Grimoire 1.5

Team Grimoire

(Dated: February 21, 2008)

I. INTRODUCTION

Grimoire is a framework for building and executing warcraft and world editor hacks. Most of the functionality is implemented in injectable DLLs. Of these features every map developer will find something useful; Grimoire is not just for JASSers.

A. A Note on Safety

One concern with any hack is the damage they can cause to a warcraft installation. *Grimoire does not modify any installed files of warcraft.* All modifications happen in memory, at runtime. Thus Grimoire can not corrupt your warcraft installation, however, buggy tools could corrupt your map. This danger is always present with WE though so regardless of whether or not you choose to use Grimoire, back up your work frequently.

Due to the in memory modifications, I recommend against running Grimoire in ladder games on Battle.net. I have not heard any reports of folks getting banned through Grimoire use, but it's better to not take any chances.

II. START UP

Grimoire's functionality is managed through a collection of lua scripts. The following scripts control start up behavior.

- findpath.lua - This script locates the warcraft installation path via the registry. If it doesn't find it, you will be prompted to browse to your warcraft installation. That path will then be saved in the registry.
- war3.lua - This script is used to start war3.
- war3.conf.lua - This script determines which DLLs to load into war3. At the moment the only DLL you will wish to load is ongameload.dll, which waits for game.dll (Blizzard) to load into warcraft, then runs ongameload.lua.
- ongameload.lua - Here you can choose DLLs to inject into war3 once game.dll has loaded.
- we.lua - This script is used to start WE.
- we.conf.lua - This script determines which DLLs to load into WE.

III. FUNCTIONALITY

Abstractly, I divide Grimoire's functionality into three classes:

- Bug fixes to War3 / WE
- New features for War3 / WE

- Gateways for driving third party tools with WE

The features are implemented in DLL libraries which can be found in bin/ . When you start War3 or the WE through Grimoire, the libraries you choose are injected into the process by exehack.exe, where they replace bits and pieces of Blizzard code with chunks of themselves.

IV. WEHACK.DLL

Wehack.dll fixes bugs and extends the World Editor. It is enabled by default via the line 'grim.injectdll(proc,"bin//wehack.dll")' in we.conf.lua. Once it loads it executes wehack.lua which controls most of the functionality. All menu items are driven by lua. To add menu entries, see the examples and the class definitions for MenuEntry in wehacklib.lua. A number of functions can be called from wehack.dll.

- clickevent(id) - this is the event handler. At the moment it is used only to dispatch messages to menu entry owners.
- testmap(cmdline) - this function is run when the World Editor attempts to start Warcraft via the test map button. cmdline is that passed to CreateProcess.
- compilemap() - this function is run when the map is saved. Insert code for processors in compilemap_path(mappath).

Wehack creates logs//wehack.txt and logs//welog.txt in which it stores debug info.

A. Menu Entries

Wehack adds a menu titled ”Grimoire” to the main world editor window. This menu contains these items:

- Start War3 with -window - check this to have warcraft start up in windowed mode on test map
- Start War3 with -opengl - check this to have warcraft start up in OpenGL rather than DirectX
- Start War3 with Grimoire - run warcraft with war3.lua and the DLLs it lists. If this is unchecked, warcraft will be start up in a pristine and safe mode.
- Enable no limits - disable Doodad, Unit, Item limits. Increase max map size to 480x480 from 256x256.
- Enable object editor hack - Allows customizing the rawcode of created and pasted objects in the object editor when enabled. It will also display the meta data rawcode of object editor fields in brackets.
- Disable WE Syntax Checker - This prevents the WE checker from running and crashing. You can also use this to save time when saving if you are working on non script related tasks.
- Don't let WE disable triggers - Prevent WE from automatically disabling triggers
- Always allow trigger enable - Allows you to enable a trigger regardless of what WE thinks
- Mute editor sounds - Turns off the startup sound, the undo/redo sound, and the sound that it makes when placing/deleting objects

- Enable war3err - Allows ongameload.lua to inject war3err.dll when you test map with Grimoire.
- Disable first save warning - When a map is saved for the first time or a Grimex tool is used on unsaved maps, a popup will tell you to save your map again. This popup will be disabled with this option.
- Integrate WEU - When a WE Unlimited installation is detected on your system this menu item will appear. When enabled it will integrate all the WEU features in Grimoire. However, there is not much point in using WEU nowadays.
- Customize test map settings - Opens up a configuration dialog where you can change the game speed, player details and other test map related options.

