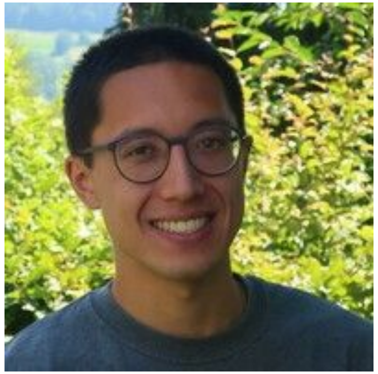
Software Security

63 73 - 34 31 32

The teaching assistants

Florian Hofhammer

Solène Husseini


Philipp Mao

Han Zheng

Introduction to Software Development in 2025

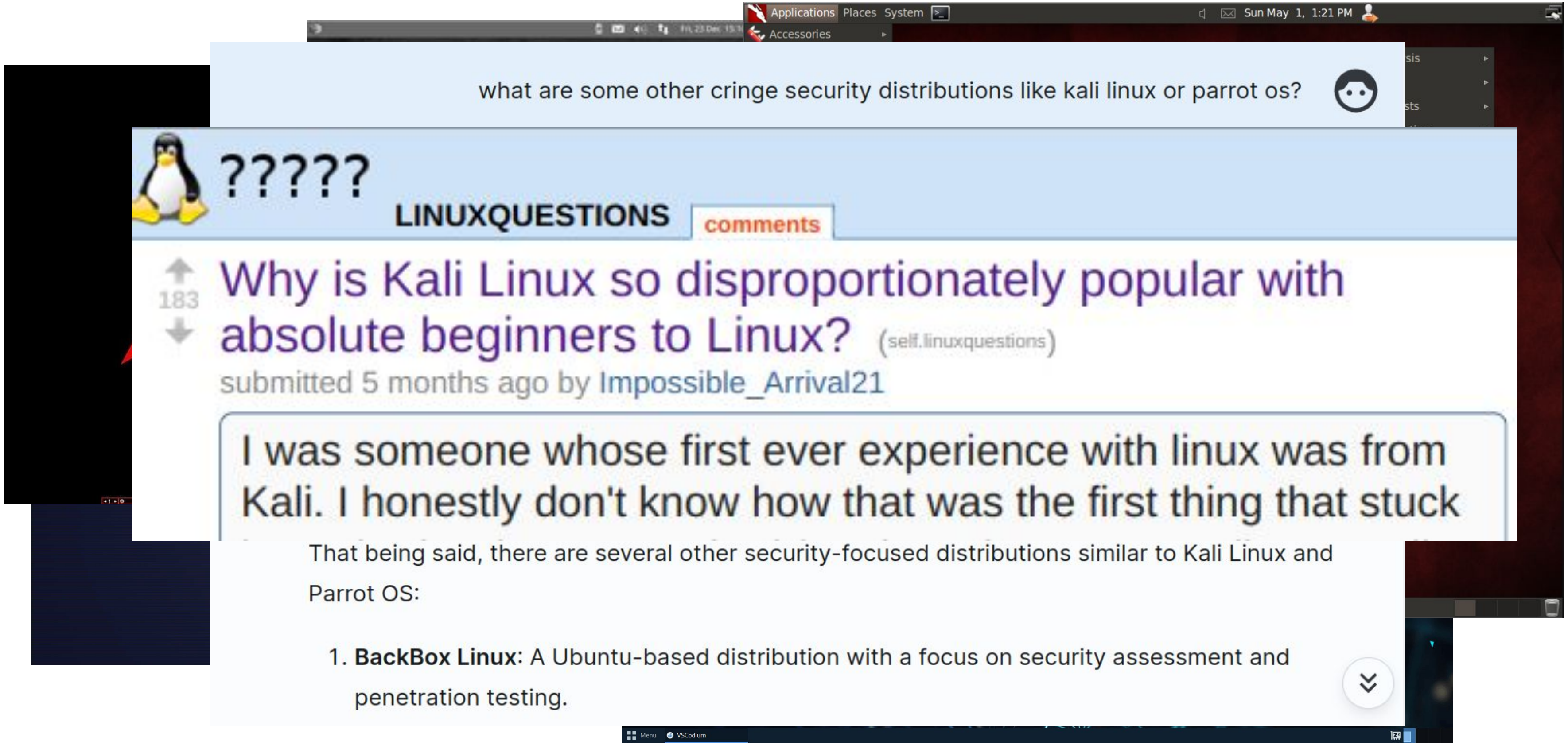
CS412

Software development in 2025

Uses JS/Python/Scala	Uses C (and rarely assembly)
Writes e.g. web applications, data analytics	Writes system software (e.g., OS, servers)
Has 90% of the code generated by an LLM	Doesn't use LLMs because they don't integrate well with `vi`
Abstractions, slow, difficult to map to machine	Pedal to the metal, full speed no regrets
Unstable new tools, fancy new frameworks, move fast break things	Ancient toolchains written into stone by the gods of computer science
Cringe	Based

(this will be you when you get a job) (this will be you after this class)

"Hacker" tools dangerous due to death from cringe



what are some other cringe security distributions like kali linux or parrot os?

