Of aSProtect in me constantly were discarded.

	1	
7C80B78D	8BFF	MOV EDI,EDI
7C80B78F	55	PUSH EBP
7C80B790	8BEC	MOV EBP,ESP
7C80B792	6A 00	PUSH 0
7C80B794	FF75 18	PUSH DWORD PTR SS:[EBP+18]
7C80B797	FF75 14	PUSH DWORD PTR SS:[EBP+14]
7C80B79A	FF75 10	PUSH DWORD PTR SS:[EBP+10]
7C80B79D	FF75 ØC	PUSH DWORD PTR SS:[EBP+C]
70808740	FF75 08	PUSH DWORD PTR SS:[EBP+8]
7C808783	E8 76FFFFFF	CALL kernel32.ManUiewOfFileEx
7C808768	50	POP EBP
7C808789	ČŽ 1400	RETN 14

But it is possible to enter differently. We see that the function **Of mapViewOfFile** is altogether only adapter to the function **Of mapViewOfFileEx**. Here on it let us place [brjak]. We harvest F9. We are interrupted and look into the window of stack.

ĺ	0012FDEC	7C80B7A8	CALL to MapViewOfFileEx from kerne
ļ	0012FDF0	0000007C	hMapObject = 0000007C (window)
l	0012FDF4	00000004	AccessMode = FILE_MAP_READ
ļ	0012FDF8	00000000	OffsetHigh = 0
	0012FDFC	00000000	OffsetLow = 0
j	0012FE00	00000000	MapSize = 0
3	0012FE04	00000000	BaseAddr = NULL
j	0012FE08	₽ 0012FE3C	
	0012FE0C	00C4867A	RETURN to 00C4867A
d			

We see that the address of recovery exactly falls into the body Of aSProtect'[a]. it is passed to it.



We see that into the register of ebx sends the indicator to the file. Theory is such, that it is necessary to emulate the function **Of mapViewOfFile**. To place interception on the command of mov of ebx, eax then to cause the function **Of virtualAlloc** for the isolation of memory under the file, then to copy the file, which loaded into the memory Of aSProtect (it is it's a pity, that it

only for reading) into the isolated by us memory and to there already lead file to that state, in which it was to [propatchivanija], and then restore the rubbed over instruction of mov of ebx, eax and that the fact that they rubbed over after it, then to substitute the address our of the original file to the address, cleaned. The place of the call **Of mapViewOfFile** we will also search for on the signature. We finish writing [patch].



Now after ASProtect will cause the function **Of mapViewOfFile** control immediately it will fall on address 0040048D, where we should cause the function **Of vitualAlloc**. But here indeed the misfortune Of aSProtect [bolshe] does not include this function in its IAT, which fills charger Windows. Now it obtains it dynamically. However, and that to us to now return back and to search for where ASProtect does cause this function and to memorize somewhere this address? But where we can it write down? I think more simply to itself to add this [fuktsiju] in IAT Of aSProtect'a. And let its Windows gives to us. Let us look with the aid of PE Of tools, where the directory of import begins.

Export Directory	00000000	00000000	Н	
Import Directory	00294A50	00000398	Н	

In principle it is possible to add function, also, with the aid of **PE Of tools** on the automaton. But me does not please itself as it this makes; therefore it is better by knobs. Let us calculate delta the displacement (read [manualy] on the import). For the beginning let us determine in what section it is located the directory of import.

S	ections Ed	itor					×
Г	Sections In	formations [HE	x]				
	Name	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics	
		0015C000	00001000	0008B400	00001000	E0000040	
		00056000	0015D000	00017200	0008C400	E0000040	
		00010000	001B3000	00001E00	000A3600	E0000040	
	.rsrc	000D1000	001C3000	0002D800	000A5400	E0000040	
	.data	00039000	00294000	00039000	000D2C00	E0000040	
	.adata	00001000	002CD000	00000000	0010BC00	E0000040	
L							
						Close	

We see that this .data, because VA of the directory of import is located exactly after VA of this section. It means Delta=00294000-000D of 2c00= of 1C1400. Well here now we can calculate RVA of the directory of import. RVA= of 00294A 50-1C1400=D3650. Let us open file into Hex editor and will pass to obtained RVA. We see massif IID.

000D3650	00	00	00	00	00	00	00	00	00	00	00	00	10	4A	29	00
000D3660	FC	49	29	00	00	00	00	00	00	00	00	00	00	00	00	00
000D3670	в8	4B	29	00	74	4C	29	00	00	00	00	00	00	00	00	00
000D3680	00	00	00	00	C4	4B	29	00	7C	4C	29	00	00	00	00	00

Let us look the name of first imported DLL. [Adres]=00294A of 10-1C1400=D3610. We look, what do we have with this address.

