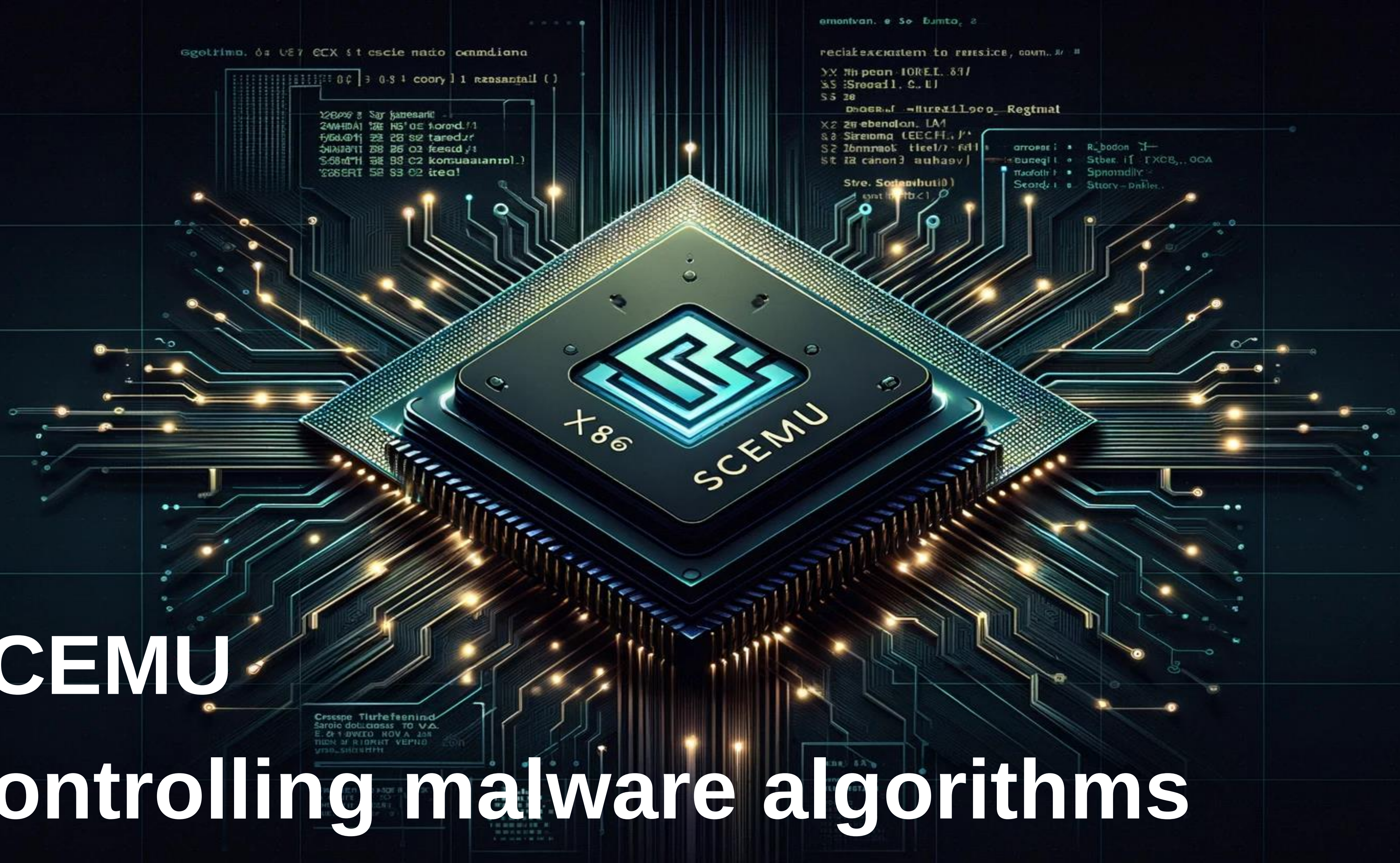


SCEMU

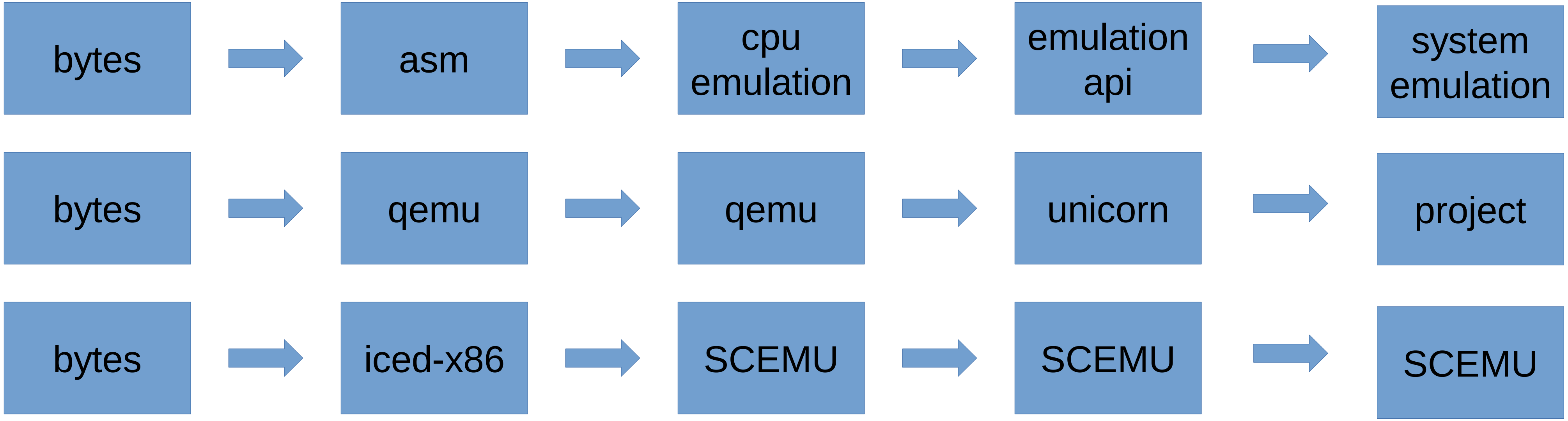
Controlling malware algorithms

FOX IT
part of nccgroup

Who am I

- @sha0coder
- jesus.olmos@fox-it.com
- Fox-it / NCCGroup

Emulation process



bytes → asm → cpu emulation → emulation api → system emulation

bytes → qemu → qemu → unicorn → project

bytes → iced-x86 → SCEMU → SCEMU → SCEMU

Ways of using SCEMU

scemu

pyscemu

libscemu

Features

1. Exceptions SEH + VEH context recovery.
2. Syscalls only linux ones for now.
3. Some Win-API.
4. Normal memory layout (TEB, PEB, LDR …)
5. Dynamic linking.
6. IAT binding.
7. Delay Loading.
8. PE32 + PE64 + ELF64 + Shellcodes.
9. Memory allocator.
10. Floating Point Unit.
11. No unsafe blocks.
12. XMM and YMM instructions.
13. Crypto-api

Limitations

1. CPU not fully implemented.
2. WinAPI not full implemented.

Targets

1. Domain name generation (DGA)
2. Keygen
3. Decryption (strings, configs, etc)
4. Encryption (for emulating communications)
5. Deobfuscation
6. Unpacking – not very effective for packers
7. Understanding

Prepare the context

1. previous function calls
2. global vars
3. Prepare params, buffers, etc.
4. External functions

Banzai mode

1. Keep up emulating cpu, avoid decoy asm.
2. Use list of crawled API params to compensate stack.
3. --banzai   or   emu.enable_banzai_mode()

Bugs

Is easy to have bugs in this type of software.

