

# Debugger slides

Goes over x64dbg. Should be used with fasm.

# Before you start

You should know that a debugger is meant to show you an instruction by instruction representation. The debugger allows you to look at the individual registers and the memory elements.

Know that x32dbg or x64dbg are two different versions of the debugger. If you see x96dbg then you be prompted for which one you want. X32dbg is the 32 bit version meant for 32 bit files and x64dbg is the 64 bit version meant for 64 bit files.

**You can find the download links for x64dbg in the Assembly text file with the links.**

# Using a file

You can choose a file by pressing F3, pressing the button under the File menu (  ) by going to File -> Open or by dragging and dropping a file and then selecting the file that you want to open.

When you start you will be in the cpu view. The left middle portion will be your asm code and the right slide will be all types of registers and the rflags/eflags.

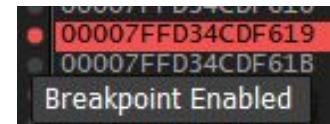
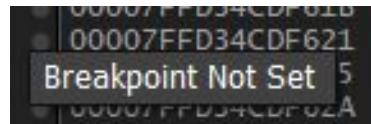
The bottom left and right have a list of memory dumps (representations of the binary/machine code of your program) and a list of essential memory addresses.

When you start interacting with your program you will probably not recognize the program because it is likely to be the windows code required to make it a program that runs with windows.

# Jumping to points

When this happens you look at the buttons on under the file, view, etc. menus. Under debug you will see an arrow (➡). This will allow you to run the program to different stopping points such as breakpoints (artificial stops) or files such as the gdi32 binary. If want to go to the main point of the file immediately under the options menu click the arrow next to a person button (➡).

You can make a breakpoint that stops a file at a certain point. This allows you to immediately check the code after it. In order to make a breakpoint you click on the dot next to the memory addresses. When a memory address is set it is red.



## Jumping to points ext.

You can go through individual instructions by pressing the button under the tracing menu (  ). **As you do this you can see the changes in registers and conditional jumps play out.**

Breakpoints are saved between when you close a file in x64dbg. You can do that with the square button under the view menu (  ). Closing a file stops the file from running and clears all of the subviewers (i.e. The dumps, registers, etc.)

It is recommended that you use x64dbg to check dataflow(how your program runs through the jumps) and gain insight into your registers.

## Additions to these slides

You can work on improving these slides by looking at the panes such as the breakpoint and the threads pane and go into detail into how you would use each one.

If you do go about this then you must request it with a pull request specifying that you want to go about doing this project. For more information go to the Individual Projects readme.

The slides after this one can be used as examples but will be extended as more work is put into these slides.

# Advanced Breakpoints

You can see the active and disabled (breakpoints that don't do anything) breakpoints in the breakpoints pane (  ). You can also go to view -> breakpoints. By right clicking on a specific breakpoint you can enable, disable (the breakpoint remains but won't do anything), remove, set conditional breakpoints and more.

Hit count allows you to customize how many time the breakpoint is hit before actually going into effect.

Conditional breakpoints allow you to customize when a breakpoint is hit. These breakpoints have c-style conditionals.

## Advanced Breakpoints ext.

You can program the conditions with registers and memory values

<Learn how to write a conditional breakpoint all the way>

<https://help.x64dbg.com/en/latest/commands/breakpoint-control/index.html>

# Threads

You can see a list of threads with a lot of useful information under the threads pane (  Threads ). You can also go to view -> Threads.

You can turn