 D00D3610
 B
 65
 72
 6E
 65
 6C
 33
 32
 2E
 64
 6C
 00
 00
 047
 Mernel32.dll.

 D00D3620
 65
 74
 50
 72
 6F
 63
 41
 64
 64
 72
 65
 73
 73
 00
 00
 00
 etProcAddress

 D00D3630
 47
 65
 74
 4D
 6F
 64
 75
 6C
 65
 48
 61
 6E
 64
 6C
 65
 41
 GetModuleHand

 It is excellent!
 That which is necessary.
 Now let us look, where
 is located the massif of indicators to the names of functions.

 [Adres]=002949FC-1C1400=D35FC.
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C</t

000D35F0 00 00 00 00 00 00 00 00 00 00 00 00 1D 4A 29 00 000D3600 2E 4A 29 00 41 4A 29 00 00 00 00 00 00 00 00 00

As we see they are imported only three functions. As you know this massif it must conclude with 00000000, and, it concludes with 00000000 00000000. As it is successful! Instead of next-to-last 00000000 we can enter address for the line "Of virtualAlloc" and charger it automatically changes it to the real address of function. But there are no lines "Of virtualAlloc" in the file! But we will enter our its scarcely higher [patcha]. Let us make this in the same Of hex editor.

0000340 56 69 72 74 75 61 6C 41 6C 6C 6F 63 00 00 00 00 VirtualAlloc.. 0000350 C6 05 87 41 69 00 E9 C7 05 88 41 69 00 DA C1 D6 ж.‡Ai.йЗ.€Ai.Ъ We enter instead of next-to-last 00000000 - > 0000033E (00000340-2)

 000D35F0
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 <

All, now we know that we can cause **VirtualAlloc** to **D3608+1C1400+00400000=694A08** at any time ©. let us write our [patch]. Let us preserve registers and let us isolate memory with size our file:

	0040048 0040048 0040049 0040049 0040049 0040049 0040049 0040049 0040049	D 60 E BB 00C01000 3 6A 04 5 68 00100000 A 53 B 6A 00 D B8 084A6900 2 FF1A	PUSHAD MOV EBX,10C000 PUSH 4 PUSH 1000 PUSH 1000 PUSH 0 MOV EAX,<&kernel32.VirtualAlloc> COL DWARD PTR AS:[FAX]
Let us copy	y into '	the choser	n memory the file, loaded By
aSProtect'	[om] and	d will suk	ostitute indicator to our:
	004004A4 004004A6 004004AA 004004AC 004004AE 004004AE 004004B0	33D2 8B7424 1C 8BF8 8BC8 F3:A4 894424_1C	XOR EDX,EDX MOV ESI,DWORD PTR SS:[ESP+1C] MOV EDI,EAX MOV ECX,EBX REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI] MOV DWORD PTR SS:[ESP+1C],EAX
Let us clea	an [pate	cha] from	PE title:
	004004B4 004004B9 004004BC 004004C3 004004C3 004004C9 ^	B9 10030000 83C1 04 C70408 0000000 81F9 8A050000 76 EE	MOV ECX,310 ADD ECX,4 MOV DWORD PTR DS:[EAX+ECX],0 CMP ECX,58A UBE SHORT F00F53~1,004004B9

Let us restore the table of import and the beginning Of aSProtect'[a]:

004004CB	B9 08360D00 MOV ECX.0D3608
004004D0	C70408 00000000000 DWORD PTR DS:[EAX+ECX].0
004004D7	B9 01100000 MOV ECX.1001
004004DC	C70408 0140690 MOV DWORD PTR DS:[EAX+ECX],F00F53~1.0069400
004004E3	61 POPAD

Well let us restore the intercepted instructions and let us return to the body Of aSProtect'[a].