????? LINUXQUESTIONS comments

183

Why is Kali Linux so disproportionately popular with absolute beginners to Linux? (self.linuxquestions)

submitted 5 months ago by Impossible_Arrival21

I was someone whose first ever experience with linux was from Kali. I honestly don't know how that was the first thing that stuck

That being said, there are several other security-focused distributions similar to Kali Linux and Parrot OS:

1. **BackBox Linux**: A Ubuntu-based distribution with a focus on security assessment and penetration testing.

This is not a course about pentesting!

This is a course about *vulnerability research.*

Tools that are useless for vuln research

- Anything that contains the phrase "ethical", "whitehat", and so on
- Anything that requires a GUI / mouse to use
- Anything that you find on "hacking forums"
- Anything that has good UX, a cool logo, or any fancy graphics

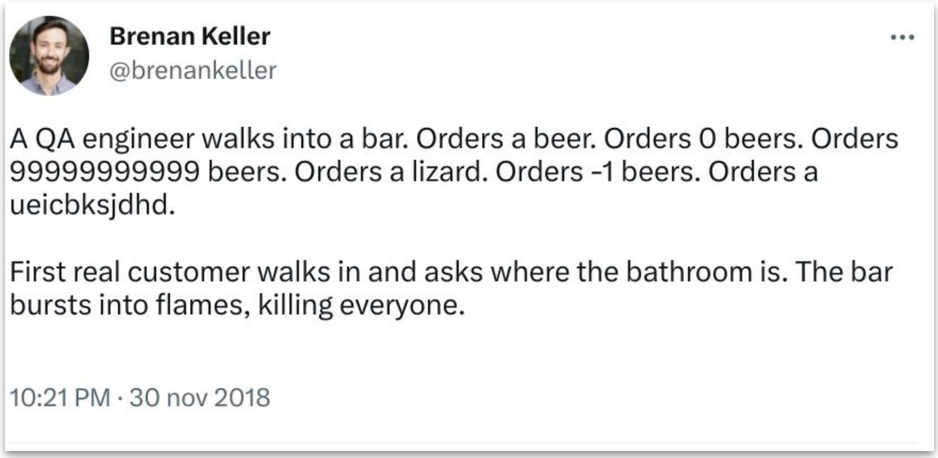
Instead, good security tools are:

- Unstable, buggy, constantly crash
- Made by a crazy schizophrenic (or other forms of neurodivergence) single dev with godlike coding skills (or, as we will see, by a three letter american agency)
- Vomit-looking, a punch to the eyes, made for machines, not humans

What is vulnerability research about

1. How does it *actually* work?

2. Can I mess with it?



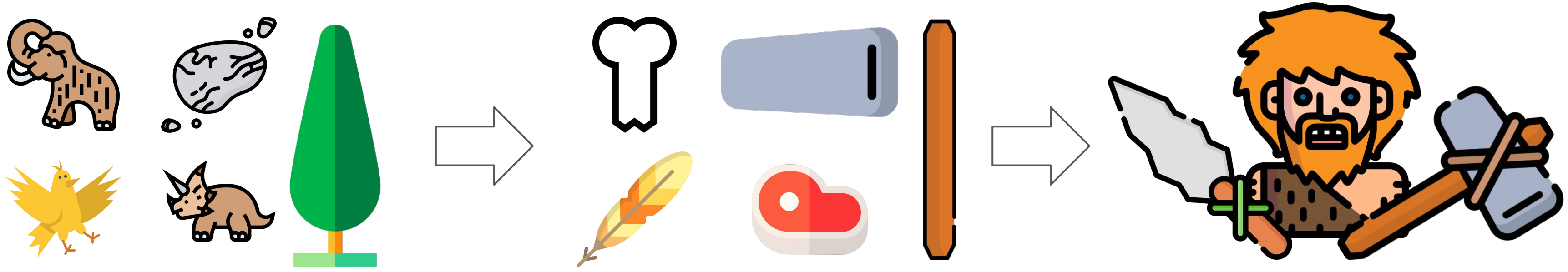
Brenan Keller
@brenankeller

A QA engineer walks into a bar. Orders a beer. Orders 0 beers. Orders 99999999999 beers. Orders a lizard. Orders -1 beers. Orders a ueicbksjdhd.

First real customer walks in and asks where the bathroom is. The bar bursts into flames, killing everyone.

