INSIDE THE ULTIMA ONLINE GOLD DEMO - UODEMO.DAT

GOAL

It's our goal to get a deep understanding of how the Ultima Online Gold Demo works. This demo is a representation of the rule set from the Ultima Online Second Age Era.

There is proof that some people have already reversed this demo partially or as a whole, however so far no tools or knowledge has been published. This project is to overcome those shortcomings.

URL's with some proof for this: <u>http://www.runuo.com/forums/general-discussion/94767-help-m-files.html</u> <u>http://azaroth.org/2008/12/31/your-topic/</u> (posting by Faust)

If we understand the demo there is a big chance we can alter the demo and even create our own demo. By default mounting horses is not possible in the demo, but what if we can alter the demo and unlock horses; can we then see how horses behaved during T2A?

This demo is 10 years old and I do not understand no one published his/her work. Maybe that DMCA thing is in the way?

UTILITIES USED

<u>IDA Pro</u>, a very professional utility, definitely worth buying, Standard version is affordable. <u>HxD</u>, a very neat hex editor and above all, it's free

ABOUT ME

I'm just a guy who loves the Ultima universe and knows a bit assembler. Why not combine the two? [©] I've been into computer starting from age twelve, and Ultima VII was the first game I bought myself. Oh yeah, I've been programming since my 13 and I started with assembler at age 15.

INITIAL LOOK

For the demo to work you must first install the Ultima Online Gold Client and then install the demo. The demo requires the client to be installed and it doesn't like newer clients, so my suggestion is: install the UO client & demo inside a virtual machine. How to do this is beyond the scope of this document and is left as an exercise for the reader.

Download the T2A era client from here: <u>http://www.uosecondage.com/downloads.aspx</u>, but do not install the accompanied patch!

After installing the demo you will see 3 files, 2 files of them being important:



Opening UODEMO.EXE is promising, it contains many readable strings thus OSI did not use encryption to protect their demo. However, UODEMO.DAT, contains 100% binary and therefore we can be sure it's compressed and/or encrypted somehow.

