

PWVN简介与学习方法

讲师：Atum

内容简介

INTRODUCTION

01 PWN简介

02 浅谈PWN学习方法

03 PWN学习案例

PWN简介

软件安全：软件安全专注于研究软件的设计和实现的安全

- 研究对象：代码（源码、字节码、汇编等）
- 研究目标：发掘漏洞、利用漏洞、修补漏洞
- 研究技术：逆向工程、漏洞挖掘与利用、漏洞防御技术

CTF PWN：软件安全研究的一个缩影

- 研究对象：可执行文件，主要是ELF文件
- 研究最终目标：夺取flag

软件安全与CTF PWN特点：入门难、进阶难、精通难

PWN简介

工具

- 静态分析：IDA Pro
- 动态调试：gdb(with peda or gef)、windbg、ollydbg
- Exploit：pwntools、zio

前置技能

- 汇编语言：程序执行、函数栈帧、函数调用等
- 编译、链接、装载、执行
- ELF文件结构
- Linux系统相关：文件描述符、系统调用、socket编程、shell命令

PWN学习方法

痛苦：

- 对于没有入门PWN的人，最最基础的PWN题也可能会让你想破脑袋也想不明白 -> 放弃
- 对于刚刚入门的PWN的人，做进阶的PWN题很容易让你头脑发胀，难以继续 -> 放弃
- 对于已经具有一定水平的PWN手，利用比较难的题目需要审计大量汇编、让人头昏脑胀 -> 放弃

学习PWN需要最重要的品质：**永不言弃！！！！**

怎样玩好LOL？lu多了就会lu了。

怎样学好PWN？ PWN多了就会PWN了。

被PWN题恶心的要死怎么办？被恶心多了就习惯了！

PWN学习方法

学习阶段一：学习套路

- 套路是有限的，假以时日一定会学完的招式
- 针对每种套路都练习1~2道习题

学习完所有常见套路，大多数国内比赛的中档题基本都可以随便切

学习阶段二：总结套路，变套路为艺术

- 漏洞利用是一门艺术，难以用套路完全概括，要想切难题不能完全靠套路
- 多刷刷国际赛的难题，刷的慢没关系，刷多了自然就会融会贯通
- 多总结思考现有的套路的本质

