HeapTracer plugin for IDA
This will let you see the evolution of the heap in realtime.

INSTALLING

- you'll need OpenGL installed and working on your Windows box

a software implementation is installed by default.
for hardware acceleration you'll need something specific for your card

- copy debug/HeapTracer.plw to IDA/plugins
- copy HeapTracer/glut32.dll to IDA

USING

Specific to IDA

- In the configuation window (Ctrl-Shift-1 by default):

turn on heap tracing
turn on the display window
select the heap to draw (yes, you have to know the base address/handle)
if you don't know the address, you can select it latter
once the application is running, every time you enter the config window you'll see a list of the known heaps with a count of the blocks in heach of them. You can change the heap to draw at any time.
If you attach to a process, you can choose to rescan the heap.

For other usage see main README

BUILDING

To build you'll need:

- Visual Studion Express Edition

get it free from http://msdn.microsoft.com/vstudio/express/

- idasdk (we used 5.1, but 5.0 is also fine).
- change settings for the project

point the include and library directories to your idasdk copy change the custom build step to copy the plugin to your IDA directory (or not)

- if you want to debug the plugin, you'll have to change where IDAg.exe is in project's settings in Visual Studio
- enjoy

It's also possible to build from the command line using the included makefile. You'll also need to do some changes in order for it to work in your setup. When using the makefile, place the folder HeapTracer inside idasdk/plugins

FEEDBACK

- Don't let all that people who said that OpenSource doesn't work be right: send feedback, bug reports, ideas, patches, improvements, etc.

KNOWN ISSUES

- uhm... some multithreading issues has been solved in this version, and it usually works pretty well... except sometimes, that it just blocks the hole IDA. If you can find the bug, please, report it back to us!
- For some strange reason IDA, the OpenGL windows and the application fight for user's attention, stealing the focus from each other, and giving it back to nobody... Again, if you understand why this is happening and you can fix it, please, share!
- There is a race condition when two threads are inside, lets say, RtlHeapAlloc() at the same time. The return address for one thread can get mixed up with the return address with the other thread, hence adding the new chunk to the wrong Heap. This can be fixed pretty easily by tracking the thread id together with all the other info in the breakpoints array. We'll do it at some point in the future. If you do it yourself, please... share!

TODO

- If you want to contribute and need some ideas, take a look at HeapTracer.cpp at the top of it you can find a nice list of what to do :-)

enjoy and share!
gera