MHxD - [C:\Target\UO - Game\UoDemo\UoDemo.exe]			
- 🔝 Eile Edit Search View Analysis Extras Window ?	_18 ×	— 🔝 Ele Edit Search View Analysis Extras Window ?	8×
ANSI - hex -		• hex +	
- B UoDemo.exe		= ii) uodemo.dat	
T Offset (b) 00 01 02 03 04 05 06 07 08 09 08 08 00 00 08 05			
	OP LT		
00215780 6F 70 72 6C 74 00 00 00 4F 50 5F 4D 4F 44 00 00	oprit OP MOD	00000000 74 D4 ZD ZZ 48 D3 06 09 F3 72 92 34 60 CO-"HODI 4 00000000 01 61 DF 59 70 53 01 70 MF 74 9F 46 2F e6VnS n t žF	
00215790 6F 70 72 6D 6F 64 00 00 4F 50 5F 47 54 45 51 00	oprmod. OP GTEO.	00000011 48 F0 76 0F 11 13 6F 94 41 F7 43 D0 13 Hav : (p"1+CD6	
00215740 6F 70 72 67 74 65 71 00 4F 50 5F 4C 54 45 51 00	oprateg.OP_LTEO.	00000027 F4 kD 52 4F 89 9B 1D 40 4F 84 36 F8 93 8-R0%) 80 66%	
002157B0 6F 70 72 6C 74 65 71 00 4F 50 5F 4E 4F 54 00 00	opriteq.OP NOT	00000034 B4 48 27 B5 A2 38 CA 8A 76 48 32 39 6C 'H'uc8ÊŠvH291	
002157C0 6F 70 72 6E 6F 74 00 00 53 4D 5F 4C 50 41 52 45	oprnotSM LPARE	00000041 00 A2 22 F2 C0 F1 93 A4 00 5D 33 FD D4 .0"òàñ~¤.]3ýÔ	
002157D0 4E 00 00 00 6F 70 72 6E 75 6C 6C 00 53 4D 5F 4C	Noprnull.SM_L	0000004E 65 E2 85 15 AB FF 86 86 AE 91 8C C1 85 ea«ÿtt@`@A.	
002157E0 42 52 41 43 4B 45 54 00 6F 70 72 6C 69 73 74 00	BRACKET.oprlist.	0000005B 89 C2 5B C7 77 6D 1D E3 F9 FC C9 ED 30 % [Çwm.äùüÉi0	
002157F0 84 03 85 03 86 03 87 03 88 03 89 03 8A 03 8B 03	"	00000068 2F E9 D6 51 5C 7D 9F 7F DE B6 1D F9 01 /éÖQ\}Ÿ.♭¶.ù.	
00215800 8C 03 8D 03 00 00 00 00 97 0E 00 00 91 0E 00 00	œ	00000075 FE 42 7D DA 4E AB 2B A3 6B 97 37 9F 8A bB)ÚN«+£k-7ŸŠ	
00215810 92 0E 00 00 93 0E 00 00 94 0E 00 00 43 0F 00 00	' ^w ^w C	00000082 Å1 6E 87 FC C5 57 15 9F D4 37 45 30 E8 ;n‡üÅW.YÖ7EOè	
00215820 4D OF 00 00 60 OF 00 00 FA OE 00 00 F3 OE 00 00	И`úó	0000008F 5D 33 0D 85 D9 C2 FC B2 20 27 48 04 84]3UAu* 'H.,	
00215830 F6 0E 00 00 F7 0E 00 00 06 00 00 05 00 00 00	o÷	0000009C BB DE 2F 43 35 11 Å3 DC 64 3B 26 F5 Å3 »Þ/C5.£Ud; &ð£	
		UUUUUUA9 76 CB UD F1 C1 56 E1 C7 U8 62 C1 1F UA VE.NAVAÇ.BA	
		000000066 9E 2D 78 E6 12 FD CC AC 14 C2 50 4C AF Z-X2.91APL	
	vre fre stortes	000000003 70 94 20 78 13 04 66 90 66 97 68 53 08 57 08 57 08 57 08	
00215880 76 61 60 75 65 60 65 73 73 00 00 00 44 65 60 65	valueless Dele	00000000 DE F6 B6 15 F1 92 8F DC 61 60 38 4F 47 DagTyú Ha SNG	
00215890 74 69 6E 67 20 25 73 2E 00 00 00 4E 6E 20 73	ting ksNo s	000000Fb 2b 2c 5c x5 1x 52 51 5c cx cc 55 1E 11 62144 .0j 6NG	
00215810 75 63 68 20 70 6C 61 79 65 72 2E 00 44 65 6C 65	uch player. Dele	000000F7 81 B0 C4 10 11 F9 D0 F9 B2 1B 9D 2F D7 .°1 .éDé*	
002158B0 74 69 6E 67 20 25 73 2E 00 00 00 44 65 6C 65	ting %sDele	00000104 CO CD D1 7E 77 90 8C 1D 66 E5 E9 C1 94 ÅÍÑ~#.@.fåéÅ"	
002158C0 74 69 6E 67 20 25 73 2E 00 00 00 00 44 65 6C 65	ting %sDele	00000111 2B 84 89 F0 2D 98 C1 74 D4 2D 22 48 D3 + "to-"Ato-"Ho	
002158D0 74 69 6E 67 20 25 73 2E 00 00 00 00 59 6F 75 72	ting %sYour	0000011E 06 09 F3 72 92 21 77 07 77 C4 51 2D 19 or'!w.wkQ	
002158E0 20 68 6F 72 73 65 20 69 73 20 74 6F 6F 20 66 61	horse is too fa	0000012B 04 65 F5 2B EF 46 2E 48 E0 76 0F A1 A3 .eö+iF.Hav.;£	
