# Game Configurations

## Pilot Game: Cabal Online

Cabal Online (Korean: 카발 온라인, stylized as CABAL Online) is a free-to-play, 3D massively multiplayer online role-playing game developed by South Korean company ESTsoft. Different localizations of the game exist for various countries and regions. Although free-to-play, the game makes use of the freemium business model by implementing an "Item Shop", both in-game and via web, allowing players to purchase special premium coins using real currency, in order to acquire exclusive game enhancements and features, useful items and assorted vanity content.[[1]](#footnote-1)

## Meta data

Here you can find the executable data of cabal online and the protega anti-hack:

1. Cabal Online:

Executable file: CabalMain.exe

Architecture: x86

Size (Without protection): ~13 MB

Size (With protection):

1. Protega:

Executable file: Protega.dll

Architecture: x86

Size: 600 KB

Since the Cabal Online executable is only available in x86, the protega dll must be compiled in x86.

## Anti-hack DLL Hook

### Intro

The Protega Dynamic Link Library must be included in the Cabal executable. Since there is no source code of the game client, the DLL has to be hooked in the game executable.

Used tool: OllyDBG ver. 1.1

Target: CabalMain.exe

### Important Assembler / Function Information

Here you can find a short explanation of the assembler commands and data types that was used to do the hook:

|  |  |
| --- | --- |
| **Commands / data types** | **Description** |
| Push | Write a word to the top of the stack. Other commands or interactions often use the word on the stack. |
| Call | Call the subroutine specified in the operand. There are four possible ways to call this function. In this implementation, the “Direct Inter-Segment call” is used:  This call is able to call a target address, which is not in the same segment as the call itself. |
| Retn | Near Return. It returns only the instruction pointer without the segment. |
| 00´s |  |
| NOP´s |  |

This table shows the functions that got called in the hook:

|  |  |
| --- | --- |
|  |  |
|  |  |

### Implementation

To hook a DLL into a compiled exe, its necessary to firstly search for free space in the executable. After that, we convert some free space to “00´s”, because the “free” space isn’t empty.

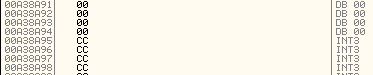


Figure 1 This Figure shows INT3 (The "free" space) and the 00´s which is the really empty space

Then we do the following steps:

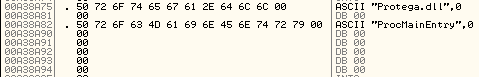
1. The application needs the following information: “Protega.dll” as a name + the function “ProcMainEntry” which is our function we want to call in our DLL. We write them into the empty space as byte code.

Figure 2 Inserted ASCII strings in the empty space

The addresses of these strings must be saved for later.

1. Now the DLL function must be called. This code must be written in NOP´s, so some INT3 space must be converted to it.

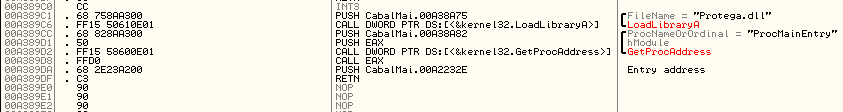


Figure DLL function call + push to entry address

First, we use push to write the “Protega.dll” string on the stack. After that, we must call the “LoadLibraryA” function in kernel32.dll. In OllyDBG, it’s possible to search those addresses (Seach for -> Name (label) in current module). The function call will use the address which is currently on the stack, so in that case LoadLibraryA will load “Protega.dll”. The next commands push the string of our function + the EAX register in the stack, to prepare the function call itself. Now, the call function tries to call “GetProcAddress” which uses the function string “ProcMainEntry” and the hModule of the EAX.

After all that steps are done, then “Call Eax” command calls the stack, which triggers the DLL function.

1. To start the game normally, we must push after the DLL function call to the entry point of the game. But that is not enough. Currently, our code is just somewhere in the executable. So, the old entry point (the game start routine) must be changed to our programed DLL function call. The program PE Explorer can easily change the old entry point 00A2232E in Figure 2 to 00A389C2 (which is the first address of our DLL function call). After that, the executable jumps first to our and second to the common game startup routine.

### Important Addresses

|  |  |
| --- | --- |
| **Address** | **Stored information** |
| 00A38A75 | “Protega.dll” as string |
| 00A38A82 | “ProcMainEntry” as string |
| 010E6150 | Address of LoadlibraryA in Kernel32.dll |
| 010E6058 | Address of GetProcAddress function in Kernel32.dll |
| 00A389C1 | New Entry point of the game executable |
| XXXXXXXX | Old Entry point of the game executable |
|  |  |

1. Seite: Wikipedia. Titel: Cabal Online. URL: https://en.wikipedia.org/w/index.php?title=Cabal\_Online&oldid=800612133 (09.10.2017). [↑](#footnote-ref-1)