PwnDBG / gdb v0.1

Examination

Disassembly
- x/[num]i [function_name | memaddr] — Examine -num- of instruction starting from memaddr/func
- disassemble [function_name | memaddr]
disass [function_name | memaddr] — Disassemble function/memaddr
- disass /r [function_name | memaddr] — Disassemble function/memaddr with opcodes

Mem Address Information
- xinfo [memaddr] — Extended info for virtual addr.

Data
- x/[num]bx [memaddr] — Examine 1 byte blocks as hex
- x/[num]hx [memaddr] — Examine 2 bytes blocks as hex
- x/[num]wx [memaddr] — Examine 4 bytes blocks as hex
- x/[num]gx [memaddr] — Examine 8 bytes blocks as hex
- x/[num]s [memaddr] — Examine strings
- x/[num][b | h | w | g] [memaddr] — Examine as signed decimal
- x/[num][b | h | w | g]u [memaddr] — Examine as unsigned int
- x/[num][b | h | w | g]f [memaddr] — Examine as float

Analysis

ELF Analysis
- elfheader — Print the section mappings
- got — Global Offset Table
- plt — Print symbols in the .plt section
- gotplt — Print symbols in .gotplt (if exists)

Exploitation
- canary — Prints Stack Cookies
- rop — Rop analysis (ROPgadget)
- ropper — Rop analysis (ropper)
- checksec — Check the security proper. of the binary
- show disable-randomization aslr — Show ASLR setting for proc.
- set disable-randomization [on|off] — Turns ASLR OFF/ON (proc. Restart needed)

Registers
- info registers
i r — Show all registers
- info registers ${reg} ${reg2} ...
i r ${reg} — Single / multiple registers

Memory Search
- find /b [f_start-memaddr], [f_end-memaddr], [f_bytes_to_find]
find /h [f_start-memaddr], [f_end-memaddr], [f_halfword_to_find]
find /w [f_start-memaddr], [f_end-memaddr], [f_word_to_find]
find /w [f_start-memaddr], [f_end-memaddr], [f_giant_to_find] — - Byte
- Half-Word
- Word
- Giant
- find [f_start-memaddr], [f_end-memaddr], "[f_str_to_find]" — '0x00' Terminated String

Process Memory
- maintenance info sections — Mapped sections
- info proc mappings — Mapped address spaces
- xinfo [memaddr] — Virtual Address information
- vmmap — Virtual Memory Map

Stack
- stack [count_s] [offset_s] — Print Stack
- info frame — Current frame
- info frame level [level] — Frame by level
- backtrace
bt — Backtrace
- backtrace full
bt full — Full backtrace (local var also)
- retaddr — Stack addr. containing Return Addresses

Heap
- heap — Prints chunks
- arenas — Prints allocated arenas
- fastbins — Prints fastbins
- smallbins — Prints smallbins
- largebins — Prints largebins
- top_chunk — Prints Top Chunk
- tchachebins — Prints current thread tcache
- vis_heap_chunks — Visualize heap chunks
- bins — Prints out the contents of the:
- tcachebins
- fastbins
- unsortedbin
- smallbins
- largebins