002158F0 74 69 67 75 65 64 20 74 6F 20 6D 6F 76 65 2E 00	tigued to move	00000138 6E 94 41 F7 43 DO A3 E4 AD 52 4F 89 9B n"A+CD£a-ROt.>	
00215900 59 6F 75 20 61 72 65 20 74 6F 6F 20 66 61 74 69	You are too fati	00000145 1D 40 4F 84 36 F8 93 B4 48 27 B5 A2 38 .00,60" H'µ08	
00215910 67 75 65 64 20 74 6F 20 6D 6F 76 65 2E 00 00 00	gued to move	00000152 CA 8A 76 48 32 39 6C 00 A2 22 F2 CO F1 ÊŠvH291.¢"òÀň	
00215920 70 6F 69 73 6F 6E 65 64 00 00 00 00 63 75 72 73	poisonedcurs	0000015F 93 A4 00 5D 33 FD D4 65 E2 85 15 AB FF ``¤.]3ýÔeâ«ÿ	
00215930 65 00 00 00 6F 6C 64 42 6F 64 79 54 79 70 65 00	eoldBodyType.	0000016C 86 86 AE 91 8C C1 85 89 C2 5B C7 77 6D ++@`CA &A[Çwm	
00215940 6F 6C 64 48 75 65 00 00 6C 6F 63 6B 65 64 43 6F	oldHuelockedCo	00000179 1D E3 F9 FC C9 ED 30 2F E9 D6 51 5C 7D .auuÉi0/éOQ\)	
00215950 6E 74 61 69 6E 65 72 00 63 6F 72 70 73 65 00 00	ntainer.corpse	00000186 9F 7F DE B6 1D F9 01 FE 42 7D DA 4E AB Y.PM.u.pB}UN«	
00215960 6C 6F 63 6B 65 64 43 6F 6E 74 61 69 6E 65 72 00	lockedContainer.	00000193 2B A3 6B 97 37 9F 8A A1 6E 87 FC C5 57 +£k-7YS;n‡uAW	
00215970 67 6F 64 20 6C 6F 67 20 6F 75 74 00 6D 69 73 63	god log out.misc	000001A0 15 9F D4 37 45 30 E8 5D 33 0D 85 D9 C2 .Y07E0e]3UA	
00215000 00 00 00 00 07 67 69 60 60 60 70 60 79 60 67	godmode.myno	000001AD FC B2 20 27 48 04 84 BB DE 2F 43 35 11 U- H., »P/C5.	
00215030 (5 /3 03 04 0f 0f /2 00 00 /9 00 0f /5 /3 05 04 00215030 (F 6F 72 00 77 69 61 74 40 FF 6F 6C 6F 62 6P 00	oor what Jun look	00000107 F1 C7 00 62 C1 1F 03 0F 25 70 F6 12 F5 40 40 40 40	
00215980 04 74 61 00 10 74 61 00 10 74 61 00 B4 74 61 00	"to to to to	00000107 EI C7 00 62 CI IF OR 9E 2D 76 E6 12 FD aç.DR2-XE.Y	
00215900 C0 74 61 00 CC 74 61 00 D4 74 61 00 F0 74 61 00	àta Ìta Ôta àta	00000104 CC AC 14 C2 30 4C AF 70 54 2C 75 13 54 14.AFL p , (.0	
	èta, ôta, Jua, Jua,	000001FF 13 D9 05 17 9F 52 DF DB F6 B6 15 F1 92 11 SVDbDmgTW1/	
00215980 14 75 61 00 20 75 61 00 28 75 61 00 30 75 61 00	.ua. ua. (ua.Oua.	000001FB 8F DC 61 60 38 4F 47 21 32 F4 34 59 DC	
002159F0 38 75 61 00 44 75 61 00 4C 75 61 00 54 75 61 00	Sua.Dua.Lua.Tua.	00000208 BC 30 79 0A EF 08 73 81 B0 C4 A0 1A E9 40v.i.s.°A .é	
00215400 00 00 00 00 50 75 61 00 68 75 61 00 74 75 61 00	\ua.hua.tua.	00000215 DO E9 B2 AB 9D 2E D7 C8 0D D2 7E 77 90 Đế*«×È.Ò~w.	
0021510 84 75 61 00 00 00 00 00 00 00 00 94 75 61 00	"ua"ua.	00000222 8C 1D 66 E5 E9 C1 94 EB 86 89 F0 2D 98 @.fåéi/et%8-"	
00215A20 9C 75 61 00 14 5B 64 00 00 00 00 A8 75 61 00	œua[d"ua.	0000022F C1 74 D4 2D 22 48 D3 06 09 F3 72 92 2D ÁtÔ-"HÓór'-	
00215A30 B8 75 61 00 C4 75 61 00 D0 75 61 00 DC 75 61 00	,ua. Ăua. Đua. Ŭua.	0000023C 70 1F 67 C4 12 6A 19 1D 3F AA 5A EF 46 p.gÅ.j?*ZIF	
00215A40 E8 75 61 00 F8 75 61 00 00 76 61 00 0C 76 61 00	èua.øuavava.	00000249 2E 48 E0 76 OF &1 &3 6E 94 41 F7 43 D0 .Hav.; £n"&+CD	
00215450 18 76 61 00 24 76 61 00 30 76 61 00 38 76 61 00	.va.\$va.Ova.8va. 🧃	00000256 A3 E4 AD 52 4F 89 9B 1D 40 4F 84 36 F8 £a-R0%>.@0"6ø	+ 1
	Dira Tira Vira dira		
Urrsec: 21580C Block: 21580C-215918 Juength: 40 Rea	idoniy jOverwrite //	Intraction Readonly	11.
Type: Application Size: 2,82 MB	NOVE -	2,82 MB 🖳 My Computer	
針 Start 🛛 🚰 🈂 🖾 🔷 🔍 🖾 UoDemo 🛛 👀 HxD	HxD		12:24