004004EE C746 04 4A0100(MOV DWORD PTR DS:[ESI+4],14A	004004E4 004004E5 004004E8 004004E8 004004EE 004004E5	5E 83EE 07 C706 8BD850E8 C746 04 4A0100 FFE6	POP ESI SUB ESI,7 MOU DWORD PTR DS:[ESI],E850D88B (MOU DWORD PTR DS:[ESI+4],14A UMP ESI
--	--	--	---

Now, if we neglect program, then it will be neglected, and it will completely normally work. Now let us find the place, where ASProtect enters in the program indicator to the key

enteerb	-		01	10	Program	THOTO	Jacor	0	0 0	110	nej.	_
Address	Hea	k di	ump							ASCI	I	
005B804C	00	00	00	00	AA AA AA AA	аа аа аа	00 00 0	аа р	30 00			
005B805C 005B806C	00 00	00 00	00 00	01 01	Backup			•	30 00 30 00	::::		
005B807C 005B808C	00 00	00 00	00 00	01 01	Сору			\mathbf{F}	90 00 90 00	::::		
005B809C 005B80AC	00	00 00	00 00	01 01	Binary				90 00 90 00	::::		
005B80BC 005B80CC	00	00 00	00 00	0	Label		1		00 00	• • • •		
005880DC 005880EC	00	00	00 00	ЙI ИI	Breakpoint			▶.	Mer	mory,	on access	
005B80FC	00	00	00	0i 0i	Search for				Mer	nory,	on write	Γ
rogram	1.	We	<u>ح</u> د	are	interr	upted	here					

We start ØØC32663

MOVS DWORD PTR ES:[EDI].DWORD PTR DS:[ESI] F3:A5 MOV ECX, EAX AND ECX, 3 89C1 83E1 03 00C32667 We harvest F8 and look, that we have to 005B804C. Address Hex dump ASCII

As we see this not that. Now ASProtect carried there the address, which was to the packing and where the further it it changes. Therefore again we harvest F9 and are interrupted here.

> 04 MOV EAX, DWORD PTR SS:[ESP+4] 35800 MOV DWORD PTR DS:[58804C], EAX 3 RETN 4 8B4424 04

0044CD79 A3 4C805 0044CD79 A3 4C805 0044CD78 C2 0400 But here this is already very interesting! Here ASProtect writes value from eax concretely with th address e interesting us. We harvest two times F8 and look, where they left:

00C4CABA 8B47 04 MOV EAX, DWORD PTR DS: [EDI+4] FFDØ CALL EAX MOV EAX,DWORD PTR DS:[C597E0] MOV EAX,DWORD PTR DS:[CAX+48] 00C4CABF A1 E097C500 8B40 48

I decided to intercept the function of call of eax and to write down with th address e interesting us indicator to th line e necessary to us, and after this to return control to the code, after call of eax. We will also search for this place on the signature. As I not tried, but so also I could not compose the unique signature of the long of 8 bytes. But that that it came out it is encountered in the file of 2 times and to us is necessary it is 2nd 4. let us write [patch]

004004E3 61 POPAD 004004E4 5E POP ESI 004004E5 83EE 07 SUB ESI,7 004004E5 C706 8B0550E8 MOV DWORD PTR DS:[ESI],E850D88B 004004E5 C706 8B0550E8 MOV DWORD PTR DS:[ESI],E850D88B 004004F5 60 PUSHAD 004004F5 64 A0100 004004F5 64 PUSHAD 004004F6 33FF XOR EDI,EDI 004004F8 46 INC ESI 004004F9 8B06 MOV EAX,DWORD PTR DS:[ESI] 004004F8 30 4704FFD0 00400500 75 F6 JN2 SHORT F00F53"1.004004F8 00400500 73 FF INC EDI 00400500 74 F0 JE SHORT F00F53"1.004004F8 00400500 74 F0 JE SHORT F00F53"1.004004F8 00400500 83EE 02 SUB ESI,2 00400500 83EE 02 SUB ESI,2 00400500 81C3 9300000 ADD EX,93 00400501 66:C746 05 FFD MOV DWORD PTR DS:[ESI+1],EBX	-	200 0.0	WEITOO LEON	/011]/
004004F5 60 PUSHAD 004004F5 33FF XOR EDI,EDI 004004F8 33FF XOR EDI,EDI 004004F8 8806 MOV EAX,DWORD PTR DS:[ESI] 004004F8 8806 MOV EAX,DWORD PTR DS:[ESI] 004004F8 304704FFD0 CMP EAX,DWORD PTR DS:[ESI] 00400508 75 F6 JN2 SHORT F00F53"1.004004F8 00400508 47 INC EDI 00400508 83FF 01 CMP EDI,1 00400508 83FF 01 CMP EDI,1 00400508 83EE 02 SUB ESI,2 00400508 83EE 02 SUB ESI,2 00400508 81C3 9300000 ADD EEX,93 00400504 81C3 9300000 ADD EEX,93 00400514 895E 01 MOV DUORD PTR DS:[ESI+1],EBX 00400517 66:C746 05 FFD MOV WORD PTR DS:[ESI+5],0D0FF 00400510 61 JMP ESI		004004E3 004004E4 004004E5 004004E8 004004E8 004004EE	61 5E 83EE 07 C706 8BD850E8 C746 04 4A0100	POPAD POP ESI SUB ESI,7 MOU DWORD PTR DS:[ESI],E850D88B MOV DWORD PTR DS:[ESI+4],14A
		004004F5 004004F8 004004F9 004004F9 004004F9 00400500 00400500 00400500 00400500 00400500 00400508 00400508 00400508 00400517 00400517 00400517	60 33FF 46 8806 3D 4704FFD0 75 F6 47 83FF 01 74 F0 83EE 02 C606 B8 81C3 93000000 895E 01 661C746 05 FFD 61 FFE6	PÚSHAD XOR EDI,EDI INC ESI MOV EAX,DWORD PTR DS:[ESI] CMP EAX,DWORD PTR DS:[ESI] CMP EAX,D0FF0447 JNZ SHORT F00F53"1.004004F8 INC EDI CMP EDI,1 JE SHORT F00F53"1.004004F8 SUB ESI,2 MOV BYTE PTR DS:[ESI],0B8 ADD EBX,93 MOV BYTE PTR DS:[ESI],0B8 ADD EBX,93 MOV WORD PTR DS:[ESI+1],EBX MOV WORD PTR DS:[ESI+5],0D0FF P0PAD JMP ESI

