Contents

Introduction

BugDbg x64 is a user-land debugger designed to debug native 64-bit applications.

BugDbg is released as Freeware.

History

version 0.7.5

- added possibility to change Rip line color
- improved jumps, loops destination drawing

version 0.7.4

- added “highlight current word” feature

version 0.7.3

- added jumps, loops destination drawing area to dissasembly window

version 0.7.2

- added Jumps and Calls highlighter

version 0.7.1

- added assemble command “a”
- added File - “Save patched executable as...” option
- added Configuration - Options - Misc page
- improvements, bug fixes

Warranty

This software is provided as-is, without warranty of ANY KIND, either expressed or implied, including but not limited to the implied warranties of merchantability and/or fitness for a particular purpose.
The author shall NOT be held liable for ANY damage to you, your computer, or to anyone or anything else, that may result from its use, or misuse.

Use it at YOUR OWN RISK

Credits

This software uses:

Qt 4.8.2

diStorm v3 - http://www.ragestorm.net/distorm/

NASM 2.10.07- http://www.nasm.us/

Contact

In case of bugs, problems, feature request, etc. feel free to contact me.

Regards,
C'Bob

bugdbg@gmail.com
www.pespin.com

Keyboard shortcuts

Disassembly Window

Navigation

Jump to previous view _____ **Esc**
Jump to next view _____ **`**
Jump to address _____ **g**

Focus command line_____ **F1**

Debugger

Insert or remove breakpoint_____ **F2**

Application shortcuts

Toggle hex code on/off _____ **F6**

Command reference

~

Syntax: ~

Description: list threads.

a

Syntax: a <address>

Description: assembles instruction mnemonics and copies the resulting opcodes into desired address.
This command uses Nasm assembler. Set path to nasm.exe (Configuration - Options – Misc)
prior to use.

Example:
a rip
a kernel32_IsDebuggerPresent
a 0000000140001020

after typing the command a dialogue where you can write assembly code will pop up, this allows to write more than one instruction at once. When pressed OK code will compiled with NASM and copy directly into memory.
Additionally, if the patch is inside target executable imagebase it's possible to save changes File - "Save patched executable as.."

ba

Syntax: ba <rw | w | e>size <address>

Description: set hardware breakpoint on access (on data execute, write, read/write).

Monitored address can be a range of 1, 2, 4, or 8 bytes, except breakpoint on execution which must be 1 byte.

Example:

ba w4 0000000140001040

ba e1 0000000140001010

ba rw8 0000000140001020

ba r8 0000000140001020 *(the same as ba rw8 command)*

bc

Syntax: bc <index | *>

Description: clear breakpoint at given index. The find the index use *bl* command.

Example:

bc 1
bc *

bd

Syntax: bd <index | *>

Description: disable breakpoint at given index. The find the index use *bl* command.

be

Syntax: be <index | *>

Description: enable breakpoint at given index. The find the index use *bl* command.

bl

Syntax: bl

Description: list breakpoints

bp

Syntax: bp <address>
bp <address> if <condition>

Description: set software breakpoint on execution

Examples:

bp 0000000140001040

bp kernel32_CreateFileW if rcx == L"C:\test\data.bin"
Break if CreateFileW - lpFileName equals "C:\test\data.bin". Note the L"" that makes string wide.

bp kernel32_CreateFileA if rcx == "C:\test\data.bin"
The same for ANSI version.

Be aware string inside "" or L"" is case sensitive.

To step out of some obfuscated loops it might be useful to set conditional bp that will break only when specific register has a desired value, e.g.

bp rip if ecx == 1

Other examples:

bp 0000000140002018 if [0x0000000140004000] >= 10
bp rip if rax+500 == 1000
bp 0000000140001120 if [rax+8] == 1000

bpdll

Syntax: bpdll <dll path>

Description: sets breakpoint on dll load (entrypoint).

Example:

bpdll c:\windows\system32\psapi.dll

To break on all dll's loaded from c:\program files\ use the command:
bpdll c:\program files*

To break on all dll's ending with deadbeef.dll use the command:
bpdll *deadbeef.dll

To print the target dll use the command:
bpdll ?

To clear the dll path use the command:
bpdll

bpinstr

Syntax: bpinstr <max number of instructions to trace in hex> <instruction-mnemonic>

Description: breakpoint on instruction, debugger will break before instruction has been executed or when max number of instruction to traced has been reached. Instruction-mnemonic is case-sensitive. It's possible to use wild-card but it must be at the end.

Example:

bpinstr 1000 "MOV R8, 0xbadbeefdeadc0de"

bpinstr 1000 "MOV [RSP+0x20], R11W"

bpinstr 5000 "MOV [RSP+0x20], *"

bpinstr 5000 "J*" *(this will pause execution on any type of jump)*

cls

Syntax: cls

Description: clears command window

code

Syntax: code <on | off> < number of spaces if code off is used by default its 1>

Description: show hexadecimal bytes of an instruction along side disassembled code.

Application shortcuts: *F6*

Example:

code off 4

createthread

Syntax: createthread <address> <parameter (optional) >

Description: creates a new thread in the target process. This command might be more useful with readmem command.

Example:

createThread 0000000140001000 0xbadbeef

detach

Syntax: detach

Description: detach debugger from the process, but keeps the target application running.

go

Syntax: go

Description: will start or resume code execution if its paused.

help

Syntax: help

Description: shows list of commands.

inject

Syntax: inject <dll file name>

Description: injects dll into debugged process.
File name path cannot contain spaces e.g. ~~c:\Program Files\hidedebugger.dll~~

Example:

inject c:\test\hideDebug.dll

r

Syntax: r <register> <value>

Description: changes the register value.

Example:

r rsi 123456789a
r ah 12
r zf 1

rr

Syntax: rr

Description: prints all 64-bit registers.

readmem

Syntax: readmem <base address - optional> <file name>

Description: reads a file into targets process memory space. The memory protection for the newly allocated pages will be set to PAGE_EXECUTE_READWRITE.

File name path cannot contain spaces e.g. ~~c:\Program Files\shellcode.bin~~

Example:

readmem c:\test\asm.bin

Allocated memory address will be displayed in command window, if readmem is succesful.

readmem 0000000140001040 c:\test\asm.bin

u

Syntax: u <address | symbol | register64>

Description: Unassemble instructions at address, symbol or 64-bit register. The same as pressing **g** in disassembly window.

writemem

Syntax: writemem <address> l <lenght> <file name>

Description: writes memory to a file.

File name path cannot contain spaces e.g. ~~c:\Program Files\dump_mem.bin~~

Example:

writemem 0000000140001000 L 1000 c:\users\u\dump.bin

x

Syntax: x module_symbol

x module_*

Description: displays the symbols address that starts with the specified pattern (eXamine).

Example:

x kernel32_Beep

x kernel32_Be*

x *_Beep

x ntdll_*