INSIDE THE ULTIMA ONLINE GOLD DEMO
- THE MAKING OF UODEMO+.EXE

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:
http://www.runuo.com/forums/general-discussion/94767-help-m-files.html
http://azaroth.org/2008/12/31/your-topic/ (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

IDA Pro, a very professional utility, definitely worth buying, Standard version is affordable.
HxD, 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. Knowing assembler comes in very handy and I never could imagine I would do the things that I'm doing now with this Ultima Online Demo.

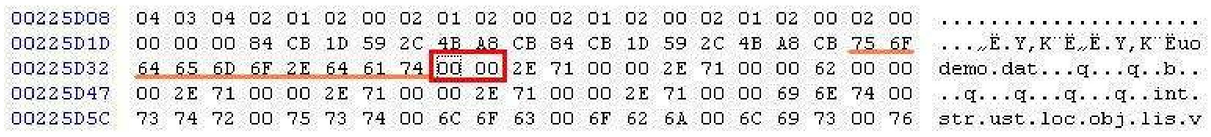
THE REASON

We're going to create a modified uodemo.exe that will allow us to make research easier. If we want to modify the uodemo.dat file without ever touching the original one, we need to create a patch for that purpose that will rename uodemo.dat.

LOCATING UODEMO.DAT

Patches like this are actually easy to make and can be done on many executables. Some programs have checksummed themselves and will give an error if you modify them, but uodemo.exe is not one of them.

Open UODEMO.EXE with your favorite hex editor and search:



```
00225D08  04 03 04 02 01 02 00 02 01 02 00 02 01 02 00 02 01 02 00 02 00  .....................
00225D1D  00 00 00 84 CB 1D 59 2C 4B A8 CB 84 CB 1D 59 2C 4B A8 CB 75 6F  ...„Ë.Y,K¨Ë„Ë.Y,K¨Ëuo
00225D32  64 65 6D 6F 2E 64 61 74 00 00 2E 71 00 00 2E 71 00 00 62 00 00  demo.dat...q...q..b..
00225D47  00 2E 71 00 00 2E 71 00 00 2E 71 00 00 2E 71 00 00 69 6E 74 00  ..q...q...q...q..int.
00225D5C  73 74 72 00 75 73 74 00 6C 6F 63 00 6F 62 6A 00 6C 69 73 00 76  str.ust.loc.obj.lis.v
```

You can change the text "dat" to "bin" and uodemo.exe will then operate on a file named uodemo.bin.

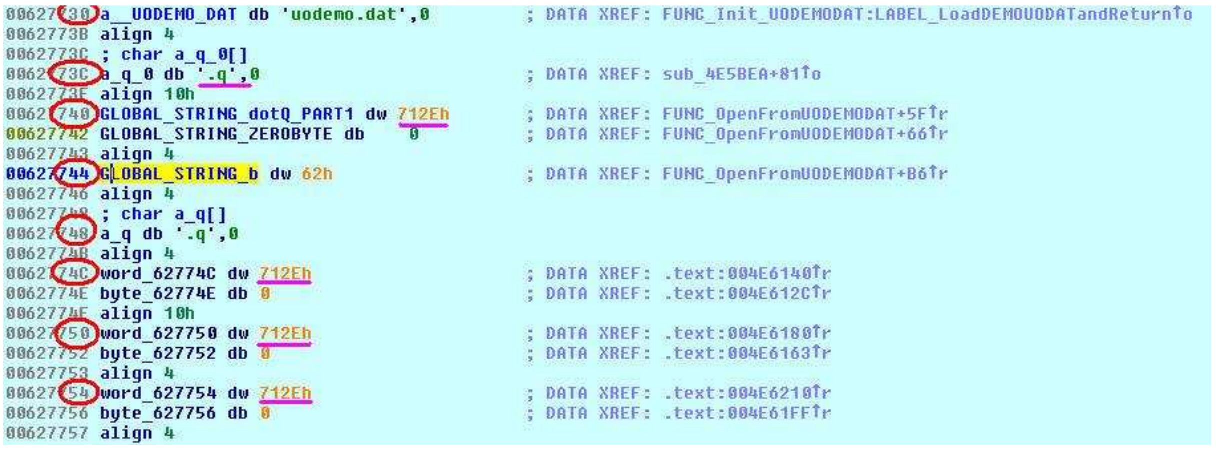
But I wanted a little more than that. The following patch is my personal ode to +ORC.

