The first thing I did, besides creating the Git Hub repository, is begin working on a stub .dll file and the python script.

I modified our example Python script for .dll injection so that It no longer required command line parameters. If the appropriate files are in the expected locations relative to the Python script, it’ll run on its own.

**Requirement #1:**

*Changes the message displayed on an illegal move to “Not in this game.”*

I started by searching in the String section for “That move is not allowed.” But I did not find it. I launched Freecell in IDA Debugger and set a series of breakpoints until I was able to isolate the function ProcessMoveRequest(x, x, x). Under the correct circumstances, this function will eventually call the Windows MessageBeep function and MessageBoxW function and display the “That move is not allowed.” text in a message box. I discovered that the offset had an empty byte in between each character (indicates which Unicode language character set is being used). The contents of this offset get moved into the ESI register before MessageBoxW is called.

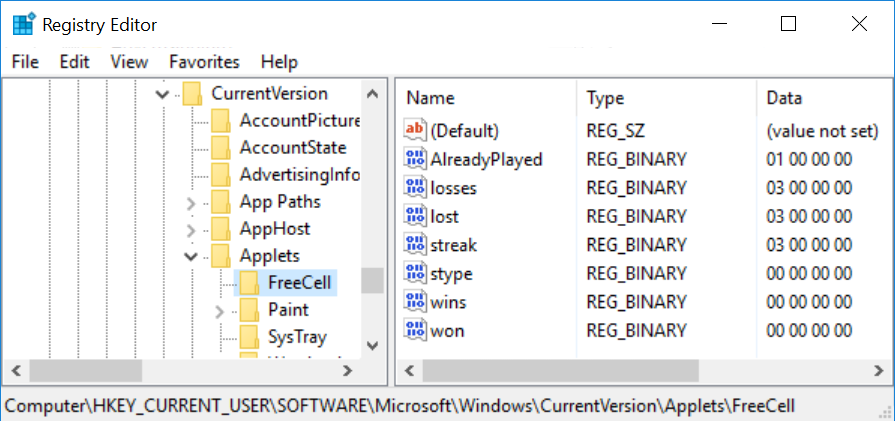
I found a tool called HxD which is another freeware hex editor. It also includes a tool for looking at the contents in RAM of a process you point it at. I used this to eventually find the location in memory where the string is located. After some messing around, I was eventually able to extract the contents of that string from memory to verify that I had located the correct location in memory.

But then I ran into another snag; my attempts to overwrite that section in memory failed. When I asked for assistance, I was reminded that the contents of RAM were protected and that I would need to use VirtualProtect() to change the access for that section on memory so that I could write to it. It took some more fiddling around - I have concluded that I am basically going to do everything incorrectly every possible way before doing it correctly – but I was eventually able to change the permissions to allow me to modify that section in memory, make the change, and then reset the permissions as before.

