Corrupting Bug Fables and other Unity games

# Jack7D1[206920373952970753], May 2021

**Abstract:**

This paper will discuss the process of corrupting Unity games using the Real Time Corrupter (RTC) and provide documentation of methods and techniques that may be used. This paper largely serves as documentation and as a guide for consequent Unity corruptions. This paper pertains to the use of the RTC and it’s Process Stub and largely to the corruption of Bug Fables. However techniques pre- sented therein have been observed to be helpful in corrupting a wide variety of modern Unity games.

# Preface:

**WARNING:** Corrupting programs is likely to cause **LOUD** noises and **FLASHING LIGHTS** from the corrupted program. Assume that corrupting a program will result in **DATA LOSS** for said program, it should be assumed that corruption of a Unity game is destructive in nature in every case.

Take this into consideration before corrupting any program.

# Tools Used:

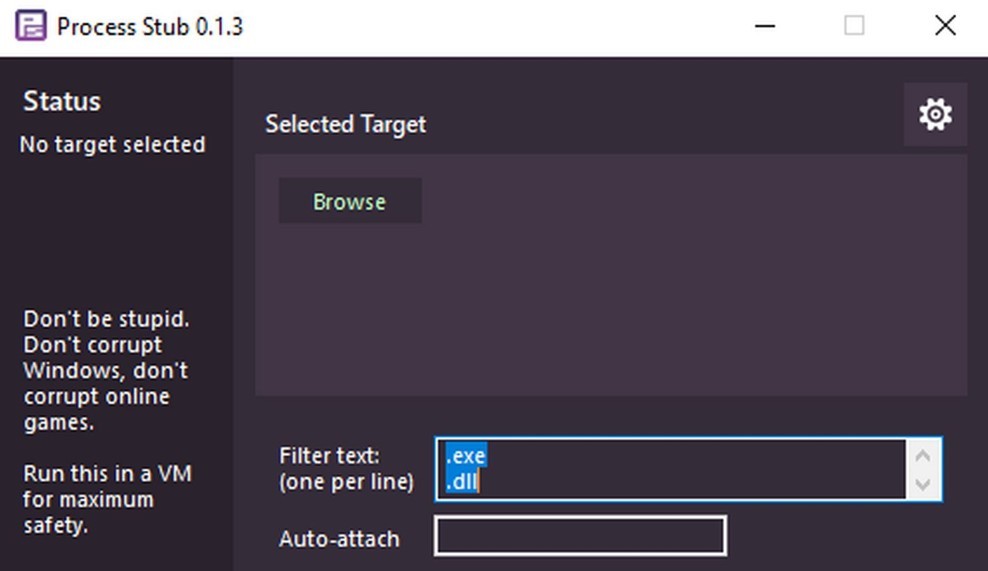
It has become experimentally apparent that the RTC is best suited for modern, complex games such as Bug Fables. As such this documentation pertains to the use of the RTC with the Process Stub enabled.

Download and install the RTC online. This portion is currently well documented on the aforementioned website and is therefore beyond the scope of this documentation.[1](#_bookmark0)

# Setup:

Begin by opening the RTC Launcher.

*Figure 1: RTC Launcher after successful start*

Ensure that ProcessStub is installed, then click on it’s icon to open the corrupter.

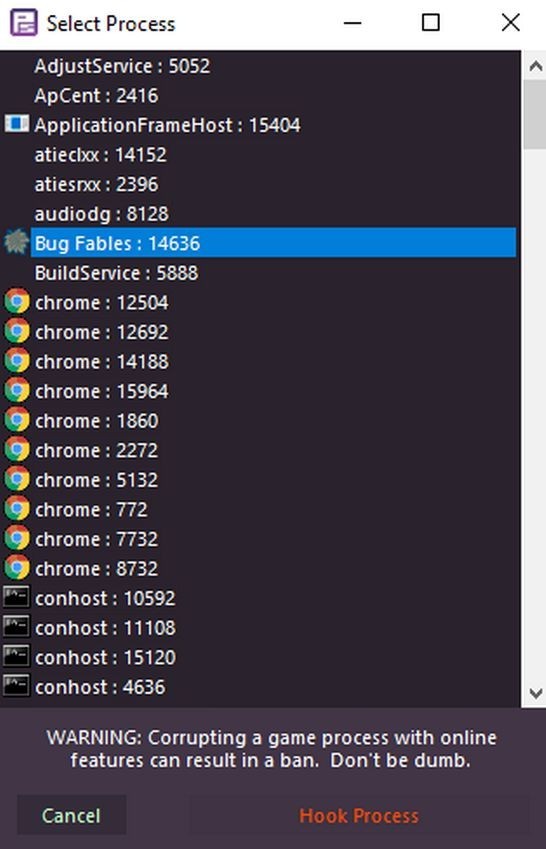
It is imperitive to heed all warnings pre- sented by the tools. However the use of a Virtual Machine (VM) is generally not required as Windows provides ample security against collateral damage from process failure.