(left side = UoDemo.exe, right side = uodemo.dat)

BEFORE WE CONTINUE

Let's not forget that the demo requires the client to be installed, that's because the graphics and sound are being loaded from the client itself.

INITIAL DISASSEMBLY

We load IDA and tell it to disassemble the demo executable.

.text:00469472				
.text:00469472	:		R O U T I N E =================================	
.text:00469472		R 6 7		
.text:00469472	: Attributes: b	o-based	Frame	
.text:00469472	A HARAK GERENERE EL	5 737753		
.text:00469472	: int stdcall	WinMain	(HINSTANCE binstance.HINSTANCE bPrevinstance.LPSTR loCodline.int oS	h
.text:00469472	WinMain@16	proc ne.	ar : CODE XREF: start+14Din	
.text:00469472				
.text:00469472	var 1C	= dword	ptr -10h	
text:00469472	uar 18	= dword	ntr -18b	
text:00469472	var 14	= dword	ntr -14h	
.text:00469472	var 10	= dword	otr -18h	
.text:00469472	var C	= dword	ptr -0Ch	
.text:00469472	var 4	= dword	ptr -4	
.text:00469472	hInstance	= dword	ptr 8	
.text:00469472	hPrevInstance	= dword	ptr 0Ch	
.text:00469472	lpCmdLine	= dword	ptr 10h	
.text:00469472	nShowCmd	= dword	ptr 14h	
.text:00469472				
.text:00469472		push	ebp	
.text:00469473		mov	ebp, esp	
.text:00469475		push	ØFFFFFFFh	
.text:00469477		push	offset WinMain@16 SEH	
.text:0046947C		mov	eax. large fs:0	
.text:00469482		push	eax	
.text:00469483		mov	large fs:0, esp	
.text:0046948A		sub	esp, 10h	
.text:0046948D		push	10h ; unsigned int	
.text:0046948F		call	??2@YAPAXI@Z ; operator new(uint)	
.text:00469494		add	esp, 4	
.text:00469497		mov	[ebp+var 14], eax	
.text:0046949A		mov	[ebp+var 4], 0	
.text:004694A1		cmp	[ebp+var 14], 0	
.text:004694A5		jz	short loc_4694B4	
.text:004694A7		mov	ecx, [ebp+var 14]	-
•				•

00068872 00469472: WinMain(x,x,x,x)

Let's start with looking for this UODEMO.DAT, we want to know where and how it loads.

