INSIDE THE ULTIMA ONLINE GOLD DEMO
- THE PACKET COMMUNICATION – PART 3

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 does 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
Explorer Suite, it did the job for this project but the tool can be improved

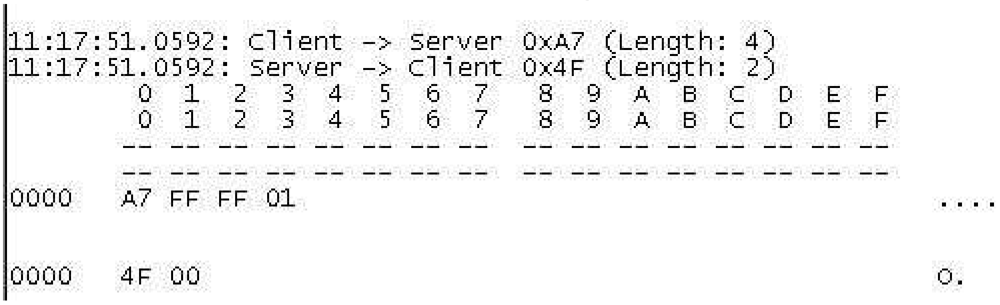
ABOUT ME

I'm just a guy who loves the Ultima universe and knows a bit assembler. Why not combine the two? ☺ When you are young you make mistakes, one of the ones I made was joining a cracking group. I think I stayed a member for one week. You not only had to compete against other cracking groups but also against other members of the group. Oh well, my advice (now that I passed the thirties I can start to give advice), if you want to crack software, do it for learning only and buy the software you like.

A BUG IN THE PACKET LOGGER

In “Part 2” of “The Packet Communication” series I explained how to code a packet logger, I spoke about thread-safety but I failed and I overlooked a situation that most likely will create weird looking log files.

Let’s take a look at a screenshot of such a log file:



```
11:17:51.0592: Client -> Server 0xA7 (Length: 4)
11:17:51.0592: Server -> Client 0x4F (Length: 2)
        0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
        0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
       -- -- -- -- -- -- -- --  -- -- -- -- -- -- -- --
       -- -- -- -- -- -- -- --  -- -- -- -- -- -- -- --
0000   A7 FF FF 01                                          ....

0000   4F 00                                                O.
```

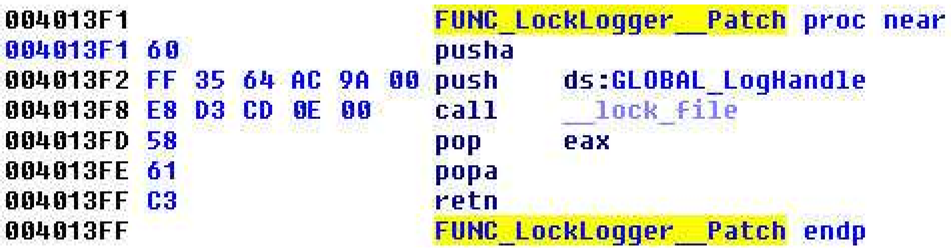
Some log lines from the client side and server side are mixed. Why is that? Because even though I added the patch in the safety of the critical sections, the critical sections were mend to only protect the linked lists. There are 2 linked lists, one for communication from client to server and one for the server to client communication. But sending and receiving can still happen simultaneously…

PATCHING THE PATCH

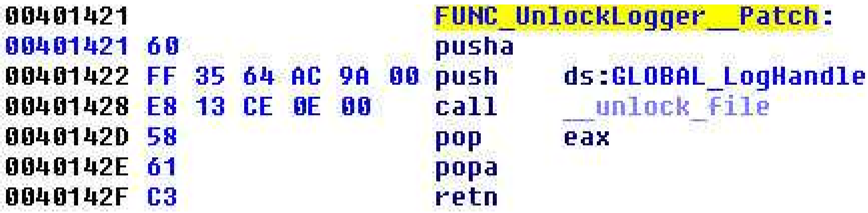
The run-time of the C++ compiler used to compile the uodemo.exe contains two functions we can use to make file access thread-safe, they are _lock_file and _unlock_file. Normally the fopen, fclose, fwrite, fprint and so on call those lock function themselves. But we can also call them ourselves to provide protection of the file during the whole time the packets are being logged.

