

And other valuable commands you will use often.

# THIS IS THE PART WHEN WE PUT THE FLAG ON THE BOARD

This is here so we don't forget

A large, dark blue, diagonal shape that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

# This is not a tutorial on how to get into terminal.

If you are still not sure, thats okay! Go to [sigpwny.com/tutorial](https://sigpwny.com/tutorial) and get set up :)

# find

Find files! Most often by filename

```
find [directory] -name 'spaghetti'
```

```
find [directory] -name '*.ext'
```

# grep

Find text within files!

```
grep -r "text you want to find" .
```

```
grep -A -B -C -r "text you want to find"
```

```
cat | grep
```

# strings

see what strings exist in a file

really good initial command for RE/PWN

## **Why does this command exist???**

Some files are not human readable, this prints out all the human readable things

There may or may not be a useful challenge for this one.

# file

Determine the file type of a file (what it really is)

Valuable to know what you are looking at before you start attacking/RE'ing something

# `gdb`

Look at executables and slowly step through them.

We can run an entire meeting on `gdb`, and we will.

`gdb --args` “is a way to have command line arguments”

`b` = breakpoint

`n` = next

`s` = step

`si` = step instruction (use this one for binaries)

`x` = look at the stack

`print` = print variables



# Get onto the pwny server!

<https://github.com/sigpwny/sigpwny.github.io/wiki/How-to-get-on-the-SIGPwny-server>

# What is r2?

- R2 is a “great” free disassembler
  - It is also known as “Radare2”
  - Easy to pick up the basics for
  - Clunky and weird to use, but can be used within your personal terminal
- You can install on your linux environment or ssh into sigpwny server
- A good starter disassembler

# git

Git is difficult, we could do meetings on meetings on meetings. For this meeting, know the following.

`git clone [url] [folder]`

Clones a repository from a url

`git add -A`

**stages** all unadded files to the **repository**

`git commit -m "Commit message"`

**Commits** those stages to your personal **branch**

`git push`

Pushes your branch to the main branch

`git pull`

Pulls the latest changes from the main branch

# Basics

```
2. r2 how2re (radare2)
[0x004005e0 23% 1120 how2re]> xc @ entry0
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF comment
0x004005e0 31ed 4989 d15e 4889 e248 83e4 f050 5449 1.I.^H..H...PTI ; [14] -r
0x004005f0 c7c0 500a 4000 48c7 c1e0 0940 0048 c7c7 ..P.@.H...@.H..
0x00400600 3507 4000 e8a7 ffff fff4 660f 1f44 0000 5.@.....f..D..
0x00400610 b867 1060 0055 482d 6010 6000 4883 f80e .g.`UH-`.`.H...
0x00400620 4889 e576 1bb8 0000 0000 4885 c074 115d H..v.....H..t.]
0x00400630 bf60 1060 00ff e066 0f1f 8400 0000 0000 .`.`.f.....
0x00400640 5dc3 0f1f 4000 662e 0f1f 8400 0000 0000 ]...@.f.....
0x00400650 be60 1060 0055 4881 ee60 1060 0048 c1fe .`.`.UH..`.`.H..
0x00400660 0348 89e5 4889 f048 c1e8 3f48 01c6 48d1 .H..H..H..?H..H.
0x00400670 fe74 15b8 0000 0000 4885 c074 0b5d bf60 .t.....H..t.].`
0x00400680 1060 00ff e00f 1f00 5dc3 660f 1f44 0000 .`.`.f.....f..D..
0x00400690 803d c909 2000 0075 1155 4889 e5e8 6eff .=. .u.UH...n.
0x004006a0 ffff 5dc6 05b6 0920 0001 f3c3 0f1f 4000 .]. . . . .@.
0x004006b0 bf20 0e60 0048 833f 0075 05eb 930f 1f00 .`.`.H.?..u.....
0x004006c0 b800 0000 0048 85c0 74f1 5548 89e5 ffd0 . . . . .H..t.UH...
0x004006d0 5de9 7aff ffff 5548 89e5 4883 ec20 4889 ].z...UH..H.. H.
0x004006e0 7de8 89f0 8955 e088 45e4 0fbe 55e4 488b }. . . . .U..E...U.H.
0x004006f0 45e8 89d6 4889 c7e8 84fe ffff 4889 45f8 E...H.. . . . .H.E.
```

Open terminal

`r2 fileName`

Press `v` and then enter to activate visual mode

That looks pretty complicated... but we can make it look a lot easier to understand.