STEP 1: open the list with labels and search for UODEMO

N Choose a name	<u>×</u>
Name	Address P.
A aWb_4	00627410
🔒 arundirArt_0	00627414
A aRb_9	0062742C
A arundir_0	00627430
A aResregions_txt	0062743C
A aR_18	0062744C
Enter the search substring	×
String UODEMO	•
	Cancel
	00627628
B al lodemo dat	00627730
A a g 0	0062773C
A a q	00627748
A aSqrt	006278E0
A aUs	006278E8
A aSs_0	006278EC
a Douturna interfa	00027900
OK Cancel	Help Search
Line 10818 of 13865	

STEP 2: go to the label and follow the XREF

STEP 3: analyze what the code does, use the built-in debugger

```
.Text:004E53DH
                                       յտը
                                                 SNORT 10C_4E530E
   .text:004E53DC
   .text:004E53DC
   .text:004E53DC loc_4E53DC:
                                                                    ; CODE XREF: sub_4E519A+17B<sup>†</sup>j
2.0
   .text:004E53DC
                                                 ebp, ebp
                                       xor
  .text:004E53DE
.text:004E53DE loc_4E53DE:
.text:004E53DE
                                                                    ; CODE XREF: sub_4E519A+2401j
                                                 offset allodemo_dat ; "uodemo.dat"
                                       push
 .text:004E53E3
                                                 ecx, ebp
[esp+2Ch+var_4], ØFFFFFFFFh
dword_701640, ebp
                                       mov
                                       mov
 .text:004E53ED
.text:004E53F3
                                       mov
                                       call
                                                 sub_4E2780
   .text:004E53F8
                                                 ecx, [esp+28h+var_C]
                                       mov
 • .text:004E53FC
                                                 edi
                                       pop
 • .text:004E53FD
                                       pop
                                                 esi
 • .text:004E53FE
                                                 ebp
                                       pop
 .text:004E53FF
.text:004E5406
                                                 large fs:0, ecx
                                       mov
                                                 ebx
                                       pop
                                                 esp, 18h
 .
   .text:004E5407
                                       add
 .
   .text:004E540A
                                       retn
  .text:004E540A sub_4E519A
                                       endp
```

This is a screenshot of the analyzed section:

	.LEAL.004E2000		
15.0	.text:004E53DC	LABEL_AllocationEailur	<pre>cuonemonat: ; CODE XREF: FUNC_Init_UODEMODAT+17BTj</pre>
	.text:004E53DC	< xor	ebp, ebp
	.text:004E53DE		EBP is zeroed
111000	.text:004E53DE	LABEL LoadDEMOUDDATand	Return: ; CODE XREF: FUNC Init UODEMODAT+2401j
4.	.text:004E53DE	push	offcot allodemo dat ; "uodemo.dat"
	.text:004E53E3	mou	ecx, ebn
•	.text:004E53E5	mov	[esp+2Ch+VAR Something], OFFFFFFFFh
•	.text:004E53ED	mov	GLOBAL Class UODEMODAT, ebp
٠	.text:004E53F3	call	FUNC MainOpenUODEMODAT
٠	.text:004E53F8	mov	ecx, [esp+28h+var C]
	.text:004E53FC	pop	edi
	.text:004E53FD	pop	esi
•	.text:004E53FE	pop	ebp
	.text:004E53FF	mov	large fs:0, ecx
•	.text:004E5406	pop	ebx
	.text:004E5407	add	esp, 18h
٠	.text:004E540A	retn	
	.text:004E540A	FUNC_Init_UODEMODAT en	dp

If you look well there is a possible crash!