10:21 PM · 30 nov 2018

Ancient toolchains



Sources (.c, .h)

- *git*
- linters

git tracks changes.
Linters enforce style rules.

Object files

- *gcc/clang*
- *make*
- *ld*

gcc compiles source files.
make tracks "recipes" and recompiles on source updates.
ld links object files to create executables, static or shared libraries.

Binaries (a.out, so, a)

- *gdb*
- *objdump*

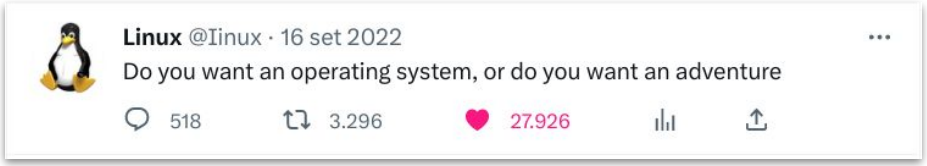
gdb allows runtime debugging.
objdump is used to inspect binaries.

Tool #0

Linux

Linux

We are friends with the penguin

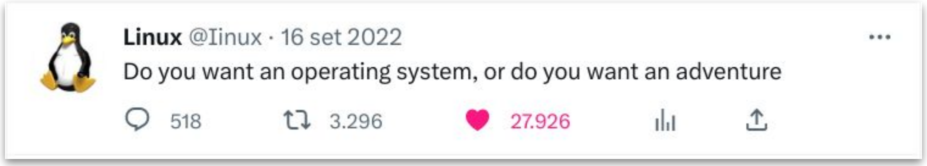

Linux @Iinux · 16 set 2022

Do you want an operating system, or do you want an adventure

Linux

We are friends with the penguin

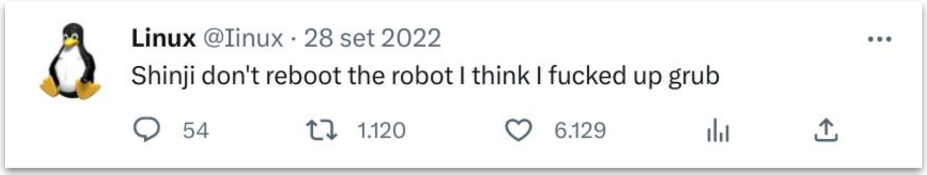
You need to be friends with the penguin



Linux @Iinux · 16 set 2022

Do you want an operating system, or do you want an adventure

518 3.296 27.926



Linux @Iinux · 28 set 2022

Shinji don't reboot the robot I think I fucked up grub

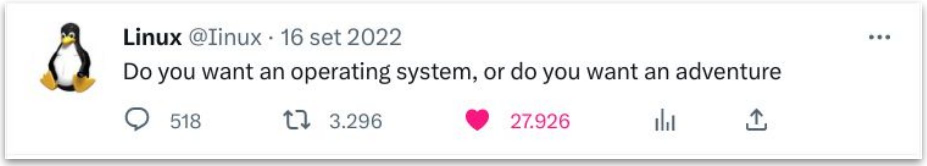
54 1.120 6.129

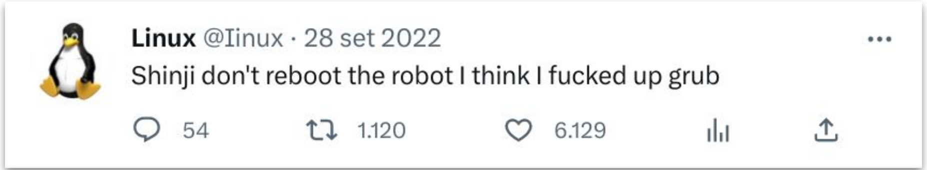
Linux

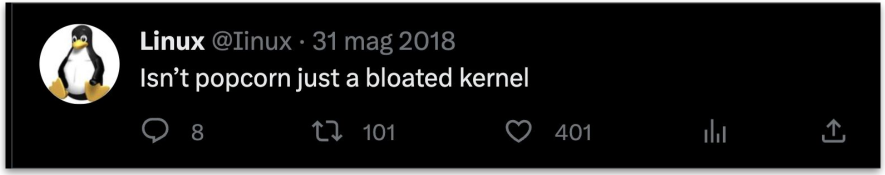
We are friends with the penguin

You need to be friends with the penguin

You're not friends with the penguin
⇒ you will not be friends with this class's labs


Linux @Iinux · 16 set 2022
Do you want an operating system, or do you want an adventure
518 3.296 27.926


Linux @Iinux · 28 set 2022
Shinji don't reboot the robot I think I fucked up grub
54 1.120 6.129


