Project Boomerang

"Nostalgia is like a boomerang. It all just comes back ..."

1.11.2019

Danda

Dandaware

Melbourne

Story

Project Boomerang came to be due to me (Danda) searched through the GOG (Good Old Games) store, in search for a game to replace my desire to replay the original Mechwarrior 3 video game.
The concept and mechanics found within this game (Parkan: Iron Strategy) was a breath of fresh air, and I wanted to play more of it, if it were not for the crashes and incredibly low frame-rate. Thus, Project Boomerang was born.

Goals

1. Investigate and comprehend the Iron3D.dll
2. Reverse Engineer Iron3D.dll and develop a DLL Injector for run-time to optimise frame-rate issues and screen properties/dimensions.

Specifications

Employing multiple EXE and DLL softwares, to ensure consistency and to validate/confirm code found from the original resource.

Ghidra

Ghidra will be the main debugging and reversing tool for this project, due to how well developed this program is (developed by the American Government branch - the NSA) we can utilize this software for our reconnaissance.

X64DBG

This will provide additional assistance in confirming the reversal or review of a code, to ensure that Assembly registers and values match and are not incorrect/false.

Radare2

Radare2 (version 3.3.0) will allow us to do furthermore debugging and reversal of encrypted/digested values.

Visual Studio (IDE)

Develop the DLL here with the available DLL template and allow for shared solutions.

Milestones

I. Iron3D.dll Review and Reversal

Review, comprehend and compose a report as we reverse Iron3D.dll, and hopefully achieve 10KB per weekend or so.

II. DLL Injector

Develop a run-time DLL (Dynamic-link Library) Injector - optimising frame-rate and allow consistent updating of application window properties/dimensions.

III. SYS.LIB file re-use

Be able to unpack the SYS.LIB found in “Parkan - Iron Strategy” and implement the possible header files accompanied inside the package.

IV. Develop a re-purpose-able DTH

We are developing a run-time Dynamic-link Library, we need to be able to repurpose the project for different programming languages - although having at run-time may prove this to be ‘challenging’.

V. Enable project for Open-Source

For the final milestone, we release the project for open-source under the GNU License (version 3) — enable other open-source teams to commercialise this project for a living and further development of the Direct Tool Help.

Iron Direct Tool Help

Employing C++ we need to first develop a program which manipulates a snapshot of the system. Once we reverse the IRON3D.DLL, the snapshot DLL program will allow us to manipulate virtual memory, hopefully reproducing something close to a stutter fixer/remover when it comes down to the operations of the Thread · Memory · Queue & Heap with the use types such as HANDLE and ProcessID · ProcessEntry ... etcetera.

Later in the project, the design of the DLL will require Assembly code for the Read/Write operations of the program as well as the inputs and outputs of binaries — we must keep in mind the incorporation and integration of Assembly in mind when developing the DTH. (Although if this scope is forgotten, we can repurpose the code for straight C++)

With the use of the Windows ToolHelp32 header file, we can create snapshot(s) of the system, including the ability to manipulate virtual memory with the DLL template found in the Visual Studio (2017) IDE.

Resources

Tool Help 32 - Taking a Snapshot and Viewing Processes

https://docs.microsoft.com/en-us/windows/win32/toolhelp/taking-a-snapshot-and-viewing-processes

Creating a Simple Dynamic-link Library

https://docs.microsoft.com/en-us/windows/win32/dlls/creating-a-simple-dynamic-link-library