This section is not ready and the images are inaccurate. The DLL is being worked on as I develope this chapter. The DLL used won't be available until this chapter is done.

Before We Begin

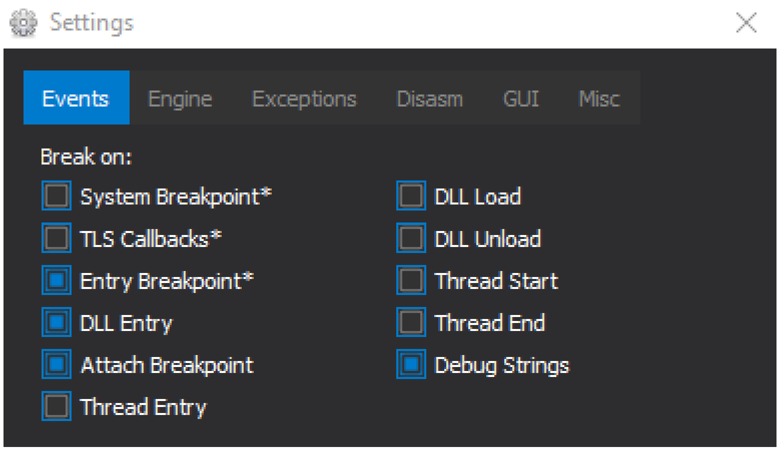
Why reverse engineer a DLL? Sometimes the documentation for a library isn't well written or written at all. If we want to use the DLL, we will need to know how it works. In some aspects, reversing DLLs can be easier than reversing executables because we have the names of the exported functions. In this chapter, we will not only reverse a DLL, but we will write a program that can use it.

Files Needed

All files used are in 0x000-IntroductionAndSetup/FilesNeeded

x64dbg DLL Loading

Make sure you have "DLL Entry" breakpoints enabled.



Break on:

- [ ] System Breakpoint*
- [ ] TLS Callbacks*
- [x] Entry Breakpoint*
- [x] DLL Entry
- [x] Attach Breakpoint
- [ ] Thread Entry
- [ ] DLL Load
- [ ] DLL Unload
- [ ] Thread Start
- [ ] Thread End
- [x] Debug Strings

DLLs cannot be executed like executables. To load the DLL, x64dbg will make a little program that will load it. The program will be called something like `DLLLoader64_#@@#.exe`. You shouldn't have to mess with the DLL loader, I just wanted to let you know so you didn't think you had malware or something on your computer. If x64dbg crashes, you will have to delete the loader executable manually.

xAnalyzer Issue / DLL Crashing x64dbg

I use xAnalyzer and I recommend you do too. When reversing the DLL in this chapter you will want to disable extended and automatic analysis or it will cause a crash. To disable it, start x64dbg and go to Plugins > xAnalyzer. Make sure there is no check mark next to "Extended Analysis" or "Automatic Analysis".