# Reverse engineering 'Black Desert Online' (1. Preface)

Jan 9, 2019

## Preface

This is going to be a short introduction to the tool sets we're going to be using to reverse engineer the MMORPG 'Black Desert Online'. You can skip this if you're slightly-experienced in reverse engineering. The goal of this series is to educate you on the basics of reverse engineering massive multiplayer online games from scratch, without any previous knowledge neither on the game nor the engine. This series will also be showcasing how exactly we would approach such a massive game as a MMORPG, and eventually end up breaking key functionality for fun ~~and profit~~. The series will gradually get harder and harder to appeal to the widest possible audience, and to create incentive to progress in the reverse engineering field.

The absolute first thing we do when we begin reverse engineering **any** new game, is to *actually play the game*. Just for a few hours, get a feeling of mechanics and how the engine might operate. Ask yourself: Would *x* be server-sided, or would that be too hard for a server to handle with *y* players per server? Looking at various functionality is key when analyzing games. It's hard to guess what exactly is handled by the server properly, and what we're able to hack with just *client* access. This may sound very trivial, but many fail to acknlowedge the fact that it is absolutely crucial to understand the game before we can understand the internals of it. We will eventually stumble upon strings, type-information leaks or various debugging information that could be very benefitial, but we wouldn't realize because we never played the game and didn't recognize the specific term mentioned. Therefore, we highly suggest to get a grasp of the game, before doing any work on it.

## Index

This series will continue gradually, and this preface will also act as an index of all the posts. If the post in the index does not have a reference to a blog post, it has not yet been published. There are no deadlines for future posts, we're writing for fun when we have some spare time.

## Tools

On to the tools; We won't spend much time arguing for the specific tools we will be using in this series, besides a *tiny deviation* in tool-choice due to anti-cheat measures taken by Black Desert Online. This aforementioned deviation is the choice of CrySearch instead of Cheat Engine. We're **avid** Cheat Engine users and it is objectively superior to CrySearch, at least in terms of the functionality you will need for reverse engineering video games.

- CrySearch
  - Description: CrySearch is a memory-hacking utility that allows for observing and modifying values in memory.
- ReClass.NET
  - Description: ReClass.NET is a visual memory-editor that any sane game analyzer uses. Any ReClass version will suffice.
- x64dbg
  - Description: x64dbg is an open-source x64/x32 debugger for windows.
  - Plugins: Any dumper plugin & ScyllaHide
- Hex-Rays -> IDA
  - Description: The IDA Disassembler is an interactive disassembler, mainly used for static-analysis.
  - Plugins: Hex-Rays & ClassInformer

## Anti-Cheat

Black Desert Online uses *XIGNCODE*, an anti-cheat solution provided by WellBia Co.,Ltd. *XIGNCODE3* is extensively used in online games provided by Tencent. The relevant features listed on the website include:

- Detect modified of function about time
- Detect time modified via using time server
- Detect of game client local time modification
- Detect of major kernel function modification
- Detect DLL injection
- Detect virtual memory code injection
- Detect illegal control of DHCP
- Detect call specific function in game
- Detect game resource modification
- Reject game process memory accessing
- Reject game process handle accessing
- Reject game process message transmission
- Reject game process keyboard/mouse input transmission

[sic]

We will not be publicly disclosing any relevant information on bypassing the respective anti-cheat measures, but we have taken the voluntary handicap of using *CrySearch* over *Cheat Engine*, so readers can follow our methods without having to completely bypass the anti-cheat, but can stick to public bypasses.

## First steps of analysis

The first thing we will need to do is set up the analysis environment. The order of operations does not really matter, but we prefer to dump the game binary from memory and let IDA go at it whilst we play the game as mentioned earlier. To dump Black Desert online, open *x64dbg* and go to *Plugins->Scylla*

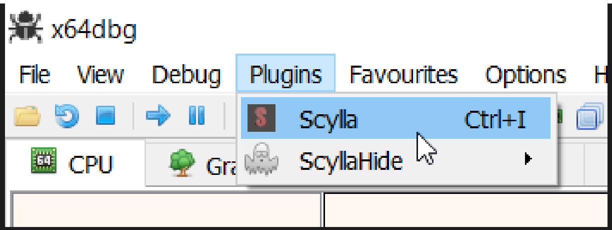


Fig.1 - Scylla

Then select the *BlackDesert64.exe* process, and hit dump. We will not be worrying about repairing the *import address table* for now, as it will most likely not be relevant for basic game analysis. When Scylla has finished dumping the game binary, let IDA have a go at it while we play. After spending 1 hour figuring out the game mechanics, we will realize IDA is still analyzing, and seems to be hanging in a specific portable-executable section called '____', this section, when manually inspected, looks like absolute garbage code and will be discarded. To discard a section, go to *View->Open subviews->Segments* (shortcut: *Shift+F7*). Then right click on the segment and choose '*Delete segment..*' (Hotkey: *Del*)