Linux @Iinux · 31 mag 2018
Isn't popcorn just a bloated kernel
8 101 401

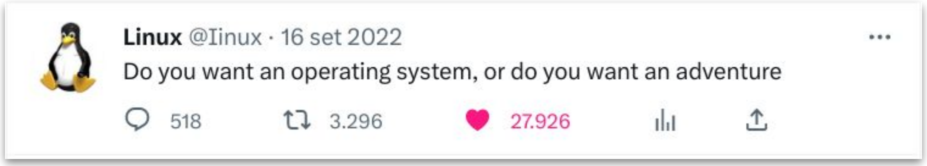
Linux

We are friends with the penguin

You need to be friends with the penguin

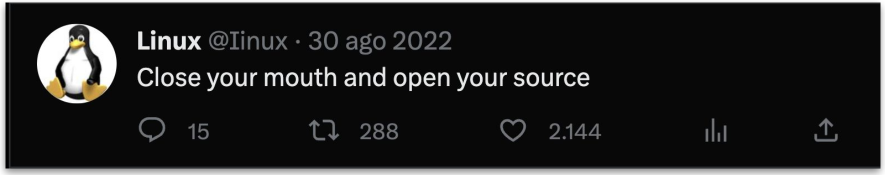
You're not friends with the penguin
$\Rightarrow$ *you will not be friends with this class's labs*

Install Linux
Use a VM
Use WSL
Use docker on Windows/macOS
Rent a Linux server


Linux @Iinux · 16 set 2022
Do you want an operating system, or do you want an adventure
518 3.296 27.926

Linux @Iinux · 28 set 2022
Shinji don't reboot the robot I think I fucked up grub
54 1.120 6.129

Linux @Iinux · 31 mag 2018
Isn't popcorn just a bloated kernel
8 101 401


Linux @Iinux · 30 ago 2022
Close your mouth and open your source
15 288 2.144

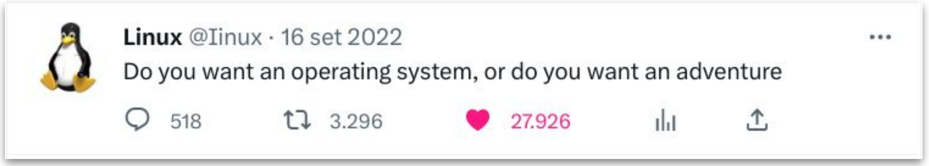
Linux

We are friends with the penguin

You need to be friends with the penguin

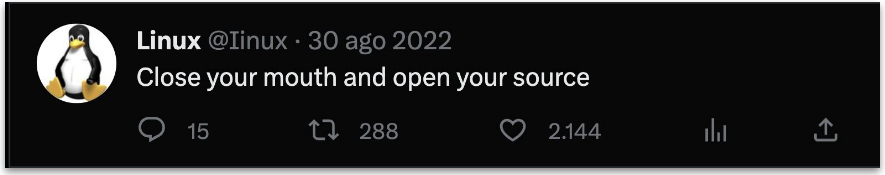
You're not friends with the penguin
⇒ you will not be friends with this class's labs

You cannot be "just friends":
you need to be comfortable spending time with the penguin.


Linux @Iinux · 16 set 2022
Do you want an operating system, or do you want an adventure
518 3.296 27.926

Linux @Iinux · 28 set 2022
Shinji don't reboot the robot I think I fucked up grub
54 1.120 6.129

Linux @Iinux · 31 mag 2018
Isn't popcorn just a bloated kernel
8 101 401


Linux @Iinux · 30 ago 2022
Close your mouth and open your source
15 288 2.144

x86? Arm? Wtf

If you have an Apple Silicon MacBook or another super scuffed Arm machine...

Lab 2 should run natively just fine

For x86 binaries in Lab 1: you will need to be able to run x86 Linux binaries!

x86? Arm? Wtf

Have a second x86 machine with Linux

Rent/get a free x86 VPS (e.g., free Azure credits, Oracle free tier)