# Print mode

Press p to toggle print mode (easier to see instructions)

You can navigate with the arrow keys, but that is slow.

```
2. r2 how2re (radare2)
[0x004005e0 23% 100 how2re]> pd $r @ entry0
;-- entry0:
;-- section..text:
;-- rip:
0x004005e0 31ed      xor ebp, ebp      ; [14] -r
0x004005e2 4989d1    mov r9, rdx
0x004005e5 5e        pop rsi
0x004005e6 4889e2    mov rdx, rsp
0x004005e9 4883e4f0  and rsp, 0xfffffffffffffff0
0x004005ed 50        push rax
0x004005ee 54        push rsp
0x004005ef 49c7c050a40. mov r8, 0x400a50
0x004005f6 48c7c1e00940. mov rcx, 0x4009e0
0x004005fd 48c7c7350740. mov rdi, 0x400735 ; main
0x00400604 e8a7ffff    call sym.imp.__libc_start_main ;[1]
0x00400609 f4        hlt
0x0040060a 660f1f440000 nop word [rax + rax]
0x00400610 b867106000 mov eax, 0x601067
0x00400615 55        push rbp
0x00400616 482d60106000 sub rax, 0x601060
```

# Moving faster

```
2. r2 how2re (radare2)
[0x00400735 29% 875 how2re]-> pd $r @ main

;-- main:
0x00400735 55          push rbp
0x00400736 4889e5      mov rbp, rsp
0x00400739 4883ec60    sub rsp, 0x60
0x0040073d 897dac      mov dword [rbp - 0x54], edi
0x00400740 488975a0    mov qword [rbp - 0x60], rsi
0x00400744 64488b042528. mov rax, qword fs:[0x28] ; [0x28:8
0x0040074d 488945f8    mov qword [rbp - 8], rax
0x00400751 31c0       xor eax, eax
0x00400753 837dac48    cmp dword [rbp - 0x54], 0x48 ; [0x
0x00400757 750f       jne 0x400768 ;[1]
0x00400759 bff800a4000 mov edi, str.Each_character_in_the_fl
0x0040075e b8000000000 mov eax, 0
0x00400763 e828feffff call sym.imp.printf ;[2]
0x00400768 488d45b0    lea rax, [rbp - 0x50]
0x0040076c 4889c6      mov rsi, rax
0x0040076f bfa90a4000 mov edi, 0x400aa9
0x00400774 b8000000000 mov eax, 0
0x00400779 e842feffff call sym.imp.__isoc99_scanf ;[3]
```

Press **n** and **N** to navigate between sections

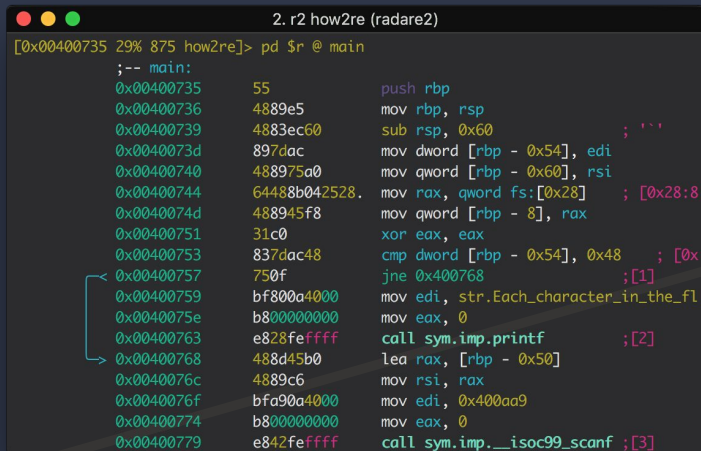
You usually want to look for the **main** function, as that is where things are going on.