If the code reaches LABEL_AllocationFailureUODEMODAT the EBP is zeroed, nothing wrong with that, but then EBP is stored in ECX. This is a standard (old) compiler mechanism; ECX stores the content of the this-pointer for those who know C++. From this you can derive that FUNC_MainOpenUODEMODAT is a class function.

Let's look at FUNC_MainOpenUODEMODAT:

	.text:004E2780							
	.text:004E2780	FUNC MainOpenUO	DEMODAT	proc	near	; CODE	XREF:	FUNC Init UODEMODAT+2591p
	.text:004E2780	Contraction of the second						
	.text:004E2780	CLASS UODEMODAT	= dword	ptr	-4			
	.text:004E2780	Filename	= dword	ptr	8			
	.text:004E2780							
٠	.text:004E2780		push	ebp				
•	.text:004E2781		MOV	ebp	, esp			
٠	.text:004E2783		push	ecx	North Contract Contract			
٠	.text:004E2784		MOV	[eb	p+CLASS U	DEMODAT],	, ecx	
٠	.text:004E2787		mov	eax	[ehp+Cl(DMADDIN 220	DAT]	
•	.text:004E278A		cmp	dwo	rd ptr [ea	ax+118h	9	
•	.text:004E2791		jz	sho	PT IUC 4E	2785	of the Local Division in which the local Division in which the local Division is not the local Division in the	
٠	.text:004E2793		MOV	ecx	, [ebp+CLi	ASS_UODEMO	IDAT]	Potential Crash
۰	.text:004E2796		MOV	edx	, [ecx+11	3h]		
•	.text:004E279C		push	edx				
٠	.text:004E279D		MOV	ecx	, [ebp+CLi	ASS UODEMO	DAT]	
٠	.text:004E27A0		call	sub	4E2995		STOCIOUS MORE	
	tovt · BBhE2705			19				

You notice that ECX is stored in [ebp+CLASS_UODEMODAT] and then that value is stored back in EAX. Very silly code? Yes it is, this tells us that the compiler used is really old and you should be amazed how compiler technology has improved. A modern C++-compiler with optimizations enabled will never ever create such redundant code.

About the possible crash: "cmp dword ptr [eax+118h]" -> if EBP was zeroed, ECX will be zero, [ebp+CLASS_UODEMODAT] will be zero and [eax+118h] will refer to [0+118h] and that is an illegal address under win32 bit. Access Violation guaranteed!