qemu-user/Docker cross-arch

qemu-system

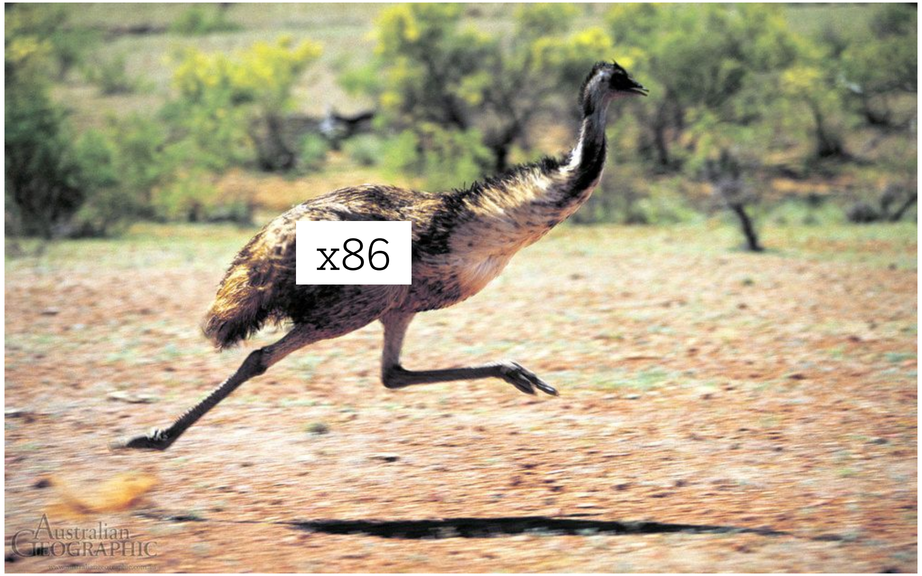
vdi.epfl.ch (no root!)

QEMU User

```
qemu-x86_64 <your binary>
```

Might need to set flags for libraries and stuff,

e.g., `qemu-x86_64 -L /usr/x86_64-linux-gnu <your binary>`



Cross-architecture Docker

First: if you can, use the Docker Engine, ***not*** Docker Desktop (applies to Linux, on macOS or Windows you don't have a chance (but you should use Linux anyways))!

Then, follow the documentation to enable cross-architecture support via QEMU: https://docs.docker.com/build/building/multi-platform/#qemu

Last, check whether you can run x86 Docker images: `docker run --rm -it --platform=linux/amd64 alpine:latest uname -m` should print `x86_64`

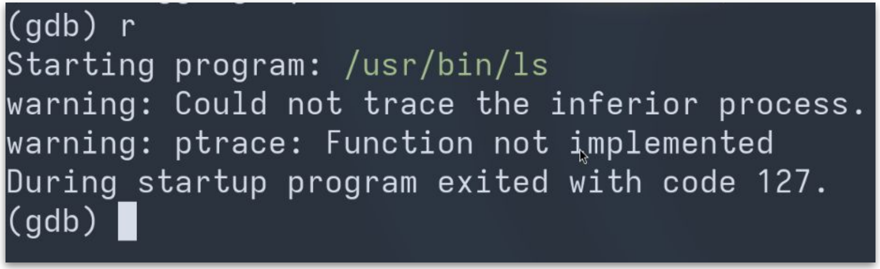
The platform flag is the important flag for cross-arch execution!

Under the hood: qemu-user!

Cross-architecture Docker

Advantage: no issue with libraries, library paths, etc.

Disadvantage:



```
(gdb) r
Starting program: /usr/bin/ls
warning: Could not trace the inferior process.
warning: ptrace: Function not implemented
During startup program exited with code 127.
(gdb)
```

Solution: `qemu-x86_64 -g 1234 <your binary>, gdb -ex “target remote :1234”`

Workarounds exist but are painful!

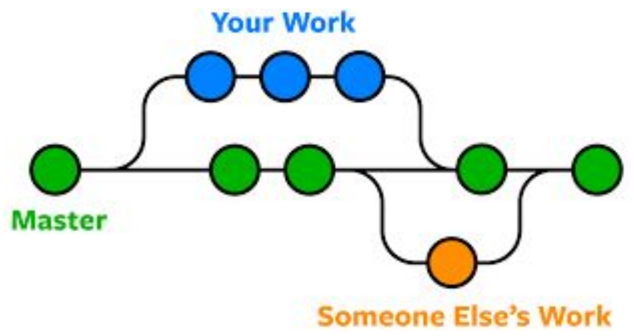
Tool #1

Git: the bathroom of programmers

Track changes to source files

Featuring:

- Branches, checkout
- Push, pull (--rebase), merge
- Diff, patch
- Blame
- .gitignore



Git: the bathroom of programmers

Track changes to source files

Featuring:

- Branches, checkout
- Push, pull (--rebase), merge
- Diff, patch
- Blame
- .gitignore

Not knowing how to cleanly use git in 2025 is like not knowing how to cleanly use the toilet

Git: the porcelain and the plumbing

git status
git commit
git pull
git stash
git checkout

git ls-tree
git update-index
git write-tree
git commit-tree
Stuff inside the .git folder

Git: the porcelain and the plumbing

git status
git commit
git pull
git stash
git checkout

You don't need to know how to do plumbing. It takes a long time and it's usually full of disgusting stuff.

BUT YOU DO NEED TO KNOW HOW TO USE THE TOILET

- You `git pull` without `--rebase`
- You get a merge conflict!