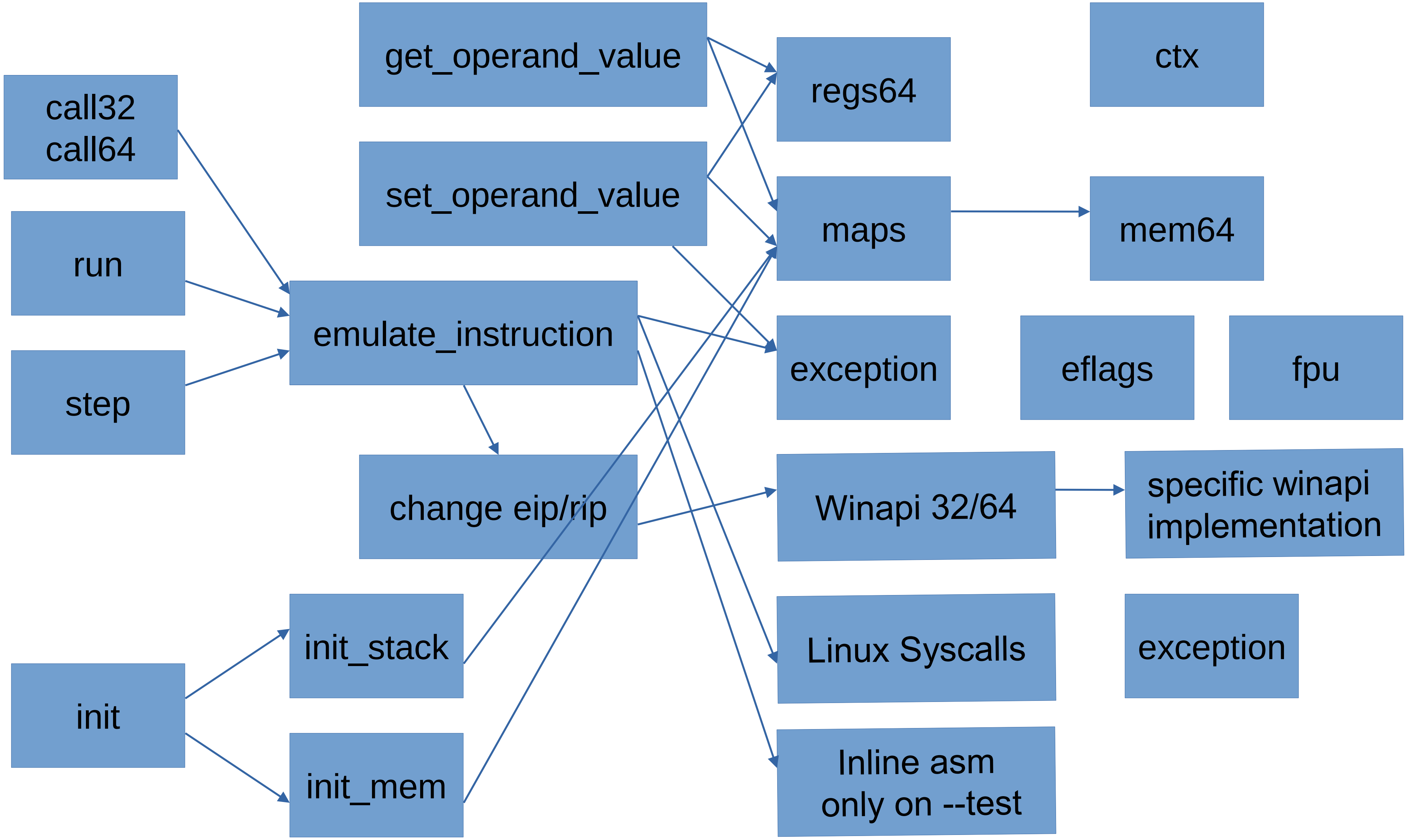
let’s hunt them in an automatic way.

* x64dbg trace diff with SCEMU to make a register and flags diff (by Brandon)

* A --test mode: emulation vs inline assembly.

Don’t use --test mode with malware.

Internals – overview



Internals – Undocumented stuff

1. xmm ymm
2. FPU
3. SHLD / SHRD → undefined behaviours
4. Different types of shifts.
5. Wrongs pseudo-codes in Intel manual.
6. Black box testing