Values:
- [arena_addr] - (hex) memory address of the arena in hex format (i.e. 0x08043000)
- [b | h | w | g] - (char in list) 'b' to examine as byte, 'h' for half word (16 bits), 'w' for word (32 bits), 'g' for giant (64 bits)
- [breakpt_id] - (integer) breakpoint id (i.e. 1, 2, 40, etc.) that you already set, returned by pwndbg
- [bp_condition] - (string) condition to match (i.e. $eax == 0, *0x0804556c != 0, etc.)
- [count_s] - (integer) Number of element of the stack to dump
- [f_start-memaddr] [f_end-memaddr] - (hex) memory addresses to use as start and end for memory search in hex format (i.e. 0x08043000)
- [f_bytes_to_find] - (hex) byte or sequence of bytes to search (comma separated). For example 0x11, 0x22, etc. See Examples
- [f_halfword_to_find] - (hex) half word (16bits) or sequence of half words to search (comma separated). For example 0x1122
- [f_word_to_find] - (hex) word (32bits) or sequence of words to search (comma separated). For example 0x11223344
- [f_giant_to_find] - (hex) giant (64bits) or sequence of words to search (comma separated). For example 0x1122334455667788
- [function_name] - (string) Function name (i.e. main, printf, ecc.)
- [function_name | memaddr] - (strings) Function name or Memory address (i.e. main, 0x08040404)
- [level] - (integer) Stack Frame level (0 is the current one)
- [memaddress] - (hex) memory address such as 0x08041b5c
- *[memaddress] - (hex) memory address in the format *0x08041b5c
- [mem_value] - (signed int | hex | char | string) Value for memory (i.e. 0x1337, 1, -1, 0xFFFFFFFF, 'A', "Str1ng__", etc.)
- [num] - (unsigned integer) Number of instructions (i.e. x/10i - 10 instructions), bytes, word, dword, etc.
- [num_of_bytes] - (unsigned int) Number of bytes to dump
- [offset_s] - (integer) Element offset from $sp
- [regex] - (string) Regular Expression, value depends on expected function/library name, etc.
- [reg] - (string) Name of a register (i.e. eax, ax, ah, al, ebx, etc.)
- [reg_value] - (signed int | hex | char) Value for register (i.e. 0x1337, 1, -1, 0xFFFFFFFF, 'A', etc.)
- [signal_action] - (string in list) 'nostop' | 'stop' | 'noprint' | 'pass' | 'ignore' | 'noignore'
- [signal_name] - (string) signal name (i.e. SIGSEGV, SIGKILL, etc.)
- [syscall_number] - (unsigned integer) number of a specific syscall (i.e. 0 for x86 Read, 1 for x86 Write, etc.)
- [syscall_name] - (string) Name of a specific syscall (i.e. 'read', 'write', 'socket', 'fork', etc.)

Signals

Info
- info handle
info signals
info signals [signal_name] — Shows Signals / Handles

Handling
- catch signal all
catch signal [signal_name] — Catch signals
- handle signal [signal_action] — Action on catch for signal

Examples

Registers/Memory Analysis (Little Endian x86 ELF)
```
0xffffd4b4: 0x54 0xd5 0xff 0xff 0x5c 0xd5 0xff 0xff

pwndbg> x/4bx 0xffffd4b4
0xffffd4b4: 0x54 0xd5 0xff 0xff
pwndbg> x/bd 0xffffd4b4
0xffffd4b4: 84
pwndbg> x/gx 0xffffd4b4
0xffffd4b4: 0xffffd55cffffd554
pwndbg> info line main
No line number information available for address 0x80485a4 <main>
pwndbg> i r $eax
eax            0x804a020          134520864
pwndbg> i r $ax
ax             0xa020             -24544
pwndbg> i r $al
al             0x20               32
pwndbg> set $ax=0x1337
pwndbg> i r $eax
eax            0x8041337          134484791
pwndbg> i r $ebx $ecx $esp
ebx            0x0                0
ecx            0x3356e5c4         861332932
esp            0xffffd4a8         0xffffd4a8
```

Search Patterns in Memory (Little Endian x86 ELF)
```
0x8048000: 0x54 0x68 0x69 0x73 0x20 0x69 0x73 0x20
0x8048008: 0x61 0x6e 0x20 0x65 0x78 0x61 0x6d 0x70
0x8048010: 0x6c 0x65 0x00 0x00
0x8048000: "This is an example"

pwndbg> find /b 0x8048000, 0x8048013, 'e','x','a','m'
0x804800b
1 pattern found.
pwndbg> find /b 0x8048000, 0x8048013, 0x6c,0x65
0x8048010
1 pattern found.
pwndbg> find /h 0x8048000, 0x8048013, 0x656c
0x8048010
1 pattern found.
pwndbg> find /w 0x8048000, 0x8048013, 0x0000656c
0x8048010
1 pattern found.
pwndbg> find /g 0x8048000, 0x8048013, 0x706d617865206e61
0x8048008
1 pattern found.
pwndbg> find 0x8048000, 0x8048013, "exampl"
Pattern not found.
pwndbg> find 0x8048000, 0x8048013, "example"
0x804800b
1 pattern found.
```
Strings are '0x00' terminated

LINKS
Pwndbg Docs: https://browserpwndbg.readthedocs.io/en/docs/
GDB Reference: https://sourceware.org/gdb/current/onlinedocs/gdb/index.html#Top

Generic commands

Disassembly style
- Show syntax — show disassembly-flavor
- Intel syntax — set disassembly-flavor intel
- AT&T syntax — set disassembly-flavor att

Misc
- Distance between mem addresses — distance [mem_addr_1] [mem_addr_2]
- Hexdump — hexdump [mem_addr] [num_of_bytes]