# Making it even easier to read.

Press d, and then f.

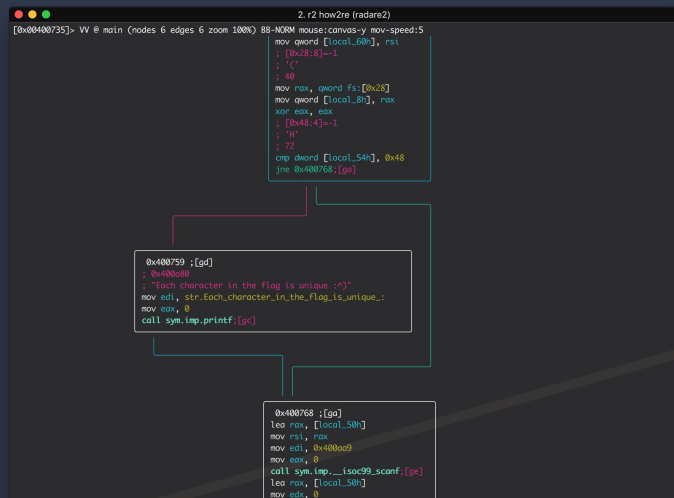
**d** means define, and **f** is function.

This **defines** main as a **function**, and allows us to look at it in a much easier to understand way (visual representation)



```
2. r2 how2re (radare2)
[0x00400735 29% 875 how2re]> pd $r @ main
;-- main:
0x00400735 55      push rbp
0x00400736 4889e5  mov rbp, rsp
0x00400739 4883ec60 sub rsp, 0x60 ; ''
0x0040073d 897dac  mov dword [rbp - 0x54], edi
0x00400740 488975a0 mov qword [rbp - 0x60], rsi
0x00400744 64488b042528. mov rax, qword fs:[0x28] ; [0x28:8
0x0040074d 488945f8 mov qword [rbp - 8], rax
0x00400751 31c0    xor eax, eax
0x00400753 837dac48 cmp dword [rbp - 0x54], 0x48 ; [0x
0x00400757 750f    jne 0x400768 ;[1]
0x00400759 bf800a4000 mov edi, str.Each_character_in_the_fl
0x0040075e b800000000 mov eax, 0
0x00400763 e828feffff call sym.imp.printf ;[2]
0x00400768 488d45b0 lea rax, [rbp - 0x50]
0x0040076c 4889c6  mov rsi, rax
0x0040076f bfa90a4000 mov edi, 0x400aa9
0x00400774 b800000000 mov eax, 0
0x00400779 e842feffff call sym.imp.__isoc99_scanf ;[3]
```

# Graphical Representation



Press V to enter visual mode, this allows you to see what is actually going on in the script in a nice visual way.

You can see where jumps go, true or false

Helpful Radare2 Book (From actual website)

[https://radare.gitbooks.io/radare2book/content/first\\_steps/intro.html](https://radare.gitbooks.io/radare2book/content/first_steps/intro.html)



# tmux



A really jank way to keep processes running after you close the terminal window

(So basically the thing everyone uses)

So if you want to... keep a ctf up, run a file sharing system, or run a Minecraft Server without needing a terminal window open.