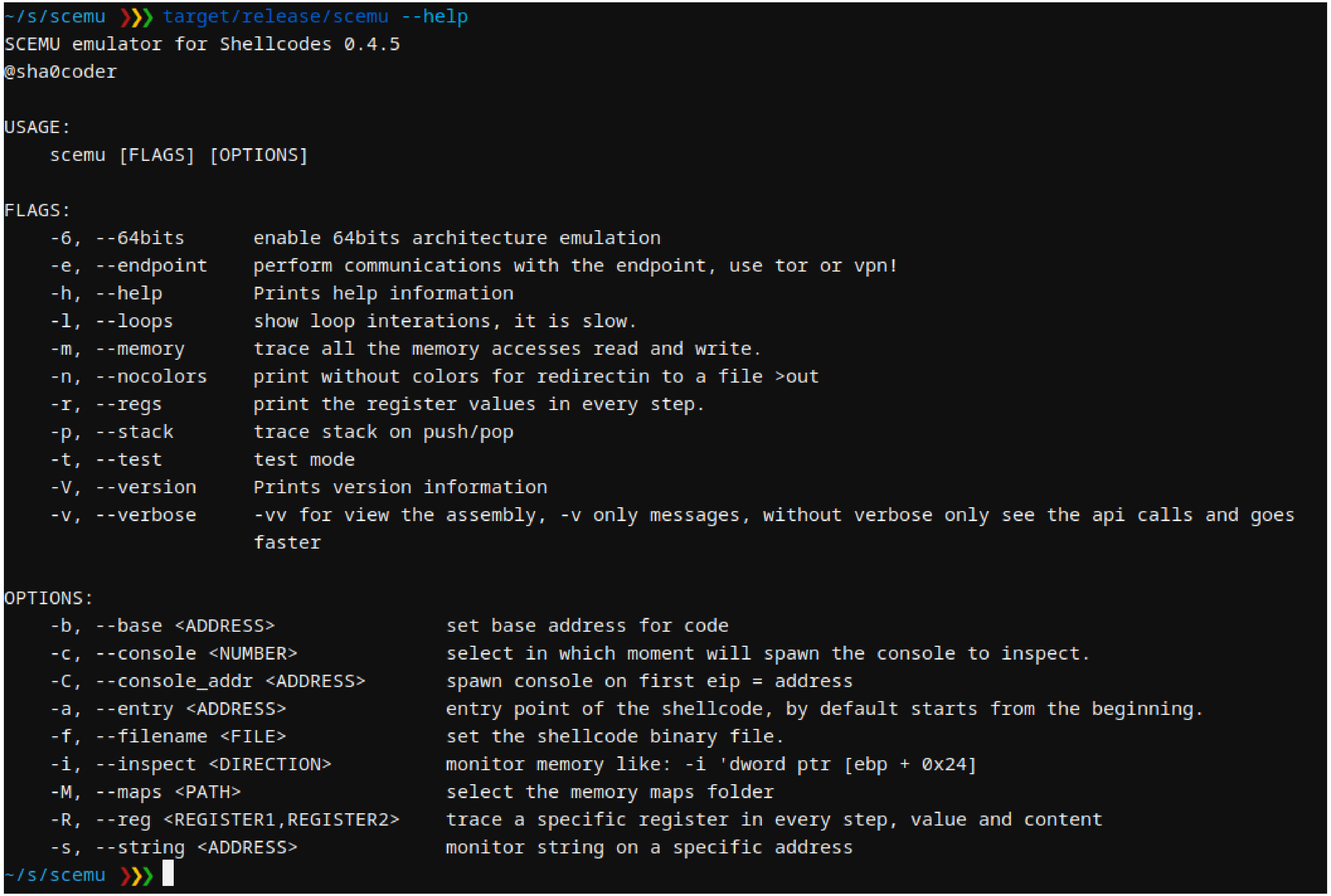
Internals – Flags

1. Substraction flags
2. Logic vs Arithmetic vs bit shifts

Internals – Speed improvements

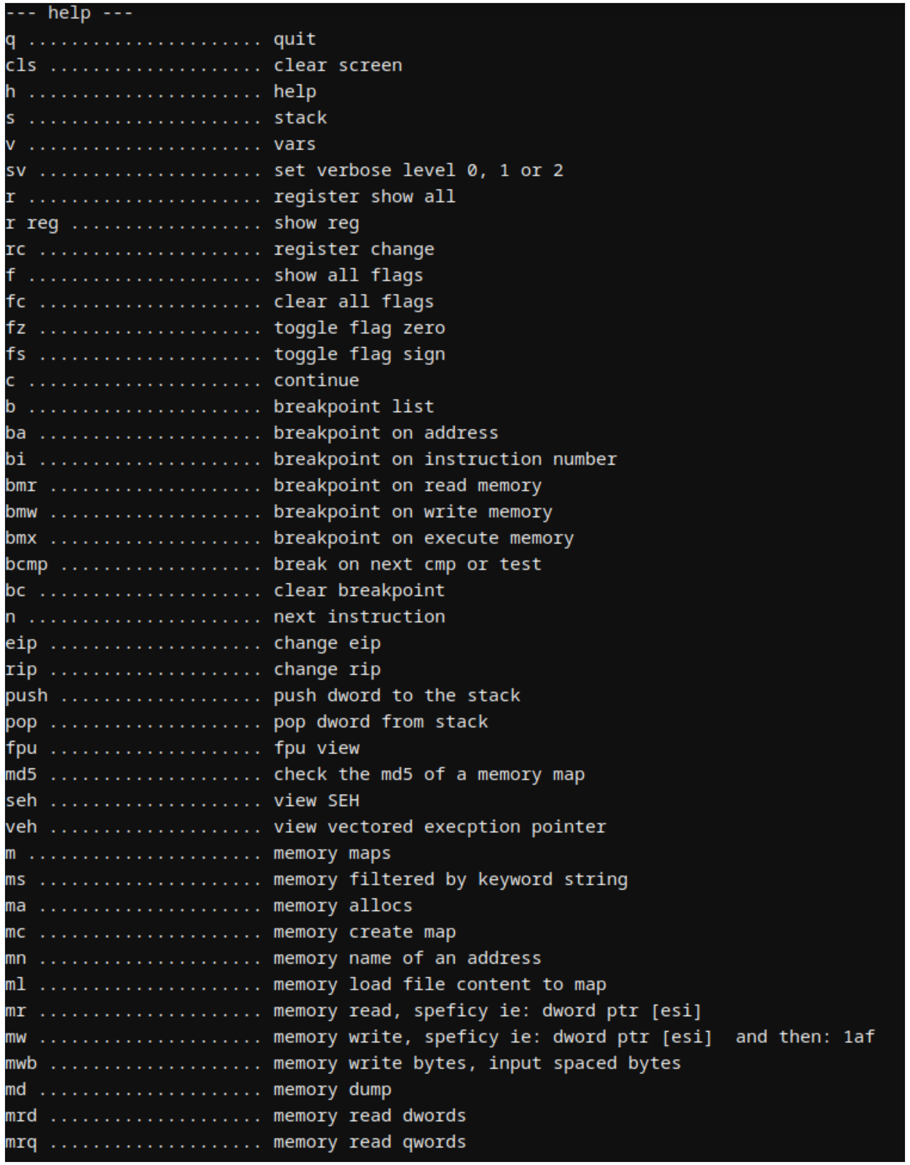
1. bit operations instead of loops.
2. Iced-x86 magic (ie opreands).
3. Simplicity.
4. Don’t use emu.step() use emu.run(until_addr) or call
5. emu.step() should reload smaller block.
6. Fast rep loops.