Git gud at git

Using all the features of git has a learning curve:

- Rebase your pulls
- Use aliases for your commands
- Use ssh auth
- Use branches and git diffs
- Use .git/hooks folder to run stuff on e.g. every commit
- Tweak your .gitconfig
- Integrate with tools: editor, plugins, shell prompt, etc
- Host your own git server

Git compared to other versioning systems

git vs mercurial

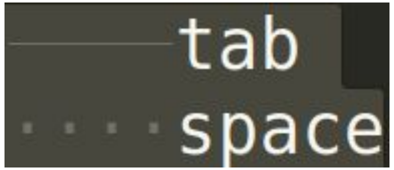
Tool #2

Linters: looking fine, everytime

Enforce rules on source file formatting: E.g. `clang-format`

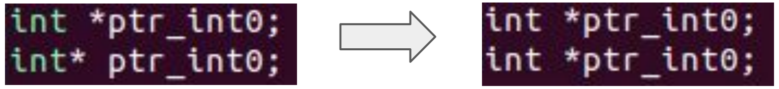
Examples of `clang-format` rules

- IndentWidth: 4



```
tab
space
```

- PointerAlignment: Right



```
int *ptr_int0;
int* ptr_int0;
```

```
int *ptr_int0;
int *ptr_int0;
```

Tool #3

gcc: parlez-vous machine code?

Compiles (translates) C/C++/Assembly code to machine code

Phases:

- Preprocessor: source code transformations
  - macros
  - include
  - #ifdef, #if, #elif, #else, #endif
- Compilation: Syntax checking, parsing, optimization, code generation
  - `-O1/O2/O3/Osize` for optimization
  - `-Wall -Werror` for catching mistakes
- Linking (ld): Merge different files

Tool #4

make: to recompile or not to recompile

Build system using rules and recipes

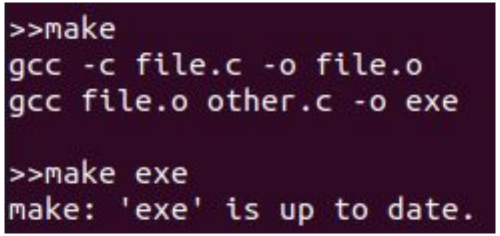
Rules define

- Targets (what to build)
- Dependencies (what is needed to build)
- Recipes (how to build)

Only compile required files

```
file.o: file.c file.h
	gcc -c file.c -o file.o

exe: file.o other.c
	gcc file.o other.c -o exe
```



```
>>make
gcc -c file.c -o file.o
gcc file.o other.c -o exe

>>make exe
make: 'exe' is up to date.
```

make: to recompile or not to recompile

I wish I could run this command on every compilation

I wish I could recompile only the files that changed

I wish I could compile multiple builds at the same time (e.g., an executable for every OS)

I wish I could provide a way to let other people easily compile my code

make: to recompile or not to recompile

I wish I could run this command on every compilation

I wish I could recompile only the files that changed

I wish I could compile multiple builds at the same time (e.g., an executable for every OS)

I wish I could provide a way to let other people easily compile my code

> Learn make. It takes you 20 mins. It saves you hours.

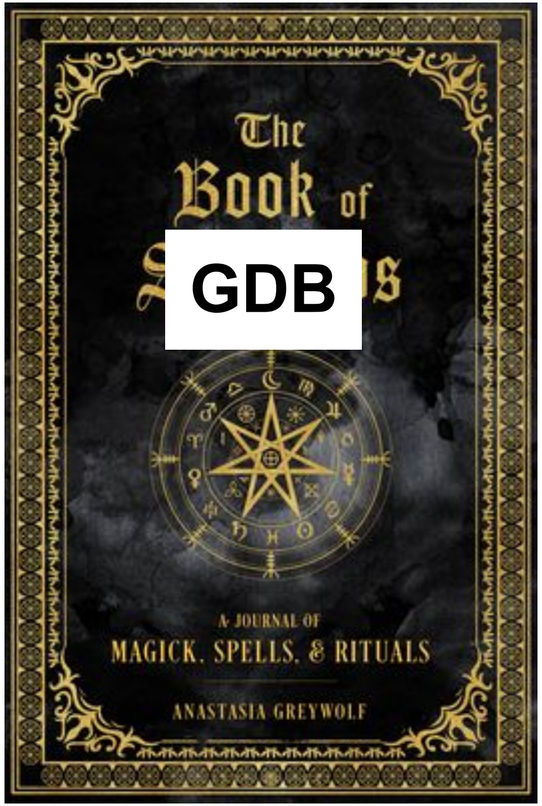
Tool #5

gdb: black magic at its finest

Allows debugging of running programs

Features:

- Breakpoints
- Watchpoints
- Inspection
- Modification



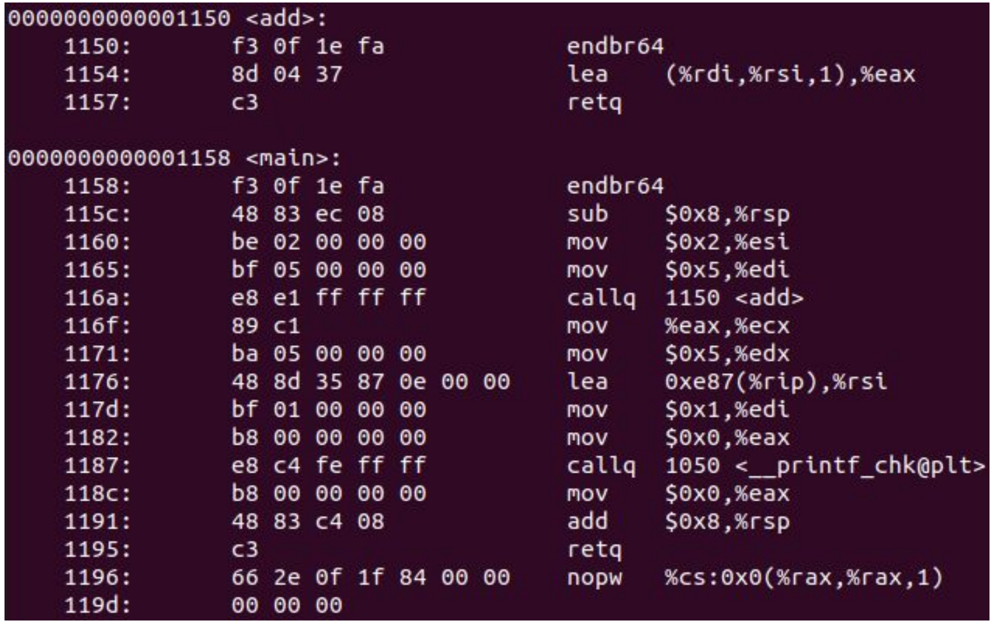
Tool #6

Some kind of disassembler/decompiler

- Objdump
- Radare2
- Ghidra
- IDA Free
- …

Up to personal preference.

If you're completely new to disassembly, we recommend Ghidra
⇒ hands-on introduction in a later lab



```
0000000000001150 <add>:
    1150:       f3 0f 1e fa             endbr64
    1154:       8d 04 37                lea    (%rdi,%rsi,1),%eax
    1157:       c3                      retq

0000000000001158 <main>:
    1158:       f3 0f 1e fa             endbr64
    115c:       48 83 ec 08             sub    $0x8,%rsp
    1160:       be 02 00 00 00          mov    $0x2,%esi
    1165:       bf 05 00 00 00          mov    $0x5,%edi
    116a:       e8 e1 ff ff ff          callq  1150 <add>
    116f:       89 c1                   mov    %eax,%ecx
    1171:       ba 05 00 00 00          mov    $0x5,%edx
    1176:       48 8d 35 87 0e 00 00    lea    0xe87(%rip),%rsi
    117d:       bf 01 00 00 00          mov    $0x1,%edi
    1182:       b8 00 00 00 00          mov    $0x0,%eax
    1187:       e8 c4 fe ff ff          callq  1050 <__printf_chk@plt>
    118c:       b8 00 00 00 00          mov    $0x0,%eax
    1191:       48 83 c4 08             add    $0x8,%rsp
    1195:       c3                      retq
    1196:       66 2e 0f 1f 84 00 00    nopw   %cs:0x0(%rax,%rax,1)
    119d:       00 00 00
```