Well, then we intercepted the function Of aSProtect'[a], which corresponds for the registration and now we must it emulate. For this let us enter to **005B804C** indicator to the key. But indeed we do not have key! But we and it will enter scarcely higher our [patcha] (above the line "Of virtualAlloc"). We will use Hex by editor.

 320
 53
 70
 65
 63
 69
 61
 62
 0
 62
 75
 69
 62
 64
 20
 66
 72
 Special build fr

 330
 6F
 6D
 20
 50
 45
 5F
 4B
 69
 62
 60
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 0

THIC I	CC UD WIICC	[paceil].	
0040051	Ē FFE6	JMP ESI	
0040052	20 60	PUSHAD	
0040052	B9 4C805B00	MOV ECX,F00F53~1.005B804C	
0040052	26 C701 20034000	MOV DWORD PTR DS:[ECX],FO0F53~1.00400320	ASCII "Special build from PE_Kill"
0040052	20 61	POPAD	
0040052	2DI 58	POP EAX	
0040052	2E 48	DEC EAX	
0040052	2Fİ C740 FA 508B47	(MOV DWORD PTR DS:[EAX-6].4478B50	
0040053	36 66:C740 FE FFD	(MOV WORD PTR DS:[EAX-2],0D0FF	
0040053	3C C600 A1	MOV BYTE PTR DS:[EAX],0Å1	
0040053	E EEEO	IMP FOY	

Now, if we neglect program, then let us see, that NAG disappeared and program was neglected as [zareganaja]. Let us open window "about the program..." :

5 0	FTWARE							
	FontExpert	2005	OK.					
L	Version: 7.00 Release 1							
	Copyright @	9 1999-2005 Proxima Software						
	License Ke	y:						
	Special t	puild from PE_Kill						
	Web Site:	http://www.ProximaSoftware.com						
		Since 1991 on the market of font tools						

As we see - all good. But now let us try to transfer time forward and by [perezapustim] program.



[Mda]... We will fight. Let us determine, where occurs checking to the time expiration of [triala].

Let us place [brjak] on **MessageBoxA**. But! To place is necessary also as to **MapViewOfFile**. For the beginning let us place [brjak] on**0040053F**in our [patche]. Let us neglect program. They interrupted. We harvest Ctrl+G, we write **MessageBoxA**, harvest Enter. Now we in this function.