Demo – tool

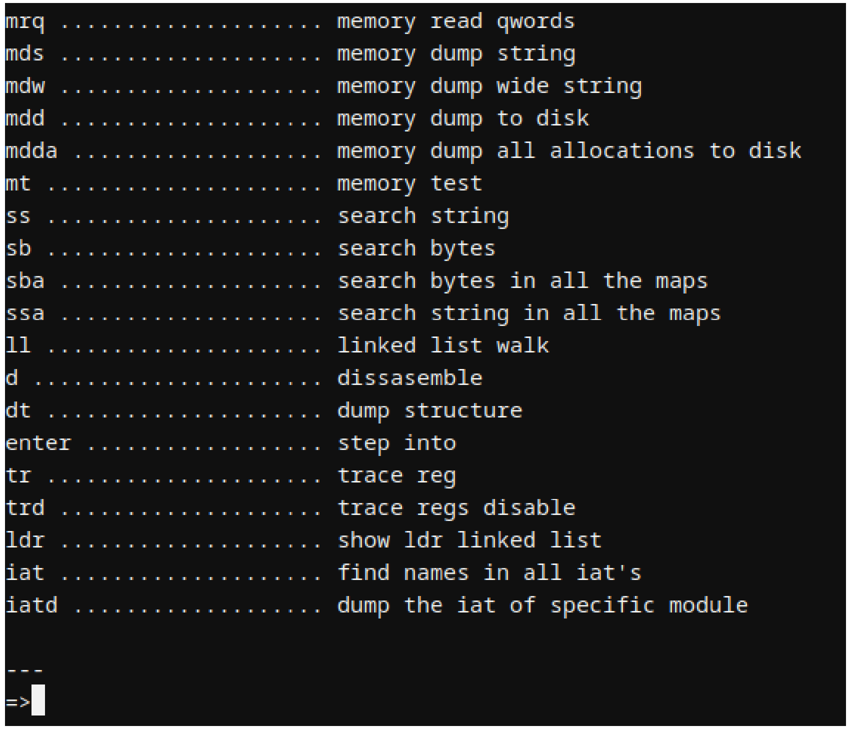


```
~/s/scemu ❯❯❯ target/release/scemu --help
SCEMU emulator for Shellcodes 0.4.5
@sha0coder

USAGE:
    scemu [FLAGS] [OPTIONS]

FLAGS:
    -6, --64bits      enable 64bits architecture emulation
    -e, --endpoint    perform communications with the endpoint, use tor or vpn!
    -h, --help        Prints help information
    -l, --loops       show loop interations, it is slow.
    -m, --memory      trace all the memory accesses read and write.
    -n, --nocolors    print without colors for redirectin to a file >out
    -r, --regs        print the register values in every step.
    -p, --stack       trace stack on push/pop
    -t, --test        test mode
    -V, --version     Prints version information
    -v, --verbose     -vv for view the assembly, -v only messages, without verbose only see the api calls and goes
                      faster

OPTIONS:
    -b, --base <ADDRESS>               set base address for code
    -c, --console <NUMBER>             select in which moment will spawn the console to inspect.
    -C, --console_addr <ADDRESS>       spawn console on first eip = address
    -a, --entry <ADDRESS>              entry point of the shellcode, by default starts from the beginning.
    -f, --filename <FILE>              set the shellcode binary file.
    -i, --inspect <DIRECTION>          monitor memory like: -i 'dword ptr [ebp + 0x24]
    -M, --maps <PATH>                  select the memory maps folder
    -R, --reg <REGISTER1,REGISTER2>    trace a specific register in every step, value and content
    -s, --string <ADDRESS>             monitor string on a specific address
~/s/scemu ❯❯❯
```