004E2780 FUNC MainOpenUODEMODAT proc near ; CODE XREF: FUNC Init UODEMODAT+2591p 004E2780 004E2780 CLASS UODEMODAT = dword ptr -4 004E2780 Filename = dword ptr 8 004E2780 004E2780 push ebp AA4F2781 mnu ebp, esp 004E2783 push ecx [ebp+CLASS_UODEMODAT] 004E2784 MOV ecx eax, [ebp+CLASS_UODEMODATT 004E2787 mov dword ptr [eax+118h] short loc_4E27A5 004E278A CMD 004F2791 jz ecx, [ebp+CLASS_UODEMODAT] 004E2793 MOV 004E2796 mov edx, [ecx+118h] 004E279C push edx ecx, [ebp+CLASS_UODEMODAT] 004E279D MOV 004E27A0 call sub_4E2995 004E27A5 CODE XREF: FUNC MainOpenUODEMODAT+111j 004E27A5 loc 4E27A5: mou ecx, [ebp+CLASS_UODEMODAT] 004E27A5 ecx, 004E27A8 and d sub_4E3030 004E27AE call 004E27B3 eax, [ebp+CLASS_UODEMODAT] mov 004E27B6 dword ptr [eax+@CD26Ch], 0 MOV : "rb+ 004F27C0 nush offset aRb 40 ecx, [ebp+Filename] 004E27C5 MOV ; Filename 004E27C8 push ecx 004E27C9 call Fopen 004E27CE esn. 8 add 004E27D1 edx, [ebp+CLASS_UODEMODAl] NUV eav 004E27D4 mov [edx+ 004E27DA mov eax, [ebp+CLASS UODEMODAT] 004E27DD dword ptr [eax+118h] CTO short LABEL_OpenUODEMODAT_OK 004E27E4 jnz 004E27E6 xor eax, eax 004E27E8 short LABEL_OpenUODEMODAT_Return jmp 004E27EA 004E27EA 004E27EA LABEL_OpenUODEMODAT OK: CODE XREF: FUNC_MainOpenUODEMODAT+641j ecx, [ebp+CLASS_UODEMODAT] 004E27EA MOV FUNC_LoadUODEMODATFileNameList 004E27F2 MOV eax. 1 **AA4E27E7** ; CODE XREF: FUNC MainOpenUODEMODAT+681j 004E27F7 LABEL_OpenUODEMODAT_Return: 004E27F7 MOV esp, ebp 004E27F9 ebp pop 004E27FA retn 4 004E27FA FUNC_MainOpenUODEMODAT endp 004E27ED: FUNC_MainOpenUODEMODAT+6D

Back, to the task at and, let us view the whole function:

There are some things going on which I didn't analyze 100% yet.

You notice that at 004E27D1 (in orange) the returned file handle is stored in [CLASS_UODEMODAT+118h]. That means the code at 004E2791 will only be executed when this function is called and the file handle is non-zero. The file handle will only be non-zero when the function has been called before. However, the XREFs teach us that this is not the case in this demo!?

LABEL_OpenUODEMODAT_OK is called when the file was opened correctly. Please know that UODEMO.DAT is opened by fopen and fopen does not support file sharing. Look up your C documentation please. ©

The following screenshot is inside this FUNC_LoadUODEMODATFileNameList (see the red line on the previous one)

	.text:004E2B29 loc_4E2B29:		; CODE XREF: FUNC_LoadUODEMODATFileNameList+1F6jj
	.text:004E2B29	mov	eax, 1
•	.text:004E2B2E	test	eax, eax
*	.text:004E2B30	jz	LABEL RestoreFilePointerAndReturn
•	.text:004E2B36	mov	ecx, [ebp+Str2]
	.text:004E2B3C	mov	edx, [ecx+118h]
•	.text:004E2B42	push	edx ; File
•	.text:004E2B43	push	1 ; Count
•	.text:004E2B45	push	118h ; ElementSize
٠	.text:004E2B4A	lea	eax, [ebp+VAR TemponaryBuffer]
	.text:004E2B50	push	eax ; DstBuf
	.text:004E2B51	call	locked fread
	.text:004E2B56	add	esp, 10h
	.text:004E2B59	push	23h ; '#'
•	.text:004E2B5B	lea	ecx, [ebp+VAR TemponaryBuffer]
•	.text:004E2B61	push	ecx
•	.text:004E2B62	call	FUNC DeXORDataOrStq2
٠	.text:004E2B67	add	esp, 8
	.text:004E2B6A	mov	edx, [ebp+Str2]
•	.text:004E2B70	mov	eax, [edx+0CD270h]
	.text:004E2B76	add	eax, 118h
	.text:004E2B7B	mov	ecx, [ebp+Str2]
•	.text:004E2B81	mov	[ecx+0CD270h], eax
•	.text:004E2B87	mov	edx, [ebp+Str2]
	.text:004E2B8D	push	edx ; Str2
	.text:004E2B8E	lea	eax, [ebp+VAR TemponaryBuffer]
	.text:004E2B94	push	eax ; Str1
*	.text:004E2B95	call	strcmp
	.text:004E2B9A	add	esp, 8
	.text:004E2B9D	test	eax, eax
	.text:004E2B9F	jnz	short loc_4E2BA6
•	.text:004E2BA1	jmp	LABEL_RestoreFilePointerAndReturn
	.text:004E2BA6 ;		

Important things to learn from this screenshot:

Data is read into a buffer of 0x118 bytes at VAR_TemponaryBuffer.