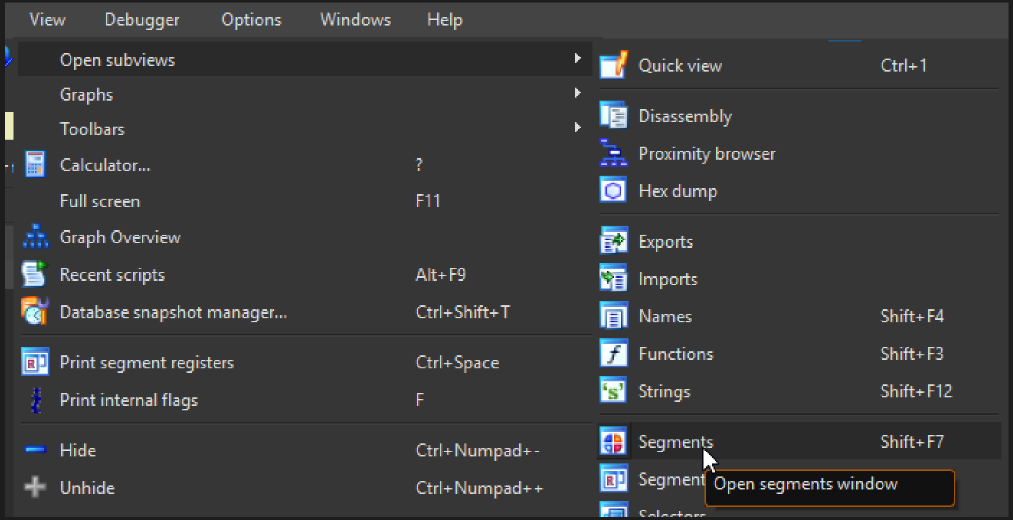
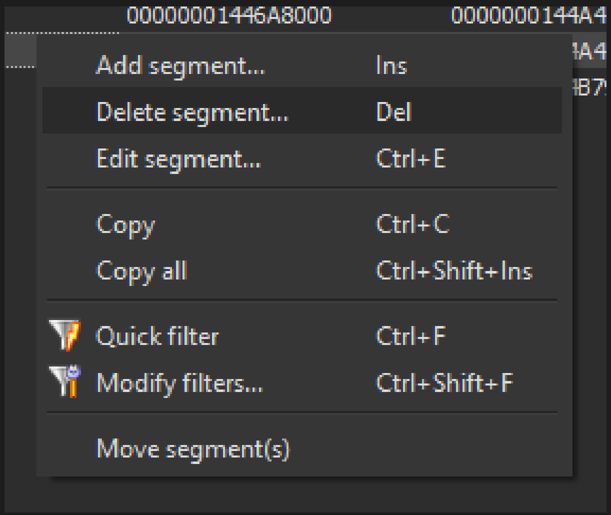
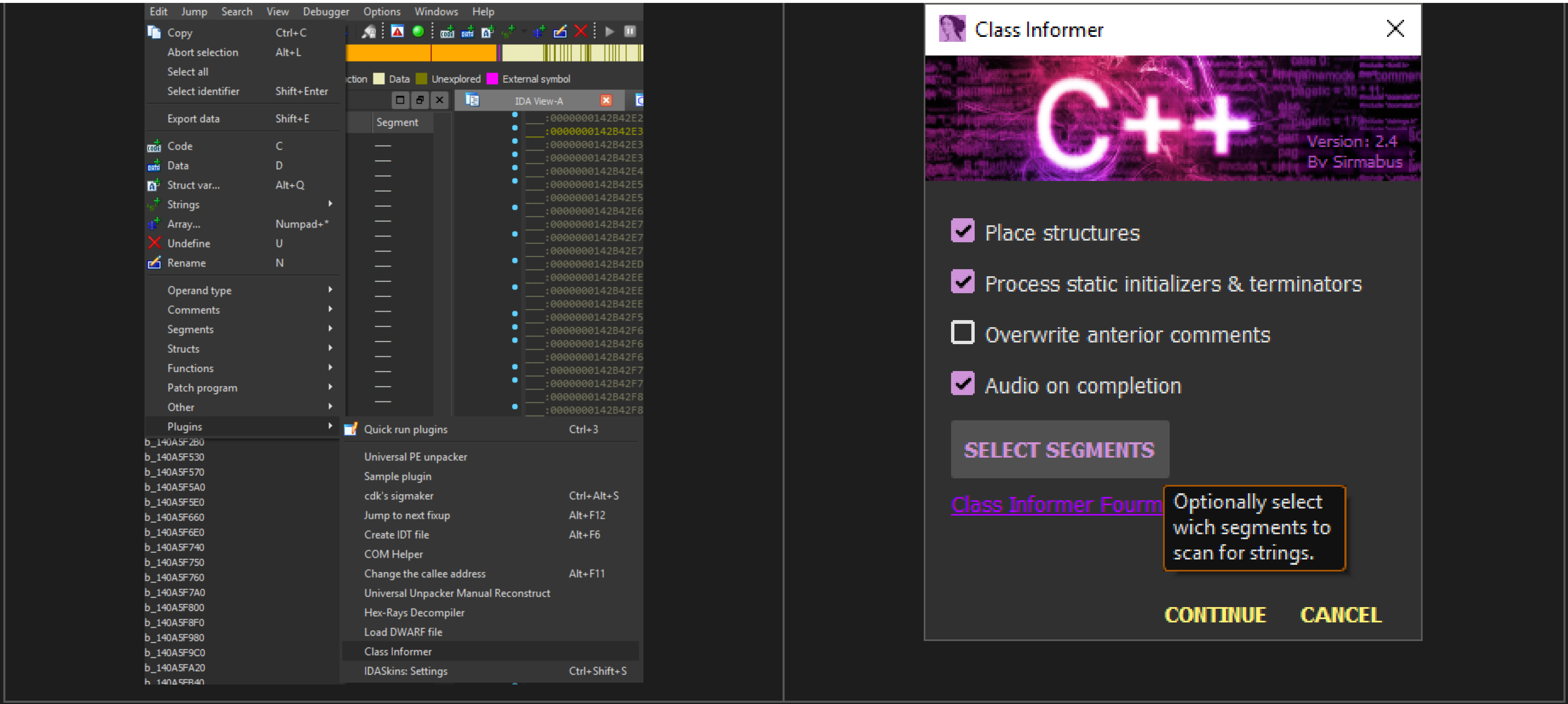
| List segments | Delete segment |
|---|---|
|  |  |