77D7050B	8BFF	MOV EDI.EDI
77D7050D	55	PUSH EBP
77D7050E	SBEC	MOV EBP.ESP
7070510	833D 1C04D977 (CMP DWORD PTR DS:[77D9041C].0
7070517	74 24	JE SHORT USER32.77D7053D
77070519	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]
77D7051F	6A 00	PUSH 0
77070521	FF70 24	PUSH DWORD PTR DS:[EAX+24]
77070524	68 F40AD977	PUSH USER32.77D90AF4
77070529	FF15 1812D377	CALL DWORD PTR DS: [<&KERNEL32.Interlocked
77D7052F	8500	TEST EAX.EAX
77070531	/ 75 0A	JNZ SHORT USER32.77D7053D
77070533	C705 <u>F00AD977</u> (MOV DWORD PTR DS:[77D90AF0],1
77D7053D	6A 00	PUSH 0
77D7053F	FF75 14	PUSH DWORD PTR SS:[EBP+14]
77D70542	FF75 10	PUSH DWORD PTR SS:[EBP+10]
7070545	FF75 ØC	PUSH DWORD PTR SS:[EBP+C]
7070548	FF75 08	PUSH DWORD PTR SS:[EBP+8]
77D7054B	E8 2D000000	CALL USER32.MessageBoxExA
7070550	5D	POP EBP
7070551	C2 1000	RETN 10

As we see this function it is also adapter to the function **Of messageBoxExA**. Let us pass to this function and let us place [brjak] on it. But now let us use [obaldennuju] [fichu] Of ollyDbg of - reverse laying out. For this let us first transfer diagnostic routine into the regime of the laying out of the code by the pressure of the keys For ctrl+F11. We await, until program interrupts on our [brjake]. It interrupted. Now we begin to harvest key "-" minus on the digital keyboard. In this case Olly begins to produce the reverse laying out of the code, in this case showing the state of registers and stack at that moment, when this instruction was carried out. We leave into this place.

			–
0044CCD1	50	DUCH EDA	
00440CD2	20		
0044CCB3	DE OGCEEZOG	MOU EST ENGEES"1 GGE4CEGG	OSCIL "FootEuropet 200E"
00440004	0070 EQ	LEO EDI DWODD DID CC.FEDD_101	HOLII FONCEXPERC 2000
0044CCB7	05	MAUS DWARD PTR 53:LEDF-101 MAUS DWARD PTR 59:LEDF101	
0044CCBD	05	MOUS DWORD PTR ESTEDII, DWORD PTR DSTESII	
0044CCBB	881D C4D75500	MOU FRY DWORD PTR DS (5507041	USER32 MessageBoyD
0044CCC4	95	MOUS DWORD PTR FS:[EDI1.DWORD PTR DS:[ESI1	ODEND2. NE SSAGEDONN
0044CCC5	6Å 13	PUSH 13	
0044CCC7	AS	MOVS DWORD PTR ES:[EDI].DWORD PTR DS:[ESI]	
0044CCC8	59	POP ECX	
0044CCC9	BÉ 30C55600	MOV ESI.F00F53~1.0056C530	ASCII "Trial period for this p:
0044CCCE	8DBD 7CFFFFFF	LEA EDI, DWORD PTR SS: [EBP-84]	·····
0044CCD4	F3:A5	REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI	
0044CCD6	6A 40	PUSH 40	
0044CCD8	8D45 FØ	LEA EAX,DWORD PTR SS:[EBP-10]	
0044CCDB	50	PUSH_EAX	
0044CCDC	8D85 7CFFFFFF	LEA EAX,DWORD PTR SS:[EBP-84]	
0044CCE2	50	PUSH EAX	
UU44CCE3	64 00	PUSH U	
0044CCE5	H4	MOVS BYTE FIR ESTLEDIJ, BYTE FIR DSTLESIJ	NCERCO, Managara Bawo
0044CCE6	FFU3	CHLL EBX	USER32.MessageBoxH
0044CCE8	5H 40	POBLECY	
0044CCER	57 DE 20045400	FUF EUA MOU EST EDAEES″1 AAE40499	00011 #01004547090100454709010
0044CCED	ODDD 7055555	1 EG ENT NMODE DED 00.FEDD-1001	HOULI 01204001070120400107012
0044CCF6	F3.05	REP MOUS DWORD PTR 53.[EDI1 DWORD PTR DS.[EST	
004400000	20.00010000	DUCU 100	

As we see precisely hence it was caused **MessageBox**. But there are no conditional jumps; therefore let us continue reverse laying out, until we leave here here.

00C4CE23	8847 28	MUV EHX,DWURD FIR DS:LEDI+28J
00C4CE26	FFDØ	CALL EAX
00C4CE28	837F 38 00	CMP DWORD PTR DS:[EDI+38],0
00C4CE2C	74 22	JE SHORT 00C4CE50
00C4CE2E	6A 01	PUSH 1
00C4CE30	68 2E57C300	PUSH 0C3572E
00C4CE35	832C24 02	SUB DWORD PTR SS:[ESP].2
00C4CE39 -	FF25 A8B4C500	JMP DWORD PTR DS:[C5B4A8]
00C4CE3F	EB ØF	JMP SHORT 00C4CE50
00C4CE41	837F 28 00	CMP DWORD PTR DS:[EDI+28],0

If passage to **00C4CE 2c** was carried out, then program would jump over through the leap **of 00C4CE39**, which leads for the communication about the end of [triala]. Will search for this place as you already they surmised on the signature. We finish writing [patch].