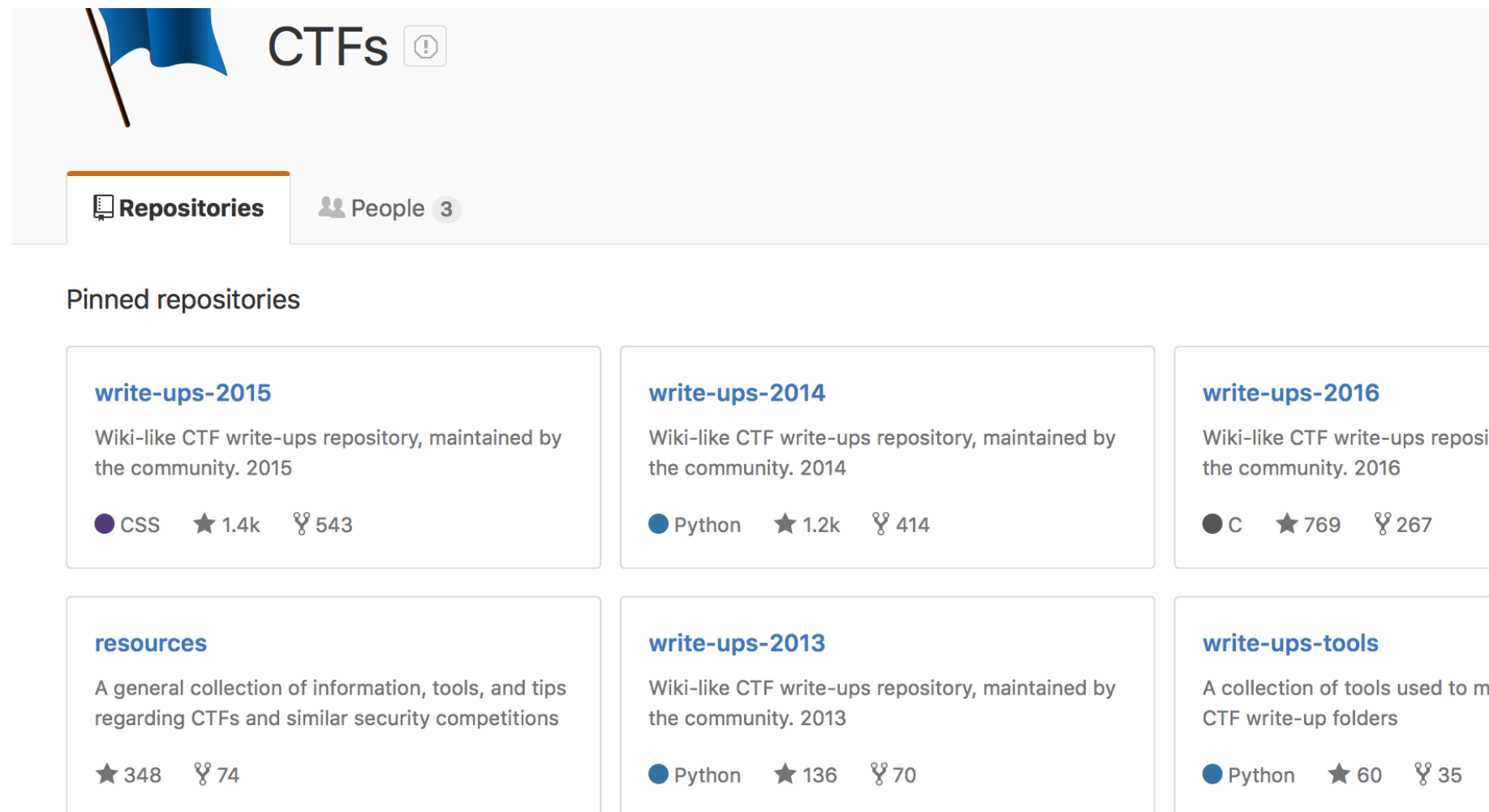
本课程后续内容主要是介绍各种常见的套路，并提供学习资料链接与习题

PWN学习方法

资源：

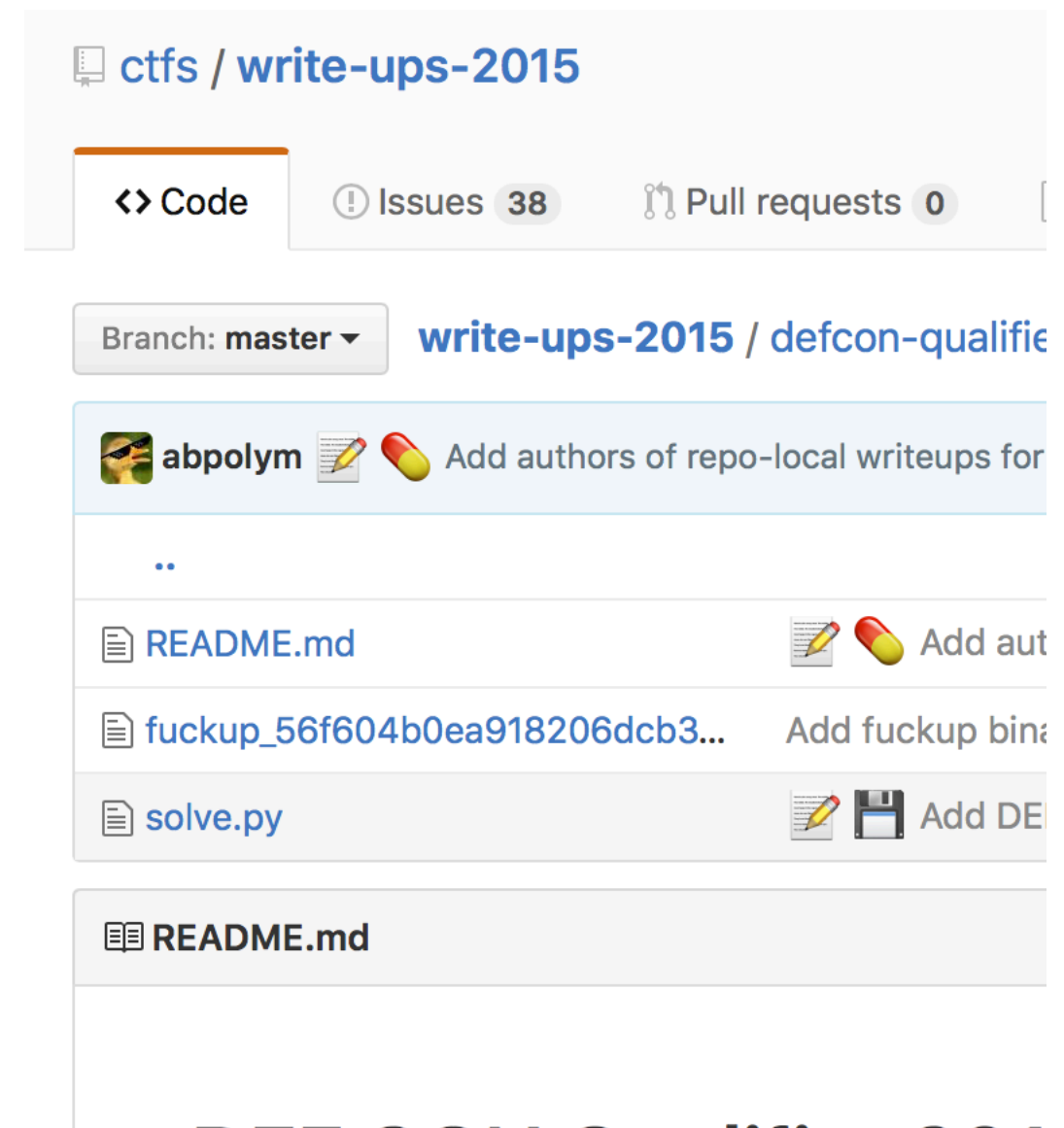
CTF Writeup Github：<https://github.com/ctfs>

聚合了各大国际比赛的习题文件以及writeup



The screenshot shows the GitHub profile page for 'CTFs'. It features a header with a flag icon and the name 'CTFs'. Below the header, there are tabs for 'Repositories' and 'People'. The 'Pinned repositories' section displays six repositories:

Repository Name	Description	Language	Stars	Forks
write-ups-2015	Wiki-like CTF write-ups repository, maintained by the community. 2015	CSS	1.4k	543
write-ups-2014	Wiki-like CTF write-ups repository, maintained by the community. 2014	Python	1.2k	414
write-ups-2016	Wiki-like CTF write-ups repository, maintained by the community. 2016	C	769	267
resources	A general collection of information, tools, and tips regarding CTFs and similar security competitions		348	74
write-ups-2013	Wiki-like CTF write-ups repository, maintained by the community. 2013	Python	136	70
write-ups-tools	A collection of tools used to maintain CTF write-up folders	Python	60	35



The screenshot shows the GitHub repository page for 'ctfs / write-ups-2015'. It includes tabs for 'Code', 'Issues' (38), and 'Pull requests' (0). The 'Branch: master' dropdown is visible. The repository is maintained by 'abpolym'. The file structure is as follows:

- ..
- README.md
- fuckup_56f604b0ea918206dcb3... (Add fuckup bin)
- solve.py (Add DE)
- README.md

PWN学习方法

Googling : XXX Writeup or XXX CTF 不要百度，因为百度收录不了github pages

shitsco writeup

全部

图片

新闻

地图

视频

更多 ▾

搜索工具

找到约 370 条结果 （用时 0.45 秒）

小提示： 仅限搜索简体中文结果。您可以在设置中指定搜索语言

Defcon Quals writeup for Shitsco (use-after-free vuln) » SkullSecurity
<https://blog.skullsecurity.org/.../defcon-quals-writeup-for-shitsco-use-after-fr...> ▾ 翻译此页
2014年5月21日 - Hey folks,. Apparently this blog has become a CTF **writeup** blog! Hopefully you don't mind, I still try to keep all my posts educational. Anyway ...

