gdb + pwndbg Cheatsheet

DEFCON Toronto Introduction to Linux 64-bit Binary Exploitation

By superkojiman

Disassembling

Disassemble a function	`disassemble vuln`
Disassemble at address	`disassemble 0x400566`

Running

Run until termination or breakpoint	`r`
Run and pause at main()	`start`
Run and provide arguments	`r arg1 arg2`
If binary prompts for input once through stdin, pass input via a file	`r < in.txt`
If binary prompts for input more than once through stdin	`r < <(echo "input1"; echo "input2")`

Stepping

Continue execution	`c`
Execute next instruction and step over a function	`ni`
Execute instruction and step into a function	`si`

Breakpoints

Set breakpoint at function	`bp vuln`
Set breakpoint at address	`bp 0x4005b5`
Set breakpoint at function + offset	`bp vuln+47`
List breakpoints	`bl`
Delete all breakpoints	`d br`
Disable breakpoint 2	`bd 2`
Enable breakpoint 2	`be 2`

Examining data

Examine two 8-byte values at RBP in hex	`x/2gx $rbp`
Examine 10 instructions at main+25	`x/10i *main+25`
Examine 4 bytes of RAX in hex	`x/wx $rax`
Print R10 in decimal	`p/d $r10`
Print sum of 0x500 and 0x39 in decimal	`p/d 0x500+0x39`
Print the address of vuln()	`p vuln`

Use the **x** or **p** command followed by the size of the data and the format letters.

Sizes include **b**yte, **w**ord, **h**alfword, and **g**iant.

Format letters include **o**ctal, he**x**, **d**ecimal, **i**nstruction, **c**har, and **s**tring.

Modifying data

Set RAX to 5	`set $rax = 5`
Set the value pointed to by an address to 5	`set *0x7fffffffe280 = 5`
Set the value pointed to by RAX-8 to 5	`set *($rax-8) = 5`
Set the RIP register to another address	`set $rip = 0x4005b5`

FLAGS register

View FLAGS register	`regs eflags`
Set ZF flag (bit 6)	`set $eflags |= (1 << 6)`
Clear ZF flag (bit 6)	`set $eflags &= ~(1 << 6)`

```
Carry: CF = 0
Parity: PF = 2
Adjust: AF = 4
Zero: ZF = 6
Sign: SF = 7
Interruption: IF = 9
Direction: DF = 10
Overflow: OF = 11
```

Display state of the program

Show state	`context`

Get address of saved return pointer

Return address of current stack frame	`x/gx $rbp+8`
Discovered return addresses on the stack	`retaddr`

Search for a string in memory

Look for "Hello"	search Hello

Get distance between addresses

Using p	`p/d 0x4005b5-0x400566`
Using distance	`distance 0x400566 0x4005b5`

Print hexdump

Dump register	`hexdump $rsp`
Dump memory address	`hexdump 0x7fffffffe248`

Display stack

View the stack	stack
View 30 rows of the stack	stack 30

Print virtual memory map pages

Display everything	`vmmap`
Display stack	`vmmap stack`
Display program	`vmmap vuln01`

Check security settings

Show binary's security	`checksec`

References

Pwndbg: https://github.com/pwndbg/pwndbg

GDB Documentation: https://sourceware.org/gdb/current/onlinedocs/gdb/