V. WAR3ERR.DLL

War3err.dll is geared towards debugging warcraft. Enable it by checking the war3err menu entry in the Grimoire menu that wehack adds to the World Editor.

A. VM Errors

Warcraft has a handful of errors that the VM detects and silenty dies on. War3err hooks these and reports them with a stack trace. The stack trace lists the function calls and arguments that led to the error.

- Op Limit - Execute 300,000 bytecode operations

- Divide by zero - Divide by integer or real 0
- Var use - Attempt to get the value of an uninitialized var
- ExecuteFunc(”func”) - If func does not exist or func has arguments, this stops a crash.
- Player(x) - If $x > 15$ or $x < 0$

B. Debugger

War3err contains a debugging engine. You can connect to it with two clients: w3jdebug.exe and pyw3jdebug.py. W3JDebug.exe is a console utility which allows you to write commands to the debugger and read them directly. Most people will want to use the GUI, pyW3JDebug.py. This requires Python, wxPython and PythonCard. To use the debugger, enable it in war3err.lua with ”debugger = true”. Then, set a breakpoint somewhere in your code by adding a call to ’Cheat(”war3err_Break”)’. When JASS execution reaches this point, warcraft will freeze and wait for you to attach a debugger. At this point (and not before) run pyW3JDebug.py.

C. Bytecode Tracer

JASS compiles down to bytecode executed by a virtual machine. The bytecode consists of operations such as adding two numbers or calling a function. Enable the bytecode tracer with the line ”bytecodetrace = true” and each operation that passes through the VM will be recorded to logs//bytecodelog.txt. This is useful for determining where crashes occur, as you can scan up to find the function that logging that stopped.

D. Cheat Commands

Several commands can be sent to war3err by running 'Cheat("command")'. These are:

- war3err_DumpGlobalHT - prints the names and values of all global variables to logs//war3err.txt
- war3err_DumpLocalHT - As above, but for the local variables of the current scope
- DebugMsg: Your Message - Echoes Your Message to logs//war3erruser.txt
- PauseTracer: If bytecode logging is enabled, this pauses recording
- ContinueTracer: If bytecode logging is enabled, this resumes recording

E. Memory Protection

War3err traces allocation of locations and groups. It keeps tracks of attempted double frees, free(null) and leaks. Leak statistic reports can be extracted by call RemoveLocation(ItoLoc(-1)) and call DestroyGroup(ItoGrp(-1)).

F. Profiling

Enable the profiler by setting "profiling = true" in war3err.lua. Add an empty function "LogJASSCalls" to your script and a variable "LogVariableUse". Call LogJASSCalls() will now report statistics on call frequency of JASS functions while getting or setting LogVariableUse will report statistics on variable use.

G. No Pause

Set "nopause = true" in war3err.lua to prevent warcraft from pausing when the window loses focus. Does not work in replays.

War3err features can be configured via war3err.lua in the main Grimoire directory.

VI. LISTFILE.DLL

Inject this DLL into Warcraft or the World Editor to list all the files that the World Editor tries to load out of MPQs.

VII. JAPI.DLL

JAPI allows you to write your own natives. Inject JAPI.dll for the engine. See nativepack.cpp for example natives. Each native that you add must have a prototype in common.j.

VIII. LOADMPQ.DLL

This lets you inject MPQs into Warcraft or the World Editor at higher priority than defaults. You can use this for tricks like multiple cliff tiles.