radare2 Reverse engineering toolkit

START HERE

`r2 -A <file>`
Open and analyse

`iz`
Strings in data

`afl`
List functions

`axt <addr>`
Who references this?

`s main`
Seek to main

`VV`
Graph view

`pdf`
Disassemble function

`<cmd>?`
Help for any command

Open r2 file `aaa` → **Analyse** functions, xrefs `s` / `pdf` → **Inspect** seek + print `w` / `ood` → **Patch or debug** write, step

CONCEPTS

Commands are short letter chains: append ? to any prefix (a?, p?, pd?) to list what it offers.

Almost everything acts at the current seek; use cmd @ addr to run it elsewhere without moving.

Writing needs write mode: open with r2 -w, or reopen with oo+ . Without it, w commands fail.

OPEN AND ANALYSE

`r2 -A <file>`
Open and run full analysis

`r2 -d <file>`
Open under the debugger

`r2 -q -c '<cmds>' <file>`
Run commands and quit
Quote the command string; separate commands with ;. Use -qq to quit even after -i scripts.

`r2 -a <arch> -b <bits> <file>`
Force architecture and bits

`aa`
Basic analysis

`aaa`
Deeper analysis of calls and refs

`afl`
List functions

`afi`
Info about function here

`afn <name> [addr]`
Rename a function

`axt <addr>`
Find references to an address

`e <key>=<value>`
Set a config variable

BINARY INFO

`iI`
Binary info and protections

`iz`
Strings in data sections

`ii`
Imported functions

`ie`
Entry points

`iS`
Sections

`is`
Symbols

BINARY INFO (cont.)

`il`
Linked libraries

`rabin2 -I <file>`
Binary info without opening r2

`rabin2 -z <file>`
Strings from data sections

SEEK AND PRINT

`s <addr>`
Seek to address, flag or function

`s- / s+`
Undo / redo seek

`pdf`
Disassemble current function

`pd <n>`
Disassemble n instructions

`px [len]`
Hexdump

`ps`
Print zero-terminated string
psw wide, psp Pascal, psb all strings in block.

`p8 <len>`
Bytes as hex pairs

`pdc`
Pseudo-decompile function
pdg needs the r2ghidra plugin for a real decompiler.

`<cmd> @ <addr>`
Run cmd at addr without seeking

`<cmd>~<text>`
Grep the output

`? <expr>`
Evaluate a number expression

`f`
List flags

`CC <text>`
Add comment here

VISUAL MODE

`V`
Enter visual mode

`VV`
Graph view of function

`p / P`
Cycle print modes

`h j k l`
Move

`Enter`
Follow jump or call

`u / U`
Seek back / forward

VISUAL MODE (cont.)

`g`
Seek to offset or flag

`c`
Toggle cursor

`;`
Add comment

`x / X`
Xrefs to / from here

`:`
Run a command

`_`
Open flag and function HUD

`s / S`
Step into / over (debug)

`q`
Back to shell

SEARCH

`/ <text>`
Search for a string

`/i <text>`
Search ignoring case

`/x <hex>`
Search hex bytes
Dots are wildcard nibbles (/x 80..80); / x 8080:ff80 applies a mask.

`/c <asm>`
Search assembly text

`/R [filter]`
Search ROP gadgets

`//`
Repeat last search

WRITE AND PATCH

`wx <hex>`
Write hex bytes

`wa <asm>`
Assemble and write
Separate instructions with ;.

`wao nop`
Patch opcode to nop

`w <text>`
Write string

`oo+`
Reopen in write mode

DEBUGGER

`ood [args]`
Reopen file in debug mode

`dc`
Start or continue

`dcu <addr>`
Continue until address

DEBUGGER (cont.)

`db <addr>`
Set breakpoint

`db -<addr>`
Remove breakpoint

`db`
List breakpoints

`ds [n]`
Step into

`dso`
Step over

`dcr`
Continue until return

`dcs`
Continue until syscall

`dr`
Show registers

`dr <reg>=<val>`
Set register

`drr`
Registers with references

`dbt`
Backtrace

`dm`
Memory maps

`r2 -d <pid>`
Attach to a running process

COMPANION TOOLS

`rasm2 -a x86 -b 64 '<asm>'`
Assemble to hex

`rasm2 -a x86 -b 64 -d <hex>`
Disassemble hex

`radiff2 -C <a> <b>`
Diff code by function

`rax2 <expr>`
Calculate and convert bases

RECIPES

First look at an unknown binary

```
rabin2 -I ./a.out
r2 -A ./a.out
iz~http
afl~main
s main; pdf
```

Find who uses a string

```
iz~password
# note the vaddr, say 0x00402010
axt 0x00402010
s <caller>; pdf
```

Break on main and inspect

```
r2 -d ./a.out
db main
dc
dr
pxr 64 @ rsp
dso
```