*Figure 2: Initial configuration of a fresh started Process*

*Stub*

1. See Extra Resources #1

It is important that the target game is fully started and in the run state before proceeding.[2](#_bookmark1)[3](#_bookmark2) At- taching the stub prior to completely loading the game may result in the game hanging or otherwise fail- ing to start or load, as such the use of Auto-attach is highly discouraged.[4](#_bookmark3) In the game Bug Fables this is achieved by running the game, awaiting the main menu and then loading/creating a save. Once all loading is completed the process stub may be attached.

Attaching the stub can be done simply. Click Browse and select the game in the resulting menu.

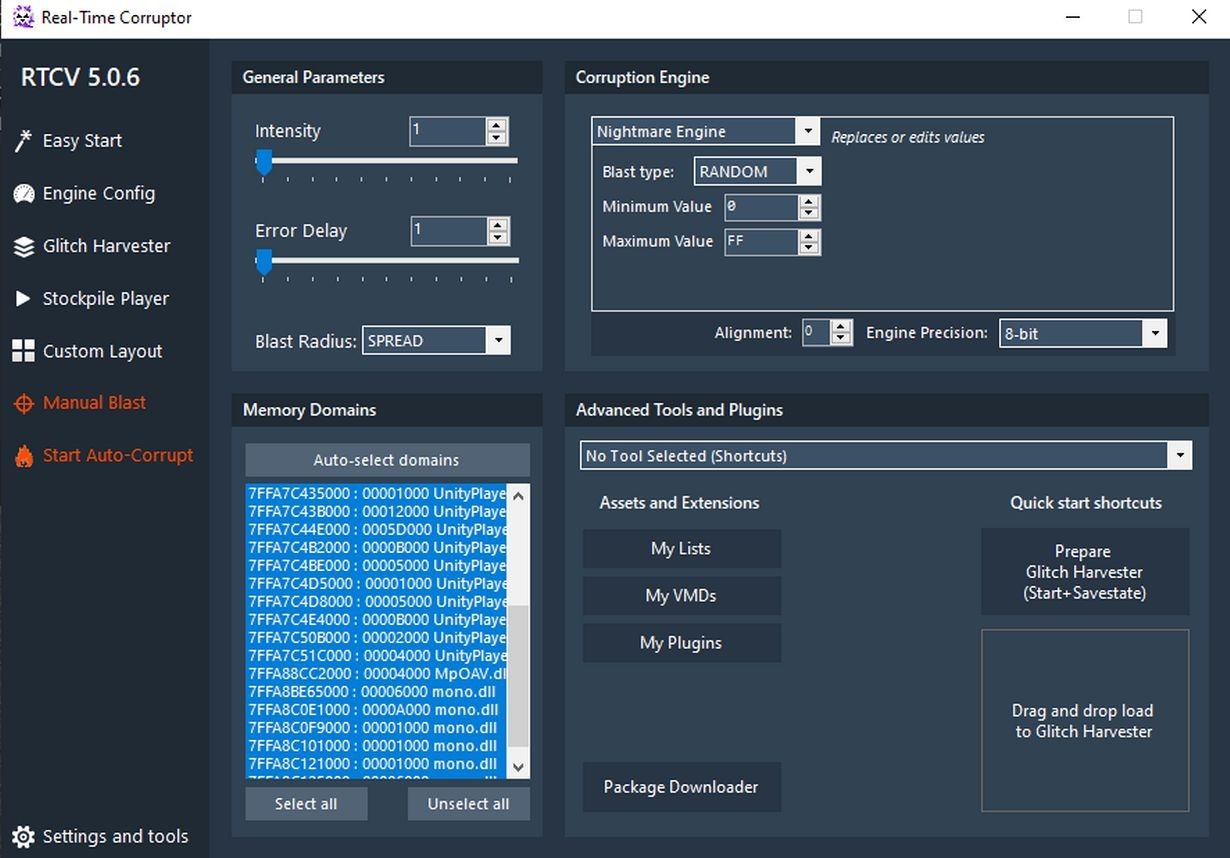
This Figure is largely an example, and appearance can vary widely. In this scenario the desired program is:

“Bug Fables : 14636”

The number following the process name is the Process ID (PID) and can be ignored for this purpose, however may prove useful for cross identification with a Command Line Interface.