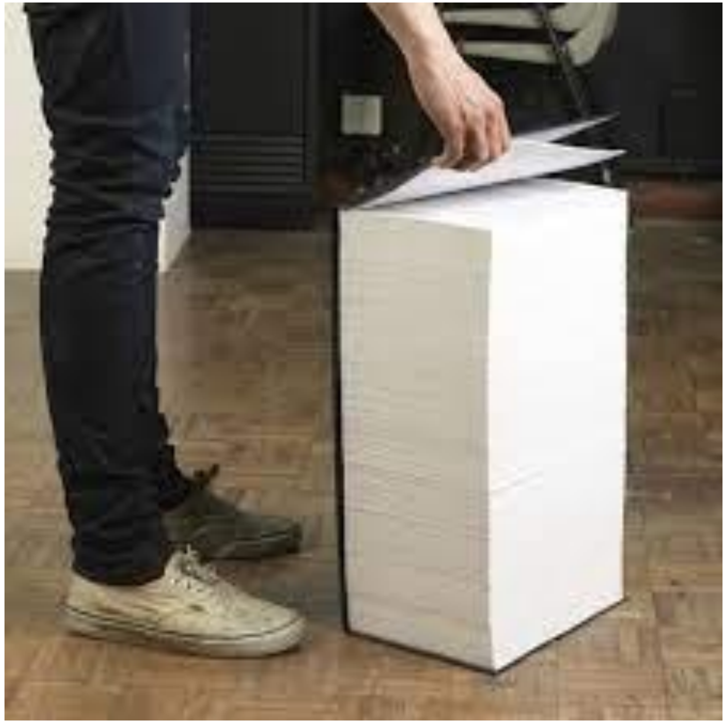
Tool #7

Manpages

Written by ancient gods, in unknown times

Read them like a religious text

Consult it on all kind of questions, self-doubts, and philosophical dilemmas



Manpages

Written by ancient gods, in unknown times

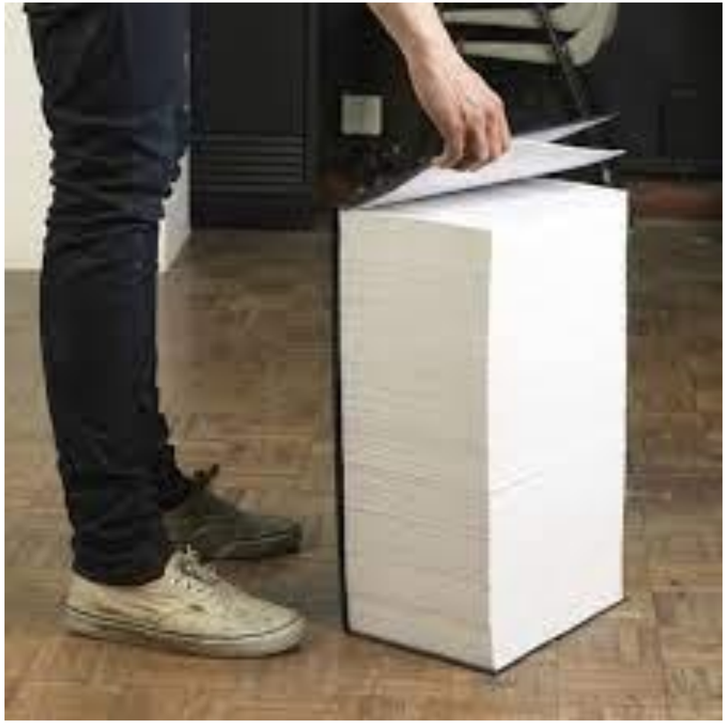
Read them like a religious text

Consult it on all kind of questions, self-doubts, and philosophical dilemmas

If you can, specify the section (man 1 printf vs `man 3 printf`)!

`man man` if you don't remember

`man gittutorial` is to this day the best tutorial for git ever written (also the rest of the git manpages)



Tool #8

Docker

Containerization

Quick testing of new tools

Clean environment to distribute your code into

A wrapper of linux namespaces (see bocker, implemented in bash)

Riddled with security pitfalls

~~Docker~~ Podman

Containerization

Quick testing of new tools

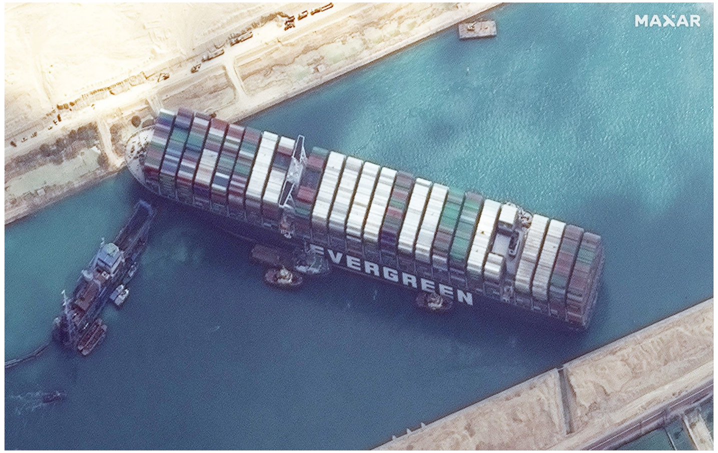
Clean environment to distribute your code into

A wrapper of linux namespaces (see bocker, implemented in bash)

Riddled with security pitfalls (but you can run it without root)

Same commands as docker

Made by RedHat



Graded Labs (40% of final grade)

CTF (find the vulns and pwn them)

- Competition + KOTH finals
- Start: February 27th
- End: March 20th

Fuzzing (find the vulns but automatically)

- Start: April 10th
- End: May 8th

Further Resources

"The Missing Semester" from MIT: https://missing.csail.mit.edu/