Fig.2/3 - IDA view segments / IDA delete segment

The reason this section exists in the dump is the fact that the executable was originally packed with Themida. This section is therefore a leftover from the unpacking mechanism, but is still marked as executable in the section headers, thus resulting in IDA treating it as code.

After we've deleted the segment, the auto-analysis should quickly redo and we will be greeted by a succesfull screen. When we work with foreign games and have explicitely handicapped ourselves by not using **any** public information, we will need to do basic reconnaissance on the respective game engine. We can usually identify crucial parts of either statically-linked dependencies or game engine internals by looking at strings, type-information or otherwise 'leaked' debug information. First, we'll look at the strings inside of the binary, by doing so, we can quickly determine if any type-information exists in any unofficial format that won't be recognized later on by *Class Informer*. By quickly glancing over the received information, we can determine that the game internally uses Lua. This is absolutely crucial as it allows for very explicit surveillance including function names, but we'll go over that in another post, as most games most likely won't use something as dogshit as Lua. The strings also revealed obvious function names that is most likely related to the Lua engine. Nothing else really jumped to us, so let's run *Class Informer*. To run *Class Informer*, go to *Edit->Plugins->Class Informer*

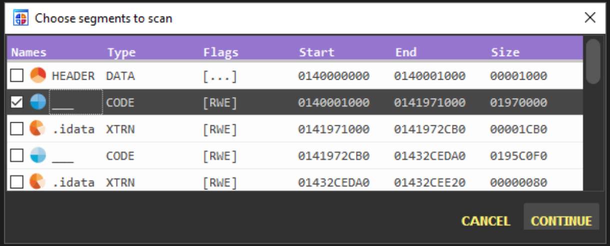
| Open Class Informer | Select segments |
|---|---|
| | |



- [✓] Place structures
- [✓] Process static initializers & terminators
- [ ] Overwrite anterior comments
- [✓] Audio on completion

Fig.4/5 - IDA open Class Informer / IDA Class Informer menu

Make sure to select the relevant sections, including the game code section as seen here:


Choose segments to scan

| | Names | Type | Flags | Start | End | Size |
|---|---|---|---|---|---|---|
| ☐ | HEADER | DATA | [...] | 0140000000 | 0140001000 | 00001000 |
| ✓ | ___ | CODE | [RWE] | 0140001000 | 0141971000 | 01970000 |
| ☐ | .idata | XTRN | [RWE] | 0141971000 | 0141972CB0 | 00001CB0 |
| ☐ | ___ | CODE | [RWE] | 0141972CB0 | 01432CEDA0 | 0195C0F0 |
| ☐ | .idata | XTRN | [RWE] | 01432CEDA0 | 01432CEE20 | 00000080 |

Fig.6 - Class Informer select segments

Hit *continue* and let *Class Informer* parse the runtime-type information, it shouldn't take more than a minute. After you hear the lovely text-to-speech of *Class Informer has completed*, you're ready for analysis! But, the rest of this series has not been written yet, so if you're willing to practice whilst we torture ourselves writing blog posts, we would suggest reading up on Lenas Reversing for Newbies from Tuts4You

Legal disclaimer

vmcall

vmcall
virtualmachinecall@gmail.com

vmcall
vm_call

vmcall's personal blog.