# How to install tmux

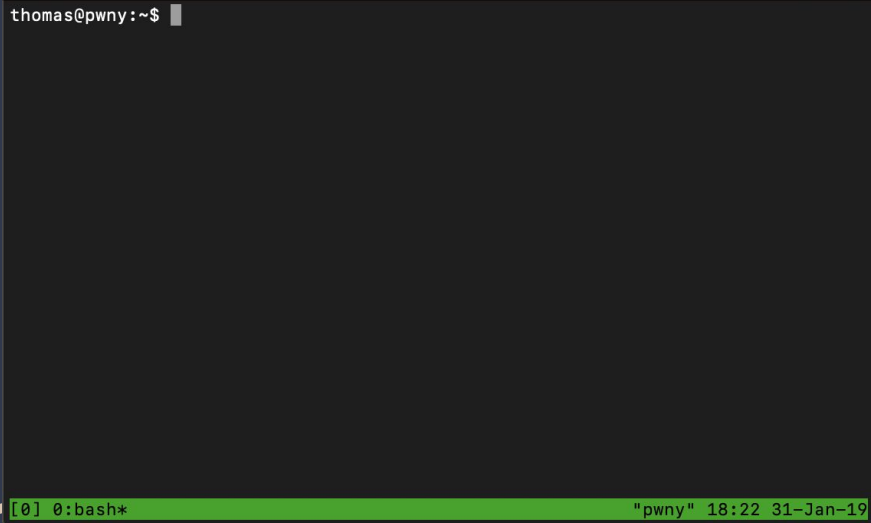
It would be a good learning experience if you figure out how to install bash extensions on your operating system. So go try to do that.

It may also already be installed, the command is...

**tmux**



# tmux basics



```
thomas@pwny:~$
```

```
[0] 0: bash*
```

```
"pwny" 18:22 31-Jan-19
```

To create a new window type **tmux**

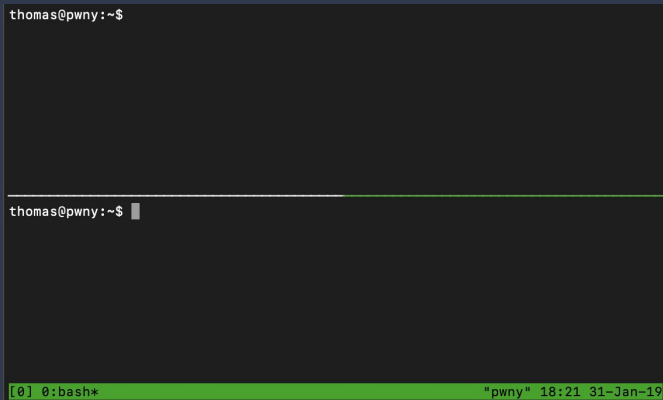
This will open up a new tmux window for you to use.

Mess around with it see what you can do, it is a fully functioning bash window.

**NEVER** nest multiple tmux windows

Just a bad idea ;P

# tmux commands

A screenshot of a tmux terminal window. The window is split horizontally into two panes. The top pane shows a prompt 'thomas@pwny:~\$' on a black background. The bottom pane also shows a prompt 'thomas@pwny:~\$' on a black background, with a cursor visible. At the bottom of the window, a green status bar displays '[0] 0: bash\*' on the left and '"pwny" 18:21 31-Jan-19' on the right.

```
thomas@pwny:~$  
  
thomas@pwny:~$ █  
  
[0] 0: bash* "pwny" 18:21 31-Jan-19
```

The command prefix is `cntrl + b + ____`, some things you can do with this are

- `d` : detach your terminal from the tmux window
  - This will keep anything you had running, still running.
  - You can get back to that session with **`tmux at -t (number_of_session)`**
- `"` : horizontal split of terminal
- `%` : vertical split of terminal
- Force close split
  - `Ctrl-d`, **`exit`**

# Detaching windows

- Cntrl+b + d to detach a window
  - Will run REALLY long
- Reattach
  - `tmux attach -t NUMBER`
  - `tmux ls`
  - `tmux new -s NAME`
  - `tmux rename-session -t NUMBER NAME`

## Easy tutorial

<https://www.hamvocke.com/blog/a-quick-and-easy-guide-to-tmux/>

# Vim



- Vim is an in terminal text editor
- It is NOT an IDE
- **vim** to open new vim window

EXIT VIM with :wq (colon + wq)

i = Insert Mode

dd = delete line

p = paste deleted line

:tabnew FILE\_NAME = open new tab, gt to navigate tabs

:LINE\_NUMBER = jump to line number

**Use .vimrc!!!**