00400520	60			PUSH	IAD									
00400521	B9 4	C80	5B00	MOV	ECX, FOØF	53~1	.00588	:04C						
00400526	C701	200	034000	MOV	DWORD PT	R DS	:[ECX]	,FOØF5	3″1.0040032	2 ASCII	"Special	build	from	PE_Kill"
00400520	61			POPF										
00400520	58 40			DEC	EHA									
0040052E	C740	FO	509847	MOU	DWORD PT	e ne	• FEOX-	61 447	29850					
00400536	66:0	740	FF FFDI	MŎŬ	WORD PTR	DS:	FFAX-2	1.0D0F	F					
0040053C	Č600	Áľ		MŎŬ	BYTE PTR	DŠ:	CEAXJ.	0A1						
0040053F	60			PUSH	IAD									
00400540	40			INC	EAX									
00400541	8B08	- 44		MOU	ECX, DWOF		R DS:C	EAXI						
00400543	81F9	- ⁷⁴²	226401		EUX, 16H2	274	~1 00 <i>4</i>	00540						
00400549 00400549	15 F	^э гр		MOU	DUTE DTE	DC.	1.004 FEOV1	AED						
0040054E	8300	24		enň.	FAX. 24	50.	LEHOJ,	OLD						
00400551	Č600	Β9		MÖŬ	BYTE PTF	DS:	CEAX1.	0B9						
00400554	C740	01	640540	MOU	DWORD PT	RDS	: CEAX+	1],F00	F53~1.00400	34				
0040055B	66:C	740	05 FFD	MOV	WORD PTF	DS:	CEAX+5	3,0D1F	F					
00400561	61			POPF	AD SOL									
00400562	FFE0				EHA									
00400564	07 C641	DE	74	MOL	BUTE DTE	ne.	recy-2	B1 74						
00400569	83F9	йŽ	17	SUB	FCX.7	00.	LEON E							
0040056C	Č701	837	7F0800	MÕŨ	DWORD PT	R DS	: [ECX]	.87F83	3					
00400572	C741	04	7423A1	MOU	DWORD PT	R DS	: [ECX+	41,80F	12374					
00400579	FFE1			JMP.	ECX									

Now, if we neglect program, then it will be neglected, and it will consider itself [zareganoj], but if you visit into Tools->Detect Of font Of problems->Installed Of fonts and will press in the appeared window OK, then program will be thrown out into the system error. To catch the reason for error is simple. It is necessary to place [brjak] on reading to **005B804C**, then to open Tools->Detect Of font Of problems->Installed Of fonts to harvest OK and we will fall on checking of [zareganosti]. After which if we not [zaregany] window is shut normally, but if yes - that here here I did not greatly understand that he occurs. The whether this what first [izvratnaja] checking on [zareganost], then whether still that, but this something causes terrible Gluck and [proga] falls. These possible places are [poshifrovany] and are deciphered only with the introduction of key, can everything much simpler. On the whole it is necessary to read a little about the limitations, what me it does not absolutely want. If to you it is interesting to completely force open this program, then I think problems in you it will not arise, since last [propatchenaja] command was already in unpacked [proge] and now you can [patchit] any bytes of program. I round on this. Successes to you in all your undertakings! PE_Kill.

Complete [iskhodnik] of [patcha] (to 400350 it goes text, but [dizasm] accepted it for the code): /* 400320 */PUSH EBX /* 400321 */JO SHORT FO0F53~1.00400388 /* 400323 */ARPL WORD PTR DS:[ECX+61], BP /* 400326 */INS BYTE PTR ES:[EDI], DX /* 400327 */AND BYTE PTR DS:[EDX+75], AH /* 40032A */IMUL EBP, DWORD PTR SS:[ESP+20], 6D6F7266 /* 400332 */AND BYTE PTR DS:[EAX+45], DL /* 400335 */POP EDI /* 400336 */DEC EBX /* 400337 */IMUL EBP, DWORD PTR SS:[ESP+EBP * 2], 0 /* 40033F */ADD BYTE PTR DS:[ESI+69], DL /* 400342 */JB SHORT FO0F53~1.004003B8 /* 400344 */JNZ SHORT FO0F53~1.004003A7 /* 400346 */INS BYTE PTR ES:[EDI], DX /* 400347 */INC ECX