BORING SCREENSHOTS

The two new functions used to lock and unlock the log file:

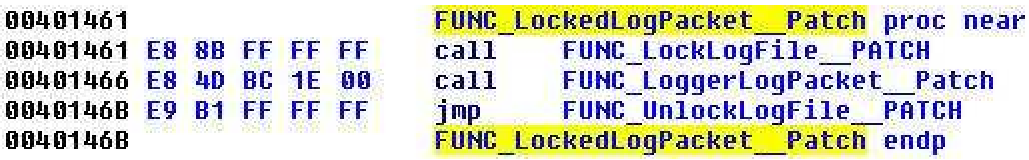


```
004013F1                   FUNC_LockLogger__Patch proc near
004013F1 60                pusha
004013F2 FF 35 64 AC 9A 00 push    ds:GLOBAL_LogHandle
004013F8 E8 D3 CD 0E 00    call    __lock_file
004013FD 58                pop     eax
004013FE 61                popa
004013FF C3                retn
004013FF                   FUNC_LockLogger__Patch endp
```



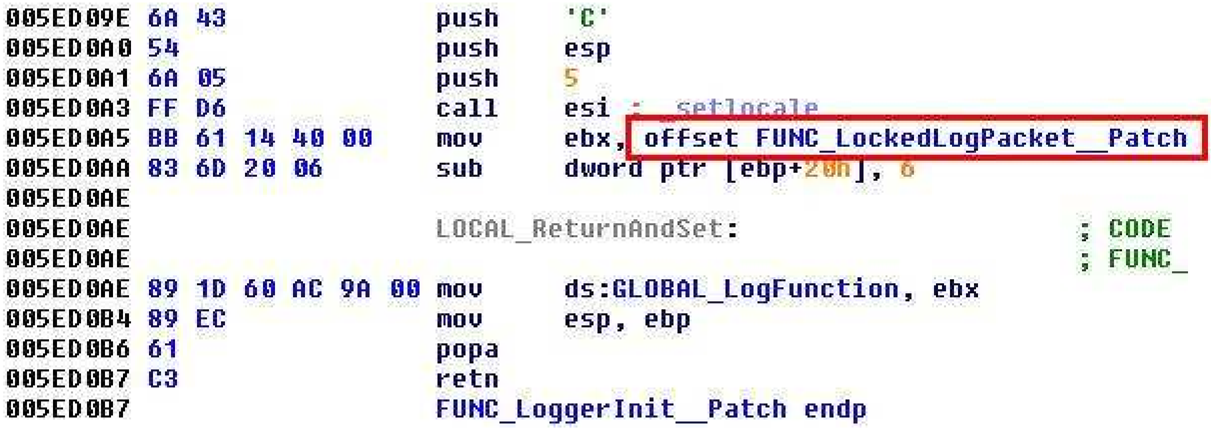
```
00401421                   FUNC_UnlockLogger__Patch:
00401421 60                pusha
00401422 FF 35 64 AC 9A 00 push    ds:GLOBAL_LogHandle
00401428 E8 13 CE 0E 00    call    __unlock_file
0040142D 58                pop     eax
0040142E 61                popa
0040142F C3                retn
```

This is the new function that locks the file, goes logging the packet and unlocks the file again:



```
00401461                   FUNC_LockedLogPacket__Patch proc near
00401461 E8 8B FF FF FF    call    FUNC_LockLogFile__PATCH
00401466 E8 4D BC 1E 00    call    FUNC_LoggerLogPacket__Patch
0040146B E9 B1 FF FF FF    jmp     FUNC_UnlockLogFile__PATCH
0040146B                   FUNC_LockedLogPacket__Patch endp
```

One existing function (FUNC_LoggerInit) required modification:



```
005ED09E 6A 43             push    'C'
005ED0A0 54                push    esp
005ED0A1 6A 05             push    5
005ED0A3 FF D6             call    esi ; _setlocale
005ED0A5 BB 61 14 40 00    mov     ebx, offset FUNC_LockedLogPacket__Patch
005ED0AA 83 6D 20 06       sub     dword ptr [ebp+20h], 6
005ED0AE
005ED0AE                   LOCAL_ReturnAndSet:                      ; CODE
005ED0AE                                                            ; FUNC_
005ED0AE 89 1D 60 AC 9A 00 mov     ds:GLOBAL_LogFunction, ebx
005ED0B4 89 EC             mov     esp, ebp
005ED0B6 61                popa
005ED0B7 C3                retn
005ED0B7                   FUNC_LoggerInit__Patch endp
```