Demo – tool

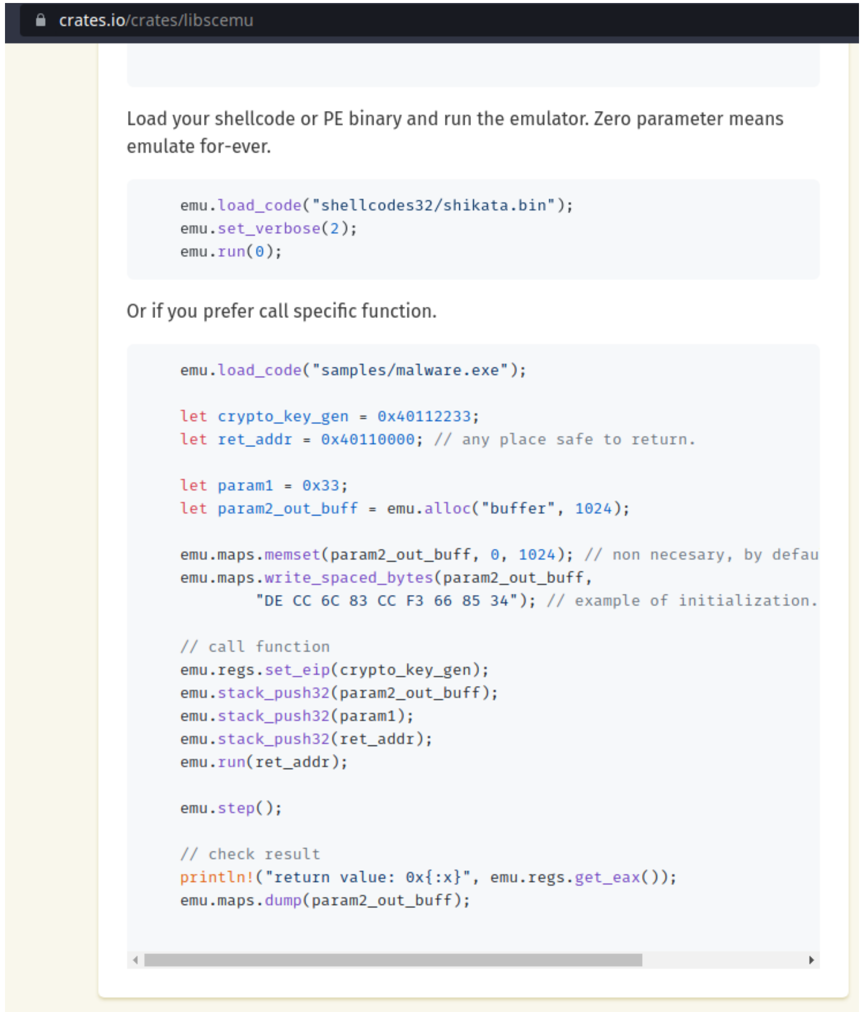


```
--- help ---
q ...................... quit
cls .................... clear screen
h ...................... help
s ...................... stack
v ...................... vars
sv ..................... set verbose level 0, 1 or 2
r ...................... register show all
r reg .................. show reg
rc ..................... register change
f ...................... show all flags
fc ..................... clear all flags
fz ..................... toggle flag zero
fs ..................... toggle flag sign
c ...................... continue
b ...................... breakpoint list
ba ..................... breakpoint on address
bi ..................... breakpoint on instruction number
bmr .................... breakpoint on read memory
bmw .................... breakpoint on write memory
bmx .................... breakpoint on execute memory
bcmp ................... break on next cmp or test
bc ..................... clear breakpoint
n ...................... next instruction
eip .................... change eip
rip .................... change rip
push ................... push dword to the stack
pop .................... pop dword from stack
fpu .................... fpu view
md5 .................... check the md5 of a memory map
seh .................... view SEH
veh .................... view vectored execption pointer
m ...................... memory maps
ms ..................... memory filtered by keyword string
ma ..................... memory allocs
mc ..................... memory create map
mn ..................... memory name of an address
ml ..................... memory load file content to map
mr ..................... memory read, speficy ie: dword ptr [esi]
mw ..................... memory write, speficy ie: dword ptr [esi]  and then: 1af
mwb .................... memory write bytes, input spaced bytes
md ..................... memory dump
mrd .................... memory read dwords
mrq .................... memory read qwords
```



```
mrq ...................... memory read qwords
mds ...................... memory dump string
mdw ...................... memory dump wide string
mdd ...................... memory dump to disk
mdda ..................... memory dump all allocations to disk
mt ....................... memory test
ss ....................... search string
sb ....................... search bytes
sba ...................... search bytes in all the maps
ssa ...................... search string in all the maps
ll ....................... linked list walk
d ........................ dissasemble
dt ....................... dump structure
enter .................... step into
tr ....................... trace reg
trd ...................... trace regs disable
ldr ...................... show ldr linked list
iat ...................... find names in all iat's
iatd ..................... dump the iat of specific module

---
=>
```

Demo – lib



Load your shellcode or PE binary and run the emulator. Zero parameter means emulate for-ever.

