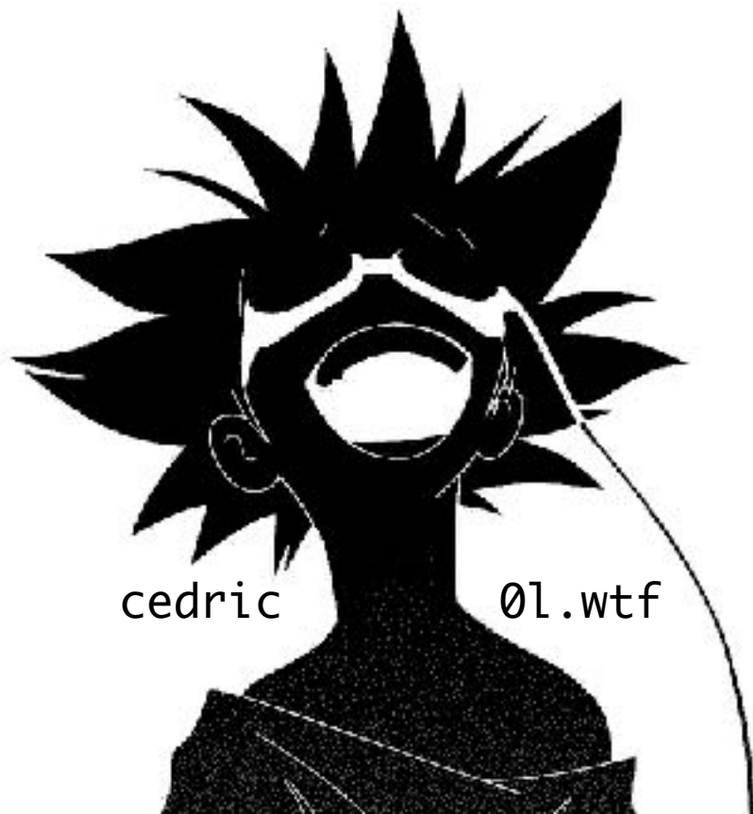
Anti-VM and Anti-Debugging Techniques

```
3 + 3 = exit(-1);
```



Anti-Debugging #1

code: ptrace.c

- Avoid ptrace'ing(2) via ggdb or lldb.

- Using direct ptrace(2) syscall to deal with `LDPRELOAD` attack if executable is statically linked.

- No posterior ptrace'ing.

Anti-Debugging #2

`code: prctl.c`

- Same approach as #1 (using direct syscall).
- prctl(2) (process control) for Linux and some BSD flavors.
- `PR_SET_DUMPABLE` to 0 to avoid core dump and avoid attach using ptrace(2).

Anti-Debugging #3

code: brk.c

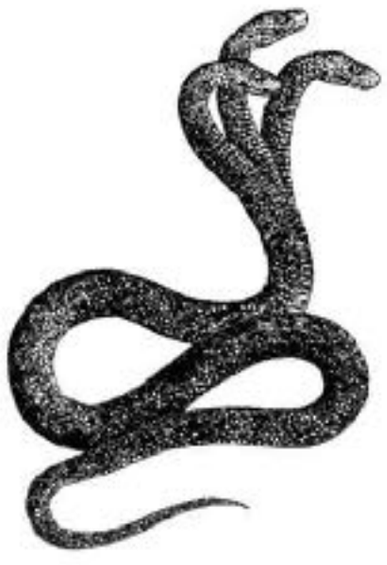
- Set addresses at the beginning and end of `.text`

- Read opcodes (machine code) and look for `int 3` execution (`0xCC`).

- If found, there's a breakpoint in that address.

Anti-VM #1

code: mac.py



- MAC addresses contain Organizational Unique Identified (OUI) in it’s first 3 octets.

- Catch a VM with those.

Anti-VM #2

`code: cpuid.c`

- Opcode: `0x0f 0xa2`
- http://0l.wtf/x86/html/file_module_x86_id_45.html

- Supplies information about the processor being used. Output is stored in `RBX RCX RDX`
- `VBoxVBoxVBox` is a positive value for VirtualBox host detection, for example.

Anti-VM #3

`code: cpu_flags.c`

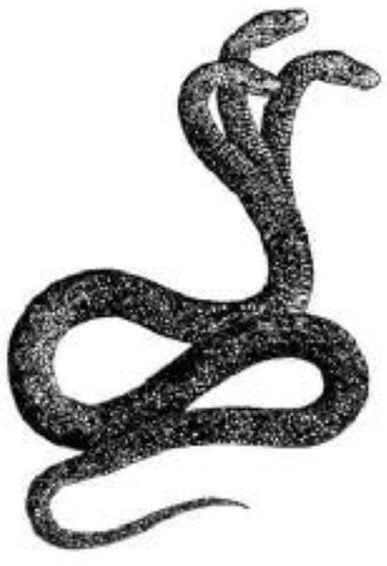
- Grab information from /proc/cpuinfo (or from cpuid(2))

- https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/arch/x86/include/asm/cpufeatures.h#n145

- `hypervisor` flag shows we are running as a guest

Moar info!1

- https://github.com/a0rtega/pafish
- PoC||GTFO 6:9: “Davinci Seal” by Ryan O’Neill. (https://www.alchemistowl.org/pocorgtfo/pocorgtfo06.pdf#41)



- The “Ultimate” Anti-debugging Reference by Peter Ferrie (http://0l.wtf/data/antidebug.pdf)
- `man(1)`