Information displaying
- Variables — info variables
- line numb -> addr and vice versa — info line *[memaddr]
info line [function_name]
- Functions — info functions
info functions [regex_func_name]

Breakpoints

Show set Breakpoints
- info breakpoints
i b

On execution
- Software — bp [memaddr]
break *[memaddr]
break [function_name]
- Hardware — hbreak *[memaddr]
hbreak [function_name]
- SW One shot — tbreak *[memaddr]
tbreak [function_name]
- HW One shot — thbreak *[memaddr]
thbreak [function_name]
- Conditional — break *[memaddr] if [bp_condition]
break [function_name] if [bp_condition]
- Set condition to an existing bp — condition [breakpt_id] [bp_condition]
- Stops on function Matching regex — rbreak [regexp]

Manage bps
- Enable a bp — enable [breakpt_id]
- Delete a bp — delete [breakpt_id]
- Disable a bp — disable [breakpt_id]
- Delete a bp using mem_addr — clear *[memaddr]

Fork

Follow mode
- The original process is debugged after a fork — set follow-fork-mode parent (gdb default)
- The new process is debugged after a fork — set follow-fork-mode child
- Current debugger response to a [v]fork — show follow-fork-mode

Exec
- set follow-exec-mode same — GDB keeps the process bound to the same inferior. The new executable image replaces the previous executable loaded in the inferior. Restarting the inferior after the exec call, with e.g., the run command, restarts the executable the process was running after the exec call. Link: https://sourceware.org/gdb/onlinedocs/gdb/Forks.html#Forks
- set follow-exec-mode new — GDB keeps the process bound to the same inferior. The new executable image replaces the previous executable loaded in the inferior. Restarting the inferior after the exec call, with e.g., the run command, restarts the executable the process was running after the exec call.
- Shows current debugger setting — show follow-exec-mode

Detach
- set detach-on-fork on — The child/parent proc (depending on the value of follow-fork-mode) will be detached and allowed to run independently.
- set detach-on-fork off — Both processes will be held under the control of GDB. One process (child or parent, depending on the value of follow-fork-mode) is debugged as usual, while the other is held suspended. Link: https://sourceware.org/gdb/onlinedocs/gdb/Forks.html#Forks.
- Shows current debugger setting — show detach-on-fork

Inferiors (childs)
- Manage inferiors — info inferiors
inferior [inferior_num]
detach inferior [inferior_num]
kill inferiors [inferior_num1]..[inferior_numN]

Catchpoints

Syscalls
- Stop execution on any syscall — catch syscall
- Stop execution on a specific syscall — catch syscall [syscall_name]
- Stop execution on a specific syscall — catch syscall [syscall_number]

Signals
- Catch a specific signal — catch signal [signal_name]
- Catch all signals — catch signal all
- Catch all signals, except SIGTRAP and SIGINT — catch signal

Fork
- Catch calls to fork — catch fork
- Catch calls to vfork — catch vfork

Exec
- Catch calls to Exec — catch exec

Libraries
- Load library — catch load [regexp]
- Unload library — catch unload [regexp]

Flow/Data Manipulation

Memory / Registers Values
- Change register value — set ${reg}=[reg_value]
- Change memory content — set *[mem_addr]=[mem_value]

Program Environment
- Show/Set/Unset env variables — show environment
set environment varname [=value]
unset environment varname
- Show Exec. Paths / Add a path — show paths
path [directory]

Stepping
- Step Into — stepi
si
- Step Over — stepover
so
nexti
ni
- Continue execution until return — finish

Navigation
- Start the prog and breaks on 'main', or entry point — start
- Start the prog and breaks on the 1st instr. — entry
- Breaks at the next jump instruction — nextjmp
- Breaks at the next call instruction — nextcall
- Breaks at the next ret instruction — nextret

Process
- Load program — pwndbg /path/to/program
- Load program with arguments — pwndbg --args ./prog [arg1] [arg2]..[argN]
- Run a loaded program — run
r
- Run a loaded program with args — run [arg1] [arg2]..[argN]
r [arg1] [arg2]..[argN]
- Run a loaded program passing inputs — run <<< `python -c 'print "AAA\n" '`
r <<< `python -c 'print "AAA\n" '`
- Divert output on a file — run > [output_file]
r > [output_file]
- View/Set arguments — argc
argv
set args [arg1] [arg2]..[argN]
- Attach to process / detach for curr. proc. — attach [process_id]
detach
- Kill debugged process — kill
- Continue execution — continue
c

Author: bdev
Website: https://reversingforfun.info