/* 400348 */INS BYTE PTR ES:[EDI], DX /* 400349 */INS BYTE PTR ES:[EDI], DX /* 40034A */OUTS DX, DWORD PTR ES:[EDI] /* 40034B */ARPL WORD PTR DS:[EAX], AX /* 40034D */ADD BYTE PTR DS:[EAX], AL /* 40034F */ADD DH, AL /* 400351 */ADD EAX, FO0F53~1.00694187 /* 400356 */JMP of 41C80922 /* 40035B */IMUL EAX, DWORD PTR DS:[EAX], FFD6C1 DA /* 400361 */JMP FO0F53~1.00694001 /* 400366 */SUB ECX, 2c0BD000 /* 40036C*/OF MOV BYTE PTR DS:[694228], 0E9 /* 400373 */MOV DWORD PTR DS:[694229], FFD6C155 /* 40037D */JMP FO0F53~1.0069418D /* 400382 */MOV SI, CX /* 400385 */MOV BYTE PTR DS:[6942FA], 0E9 /* 40038C*/OF MOV DWORD PTR DS:[6942FB], FFD6C09C /* 400396 */JMP FO0F53~1.00694248 /* 40039B */MOV EDX, 0A8E5462 /* 4003A0 */MOV BYTE PTR DS:[6943A3], 0E9 /* 4003A7 */MOV DWORD PTR DS:[6943A4], FFD6C00E /* 4003B1 */JMP FO0F53~1.006942FF /* 4003B6 */MOV BYTE PTR DS:[6943A3], OF /* 4003BD */MOV DWORD PTR DS:[6943A4], 1F82 /* 4003C7 */MOV BYTE PTR DS:[694581], OBA /* 4003CE */MOV DWORD PTR DS:[694582], FO0F53~1.004003E7 /* 4003D8 */MOV DWORD PTR DS:[694586], 8B90D2FF /* 4003E2 */JMP FO0F53~1.006943A3 /* 4003E7 */ADD EAX, EDI /* 4003E9 */MOV DWORD PTR SS:[EBP+1C7], EAX /* 4003EF */ADD EDX, 22 /* 4003F2 */MOV BYTE PTR DS:[EAX+F3], 0B9 /* 4003F9 */MOV DWORD PTR DS:[EAX+F4], EDX /* 4003FF */MOV WORD PTR DS:[EAX+F8], OD1FF /* 400408 */RETN /* 400409 */ADD ECX, 1a /* 40040C*/OF MOV BYTE PTR DS:[EBX+7], OBB /* 400410 */MOV DWORD PTR DS:[EBX+8], ECX /* 400413 */MOV WORD PTR DS:[EBX+C], 0D3FF /* 400419 */POP EBX /* 40041A */PUSH 8000 /* 40041F */PUSH 0 /* 400421 */JMP EBX /* 400423 */POP EAX /* 400424 */MOV DWORD PTR DS:[EAX- "], 0C3 /* 40042B */MOV WORD PTR DS:[EAX-2], 0 /* 400431 */PUSHAD /* 400432 */MOV EAX, DWORD PTR SS:[ESP+20] /* 400436 */INC EAX /* 400437 */MOV ECX, DWORD PTR DS:[EAX] /* 400439 */CMP ECX, 1b80875 /* 40043F */JNZ SHORT F00F53~1.00400436 /* 400441 */ADD EBX, 2F