Once the desired game is selected click Hook Process to proceed.

*Figure 3: Process Stub Browse Window, Bug Fables selected.*

If the corrupter window does not look like this ensure it is not in Easy mode by Clicking the ‘Normal Mode’ button in the bottom right.

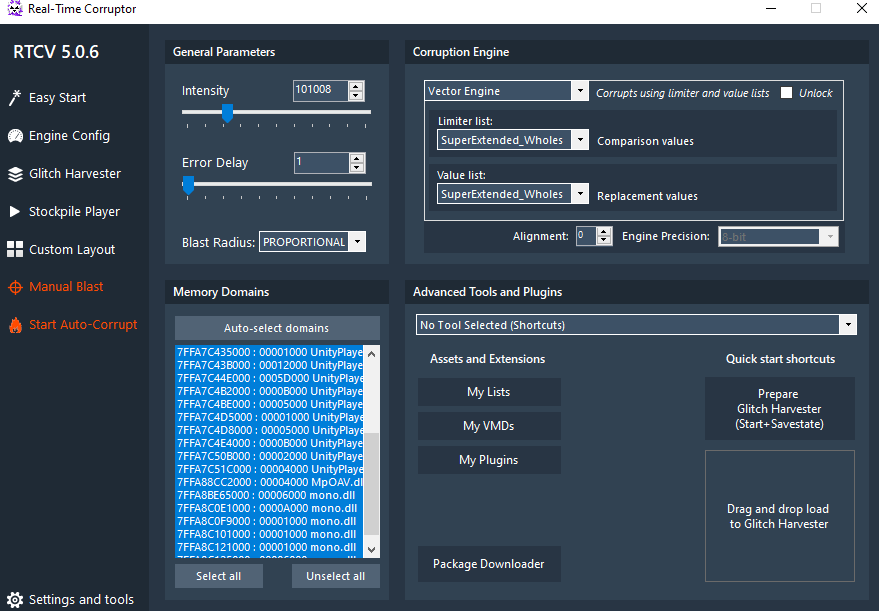
The Addresses in the Memory Domains section can vary greatly on circumstance and are largely non-useful for the purpose at hand.

Configure the Corruption Engine to Vec- tor Engine and the Blast Radius to Pro- portional.

*Figure 4: RTC window after successful attach with Bug Fables.*

1. Unity performs most all file integrity checks during program initialization, the use of Process Stub during runtime serves to bypass these checks.
2. Hooking prior to full load will result in malformed address spaces as the data block domains are too small
3. Auto attach uses the window title as a discriminator.

The window should now look like this.



At this time setup is complete and a corruption can be properly applied.

# Corrupting:

The proper use of Memory Domains is important for successful corrupting, each dll and data block is considered a domain, however they should be organized into two “Super-Domains”.

This table has been determined experimentally.

|  |  |  |
| --- | --- | --- |
| Super-Domain Name | Dll | Resources |
| Filter text: | .dll | “UNKNOWN” |
| Corruption effects: | Physics, loading zones, base logic, nu- merical handling, events and story pro- gression, movement. | Sprites and textures, sounds and music, normal maps, lighting maps, animations, text, events, level geometry, movement. |
| File Domains: | All .dll files. | “data.unity3d” or ∑ ”level###” files |
| Stability: | Exceptionally touchy, responsible for all computation, corruption is likely to lead to crashes, however opens many corrup- tion possibilities. | Very stable, to the point of re- quiring a large bolus of intensity for observable effect to be no- ticed. |
| Suggested Blast Radius: | SPREAD, EVEN, PROPORTIONAL | PROPORTIONAL, BURST, NORMALIZED |
| Suggested Intensity: | ~2000 | ~100000 |
| Mode Size: | 0x1000 bytes | 0x401000 bytes |

Corrupting both super-domains simultaneously is highly unstable and a proper method had not been found.

Switching between them can be done by entering the filter text of the target super domain into the filter text entry of the Process Stub.

Experimentation is highly encouraged by excluding certain domains via deselection or by adjusting in- tensity, blast radius, limiter list, and value list.

To apply the corruption click Manuel Blast, the corruption should now be applied. If a success then this guide is complete and may be repeated for an additional corruption.[5](#_bookmark4)

If the game crashes upon application then consider reducing the intensity, changing other settings or simply trying again.

1. Manual blasts stack, however additional corruptions exponentially increase the risk of crash.

# Extra Resources:

<https://redscientist.com/rtc> For aquiring the RTC.

<https://corrupt.wiki/corruptors/rtc>

Further guides and documentation that are outside of the scope of this paper.