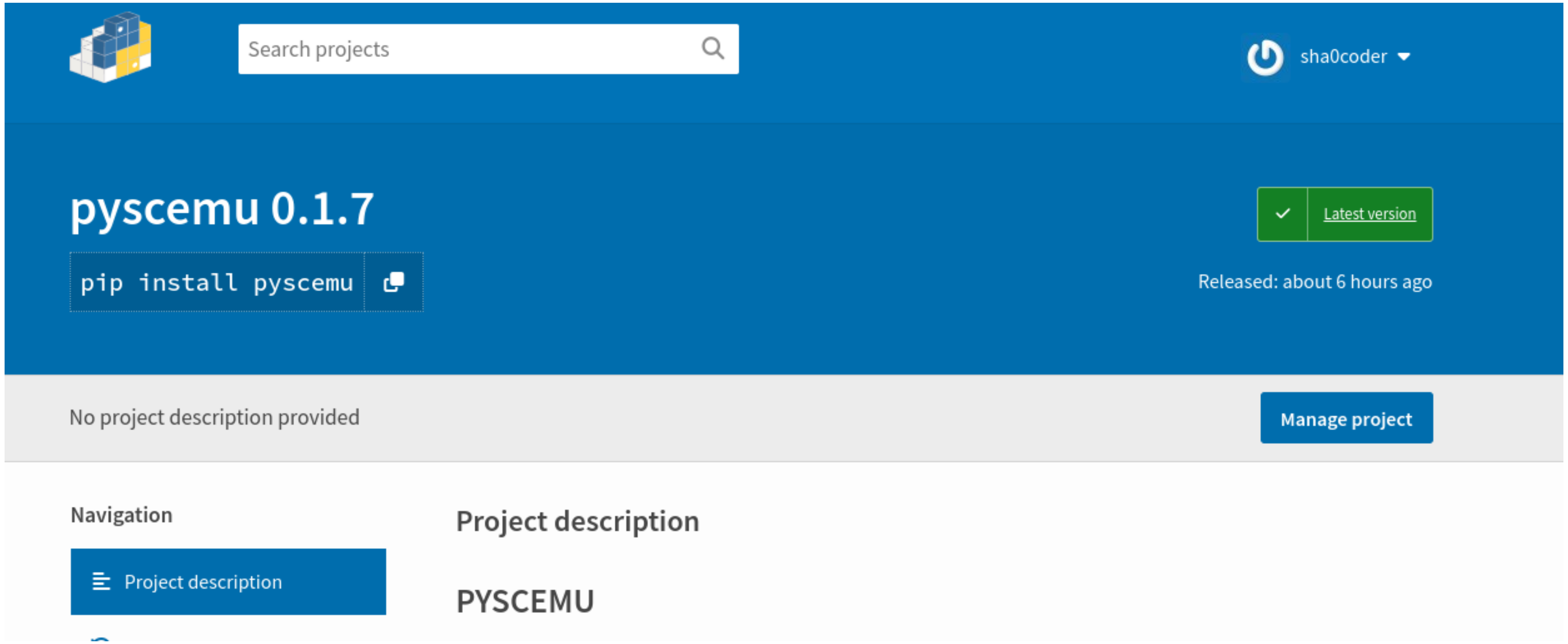
```
emu.load_code("shellcodes32/shikata.bin");
emu.set_verbose(2);
emu.run(0);
```

Or if you prefer call specific function.

```
emu.load_code("samples/malware.exe");

let crypto_key_gen = 0x40112233;
let ret_addr = 0x40110000; // any place safe to return.

let param1 = 0x33;
let param2_out_buff = emu.alloc("buffer", 1024);

emu.maps.memset(param2_out_buff, 0, 1024); // non necesary, by defau
emu.maps.write_spaced_bytes(param2_out_buff,
        "DE CC 6C 83 CC F3 66 85 34"); // example of initialization.

// call function
emu.regs.set_eip(crypto_key_gen);
emu.stack_push32(param2_out_buff);
emu.stack_push32(param1);
emu.stack_push32(ret_addr);
emu.run(ret_addr);

emu.step();

// check result
println!("return value: 0x{:x}", emu.regs.get_eax());
emu.maps.dump(param2_out_buff);
```

https://crates.io/crates/libscemu

Demo – pyscemu



pyscemu 0.1.7

```
pip install pyscemu
```

Released: about 6 hours ago

No project description provided

Project description

PYSCEMU

https://pypi.org/project/pyscemu/

Questions?

Modules:
https://pypi.org/project/pyscemu/
https://crates.io/crates/libscemu

Github:
https://github.com/sha0coder/scemu
https://github.com/sha0coder/libscemu
https://github.com/sha0coder/pyscemu