/* 400444 */MOV BYTE PTR DS:[EAX], 0B8 /* 400447 */MOV DWORD PTR DS:[EAX+1], EBX /* 40044A */MOV WORD PTR DS:[EAX+5], 0D0FF /* 400450 */POPAD /* 400451 */RETN /* 400452 */POP EAX /* 400453 */SUB EAX, " /* 400456 */MOV DWORD PTR DS:[EAX], 1b80875 /* 40045C*/OF MOV DWORD PTR DS:[EAX+4], C2000000 /* 400463 */PUSH EAX /* 400464 */PUSHAD /* 400465 */PUSHFD /* 400466 */ADD EAX, OB /* 400469 */MOV EAX, DWORD PTR DS:[EAX] /* 40046B */DEC EAX /* 40046C*/OF MOV ECX, OF DWORD PTR DS:[EAX] /* 40046E */CMP ECX, D88BD0FF /* 400474 */JNZ SHORT FO0F53~1.0040046B /* 400476 */ADD EAX, 2 /* 400479 */MOV BYTE PTR DS:[EAX], OBB /* 40047C*/OF MOV DWORD PTR DS:[EAX+1], F00F53~1.0040048D /* 400483 */MOV DWORD PTR DS:[EAX+5], 5690D3FF /* 40048A */POPFD /* 40048B */POPAD /* 40048C*/OF RETN /* 40048D */PUSHAD /* 40048E */MOV EBX, 10C000 /* 400493 */PUSH 4 /* 400495 */PUSH 1000 /* 40049A */PUSH EBX /* 40049B */PUSH 0 /* 40049D */MOV EAX, <&kernel32.VirtualAlloc> /* 4004A2 */CALL DWORD PTR DS:[EAX] /* 4004A4 */XOR EDX, EDX /* 4004A6 */MOV ESI, DWORD PTR SS:[ESP+1C] /* 4004AA */MOV EDI, EAX /* 4004AC */MOV ECX, EBX /* 4004AE */REP MOVS BYTE PTR ES:[EDI], BYTE PTR DS:[ESI] /* 4004B0 */MOV DWORD PTR SS:[ESP+1C], EAX /* 4004B4 */MOV ECX, 310 /* 4004B9 */ADD ECX, 4 /* 4004BC */MOV DWORD PTR DS:[EAX+ECX], 0 /* 4004C3 */CMP ECX, 58A /* 4004C9 */JBE SHORT FO0F53~1.004004B9 /* 4004CB */MOV ECX, 0D3608 /* 4004D0 */MOV DWORD PTR DS:[EAX+ECX], 0 /* 4004D7 */MOV ECX, 1001 /* 4004DC */MOV DWORD PTR DS:[EAX+ECX], F00F53~1.00694001 /* 4004E3 */POPAD /* 4004E4 */POP ESI /* 4004E5 */SUB ESI, " /* 4004E8 */MOV DWORD PTR DS:[ESI], E850D88B /* 4004EE */MOV DWORD PTR DS:[ESI+4], 14A

/* 4004F5 */PUSHAD /* 4004F6 */XOR EDI, EDI /* 4004F8 */INC ESI /* 4004F9 */MOV EAX, DWORD PTR DS:[ESI] /* 4004FB */CMP EAX, D0FF0447 /* 400500 */JNZ SHORT FO0F53~1.004004F8 /* 400502 */INC EDI /* 400503 */CMP EDI, 1 /* 400506 */JE SHORT FO0F53~1.004004F8 /* 400508 */SUB ESI, 2 /* 40050B */MOV BYTE PTR DS:[ESI], 0B8 /* 40050E */ADD EBX, 93 /* 400514 */MOV DWORD PTR DS:[ESI+1], EBX /* 400517 */MOV WORD PTR DS:[ESI+5], 0D0FF /* 40051D */POPAD /* 40051E */JMP ESI /* 400520 */PUSHAD /* 400521 */MOV ECX, FO0F53~1.005B804C /* 400526 */MOV DWORD PTR DS:[ECX], FO0F53~1.00400320 /* 40052C*/OF POPAD /* 40052D */POP EAX /* 40052E */DEC EAX /* 40052F */MOV DWORD PTR DS:[EAX- '], 4478B50 /* 400536 */MOV WORD PTR DS:[EAX-2], 0D0FF /* 40053C*/OF MOV BYTE PTR DS:[EAX], OA1 /* 40053F */PUSHAD /* 400540 */INC EAX /* 400541 */MOV ECX, DWORD PTR DS:[EAX] /* 400543 */CMP ECX, 16A2274 /* 400549 */JNZ SHORT FO0F53~1.00400540 /* 40054B */MOV BYTE PTR DS:[EAX], 0EB /* 40054E */ADD EAX, 24 /* 400551 */MOV BYTE PTR DS:[EAX], 0B9 /* 400554 */MOV DWORD PTR DS:[EAX+1], F00F53~1.00400564 /* 40055B */MOV WORD PTR DS:[EAX+5], 0D1FF /* 400561 */POPAD /* 400562 */JMP EAX /* 400564 */POP ECX /* 400565 */MOV BYTE PTR DS:[ECX-2b], 74 /* 400569 */SUB ECX, " /* 40056C*/OF MOV DWORD PTR DS:[ECX], 87F83 /* 400572 */MOV DWORD PTR DS:[ECX+4], B0A12374 /* 400579 */JMP ECX