UODEMO+ IS BORN

I want to have an application named uodemo+.exe that operates on a file named uodemo+.dat. If you look at the screenshot above, you notice that there are 2 0-bytes following uodemo.dat. The first one is the <u>required</u> C string terminator for sure, but what's the second one?

QUICK ANALYSIS

Let’s open IDA again and locate the uodemo.dat string:

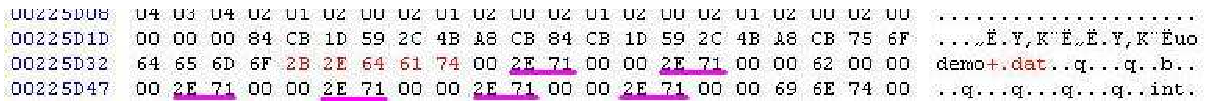


```
00627730 a__UODEMO_DAT db 'uodemo.dat',0           ; DATA XREF: FUNC_Init_UODEMODAT:LABEL_LoadDEMOUODATandReturn↑o
0062773B align 4
0062773C ; char a_q_0[]
0062773C a_q_0 db '.q',0                           ; DATA XREF: sub_4E5BEA+81↑o
0062773F align 10h
00627740 GLOBAL_STRING_dotQ_PART1 dw 712Eh         ; DATA XREF: FUNC_OpenFromUODEMODAT+5F↑r
00627742 GLOBAL_STRING_ZEROBYTE db    0            ; DATA XREF: FUNC_OpenFromUODEMODAT+66↑r
00627743 align 4
00627744 GLOBAL_STRING_b dw 62h                    ; DATA XREF: FUNC_OpenFromUODEMODAT+B6↑r
00627746 align 4
00627748 ; char a_q[]
00627748 a_q db '.q',0
0062774B align 4
0062774C word_62774C dw 712Eh                      ; DATA XREF: .text:004E6140↑r
0062774E byte_62774E db 0                          ; DATA XREF: .text:004E612C↑r
0062774F align 10h
00627750 word_627750 dw 712Eh                      ; DATA XREF: .text:004E6180↑r
00627752 byte_627752 db 0                          ; DATA XREF: .text:004E6163↑r
00627753 align 4
00627754 word_627754 dw 712Eh                      ; DATA XREF: .text:004E6210↑r
00627756 byte_627756 db 0                          ; DATA XREF: .text:004E61FF↑r
00627757 align 4
```

The picture tells us that all the strings are aligned on a 4-byte (DWORD) boundary. This is a compiler optimization technique. The plus side is that we have one extra byte available for renaming uodemo.dat to uodemo+.dat. The purple lines are only there to awaken your curiosity as they refer to “.q” strings.

FINAL PATCH

This is my final patch, don’t forget to save your new EXE as UODEMO+.EXE, nothing special to it:



```
00225D08  04 03 04 02 01 02 00 02 01 02 00 02 01 02 00 02 01 02 00 02 00  .....................
00225D1D  00 00 00 84 CB 1D 59 2C 4B A8 CB 84 CB 1D 59 2C 4B A8 CB 75 6F  ...„Ë.Y,K¨Ë„Ë.Y,K¨Ëuo
00225D32  64 65 6D 6F 2B 2E 64 61 74 00 2E 71 00 00 2E 71 00 00 62 00 00  demo+.dat..q...q..b..
00225D47  00 2E 71 00 00 2E 71 00 00 2E 71 00 00 2E 71 00 00 69 6E 74 00  ..q...q...q...q..int.
```

+ORC, you’re my hero!