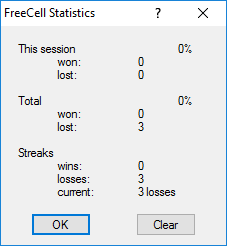
**Requirement #2:**

*Sets the total won to 1000 as shown by the Game->Statistics dialog.*

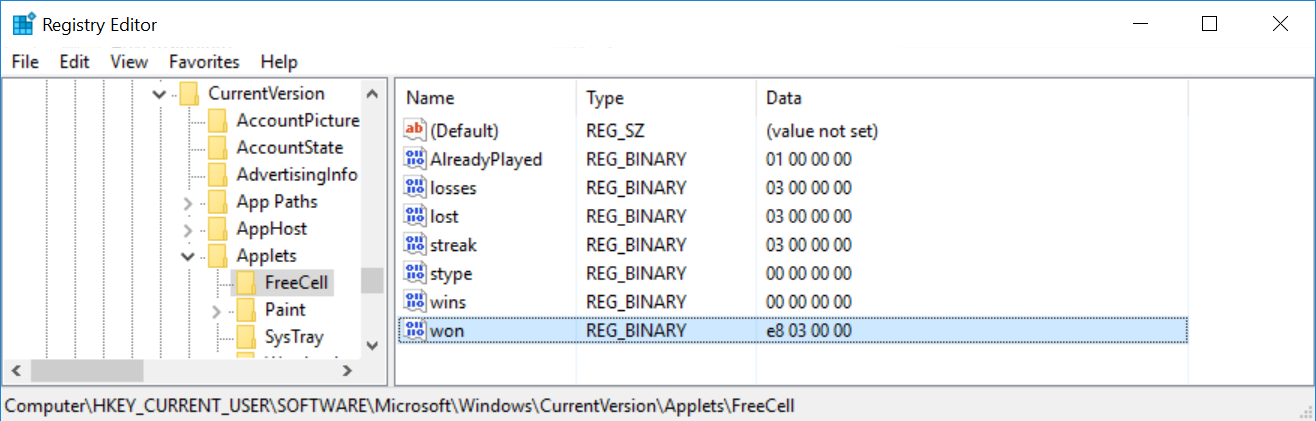
For Freecell to know how many games I have won and lost, it has to have a way to permanently store these statistics in a way that is preserved even when the freecell.exe process is no longer running. I have been writing AutoIt scripts for almost 18 years now for work. These scripts do things like automate installations of programs, create and modify registry keys that affect the behavior of programs, etc. Since I did not see any .INI files being created in the same directory as freecell.exe to store this data, I figured that it was stored in the registry. It would not make sense to store this data in HKEY Local Machine hive since that is for system-wide settings. So I launched Reg Edit and navigated to the HKEY Current User hive since that contains settings specific to my user account (i.e. would contains statistics relevant to my local Windows account). I performed a search using Control-F with the search string “freecell.exe”. I had to skip past references to IDA pro and Visual Studio until I found the key “HKEY\_CURRENT\_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Applets\FreeCell”.

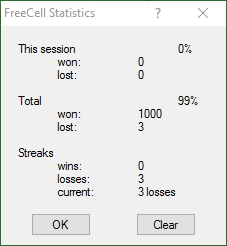


I noticed that the REG\_BINARY value names and values appeared to directly or indirectly reflect those in the FreeCell Statistics window, as shown below.



I have never worked with REG\_BINARY values before so it took a little bit of time to figure out how to manipulate the value properly. I used Windows 10’s calc.exe in Programmer mode and discovered that 1,000 in decimal is equivalent to 0x03E8. When I manually changed the key to that value in Reg Edit the FreeCell Statistics window gave the incorrect value in decimal. I eventually figured out why – I had to enter it in Little Endian format. These screenshots show the correct value and result:





I was able to find example code that allowed me to write to the registry once I was able to determine the proper values for the function.a

**Requirement #3:**

*The next valid move wins the game.*

**Requirement #4:**

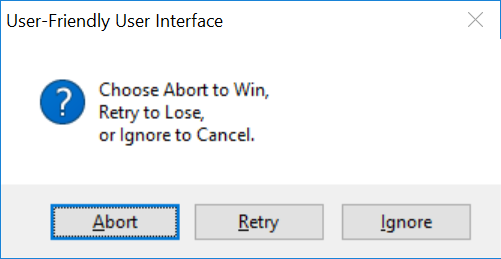
*Freecell contains a cheat code, that, when Ctrl – Shift - F10 is pressed, a dialog is presented to the user to “Abort, Retry, or Ignore”. If the user clicks “Abort”, the game is automatically won. Your DLL should change this cheat-code sequence to Ctrl - Shift – F6.*

Conor and I met multiple times on this project. We used a couple of tools to help with this one - IDA Pro Free to disassemble and debug freecell.exe, and Resource Hacker. Using Resource Hacker, we determined that there is an Accelerator table called FREEMENU which includes the keystroke combination hotkeys for freecell.exe. In the Accelerator table, the line of code which we are interested in is this one:

VK\_F10, 114, CONTROL, SHIFT, VIRTKEY

It is currently set to Control-Shift-F10 but we need to patch it to Control-Shift -F6.

While trying to isolate where in freecell.exe the accelerator tables are referenced, In IDA Pro we found a switch statement at address 0x01001EA1. This switch statement references off\_1002108 which contains a collection of 15 offsets. Each offset contains a memory location to which the program execution should continue based upon which hotkey combination was entered by the user. We identified the offset responsible for jumping to loc\_10020B9, which contains the instructions for displaying the following window:



After a lot of discussion and experimentation, we decided to find the location of the accelerator table in memory and patch it to use Control-Shift-F6 instead of Control-Shift-F10. By setting a breakpoint at 0x0100225F and launching IDA in Debugger mode, we learned that the Accelerator table is loaded into memory during initialization of freecell.exe.

**Requirement #5:**

*Ctrl-Shift-F2 wins the game.*