Data is XORed (the parameter 0x23 equals 0x118 / 4 / 2).

The function stops when VAR_Temponary is equal to Str2.

Str2 is not shown on the screenshot, but its content is "@@@.@@@".

The next obvious step is to go analyze the FUNC_DeXORDataOrStg2:

.text:004E4F02		
.text:004E4F02 loc 4E4F02:		; CODE XREF: FUNC DeXORDataOrStq2+A31j
.text:004E4F02	cmp	edi, ebp
.text:004E4F04	jge	short loc_4E4F6A
.text:004E4F06	mov	eax, [esp+20h+var_10]
.text:004E4F0A	add	eax, 1010101h
.text:004E4F0F	cmp	eax, 1010101h
.text:004E4F14	mov	[esp+20h+var 10], eax
.text:004E4F18	jnb	short loc 4E4F1F
.text:004E4F1A	inc	eax
.text:004E4F1B	mov	[esp+20h+var 10], eax
.text:004E4F1F		
.text:004E4F1F loc 4E4F1F:		; CODE XREF: FUNC DeXORDataOrStq2+531j
.text:004E4F1F	mov	eax, [esp+20h+var C]
.text:004E4F23	add	eax, 1010104h
.text:004E4F28	cmp	eax, 1010104h
.text:004E4F2D	mov	[esp+20h+var C], eax
.text:004E4F31	jnb	short loc_4E4F38
.text:004E4F33	inc	eax
.text:004E4F34	mov	[esp+20h+var_C], eax
.text:004E4F38		

And this is were at all became interesting, I'm not going to copy/paste the whole algorithm here, but the values 0x1010101 and 0x1010104 I found so weird and interesting that I used google to look them up. I reached a Russian forum that contained the source code of the GOST cipher. That source code fitted into this main loop directly!

```
/* The constants for addition */
#define C1 0x01010104
#define C2 0x01010101
void
gostofb (word32 const *in, word32 *out, int len,
       word32 const iv[2], word32 const key[8])
{
                               /* Counter */
       word32 temp[2];
       word32 gamma[2];
                               /* Output XOR value */
        /* Compute starting value for counter */
        gostcrypt(iv, temp, key);
       while (len--) {
               temp[0] += C2;
               if (temp[0] < C2)
                                     /* Wrap modulo 2^32? */
                       temp[0]++;
                                      /* Make it modulo 2^32-1 */
               temp[1] += C1;
               if (temp[1] < C1)
                                      /* Wrap modulo 2^32? */
                                      /* Make it modulo 2^32-1 */
                       temp[1]++;
               gostcrypt(temp, gamma, key);
               *out++ = *in++ * gamma[0];
               *out++ = *in++ * gamma[1];
       }
```

So, UODemo uses the GOST cipher, <u>http://en.wikipedia.org/wiki/GOST_28147-89</u>. For implementations of the cipher try this google search: <u>http://www.google.com/search?q=gost+gostofb&meta=</u>.

I'm sure that 10 years ago it would have been more difficult to discover this. But, what saddens me is that other people have already decrypted the demo and they never published this interesting fact. Come on people! Share you knowledge. Don't take it with you when you die. Aargh! Calm Blue Ocean. Calm Blue Ocean.

With the information we have now, we can actually write a utility that will decrypt this UODEMO.DAT file.

NOTE: the UoDemo only supports up to 3000 files (look at the disassembly yourself to verify this)

An idea I have: remove the encryption code from the demo and have it operate on an unencrypted version of UODEMO.DAT named UODEMO.BIN (for example).

That's it for now, I got bored of writing. Back to code please.