[write-ups-2014/def-con-ctf-qualifier-2014/shitsco at master · ctfs/write ...](https://github.com/ctfs/write-ups-2014/tree/master/def-con-ctf.../shitsco)
<https://github.com/ctfs/write-ups-2014/tree/master/def-con-ctf.../shitsco> ▾ 翻译此页
Write-up. Shitsco is a small router emulator. It allows you to ping, to tracert, to use the shell (Yhea, right! :D) and to set and view variables. As we smell a rat in the ...

DEFCON 2014 Writeup shitsco · GitHub
<https://gist.github.com/3aa0858e2f540716c832> ▾ 翻译此页
from struct import pack. from socket import *. from pwn import process. #p = process("shit"). p = socket(AF_INET, SOCK_STREAM).

double free ctf

全部

图片

视频

新闻

地图

更多 ▾

搜索工具

找到约 93,200 条结果 （用时 0.36 秒）

小提示： 仅限搜索简体中文结果。您可以在设置中指定搜索语言

Advanced Heap Exploitation: OCTF 2015 'freenote' writeup - kitctf
<https://kitctf.de/writeups/Octf2015/freenote> ▾ 翻译此页
2015年3月30日 - struct malloc_chunk* fd; /* **double** links -- used only if **free**. which directly spawns /bin/sh when we jump to it (from [Dragon's notes on **CTF**]).

Octf 2015 Freenote Write Up - Winesap's Blog - Logdown
winesap.logdown.com/posts/258859-Octf-2015-freenote-write-up ▾ 转为简体网页
2015年3月30日 - March 30, 2015 | 2 Comments. **Double free** 漏洞利用，順便來試試之前想到的某個好用的 unlink() 利用法，建議對heap exploitation 先有一些瞭解 ...

PWN学习案例-ROP

ROP：现代栈溢出中最基础的利用技术（最简单最基础的套路）

Googling 筛选到比较好的题目：r0pbaby

ctf ROP

[全部](#) [图片](#) [视频](#) [新闻](#) [地图](#) [更多](#) [搜索工具](#)

找到约 114,000 条结果（用时 0.27 秒）

小提示： 仅限搜索简体中文结果。您可以在[设置](#)中指定搜索语言

Binary Exploits 2 | CTF Field Guide

<https://trailofbits.github.io/ctf/exploits/binary2.html> [▼ 翻译此页](#)

In this module, we continue to examine the ways that native applications can be exploited and focus on using return-oriented programming (**ROP**) to achieve that ...

write-ups-2013/pico-ctf-2013/rop-2 at master · ctfs/write-ups ... - GitHub

<https://github.com/ctfs/write-ups-2013/tree/master/pico-ctf-2013/rop-2> [▼ 翻译此页](#)

ROP is a classic technique for getting around address randomization and non- executable memory. This sequence will teach you the basics. Problem available ...

Defcon Quals: r0pbaby (simple 64-bit ROP) » SkullSecurity

<https://blog.skullsecurity.org/2015/defcon-quals-r0pbaby-simple-64-bit-rop> [▼ 翻译此页](#)

2015年5月20日 - This past weekend I competed in the Defcon **CTF** Qualifiers from the Legit ... **ROP** - return-oriented programming - is an exploitation technique ...

PWN学习案例-ROP

学习writeup，可以参考多篇writeup，查询不懂的名词

Defcon Quals: r0pbaby (simple 64-bit ROP)

1 Reply

This past weekend I competed in the [Defcon CTF Qualifiers](#) from the [Legit Business Syndicate](#). In the past it's been a pretty tough competition, and this year was no exception!

Unfortunately, I got stuck for quite a long time on a 2-point problem ("wwtw") and spent most of my weekend on it. I was joined by some other folks - r0pbaby included - and am excited to write about them, as well!

[r0pbaby](#) is neat, because it's an absolute bare-bones ROP (return-oriented programming) level. Quite honestly, I actually prefer using a ROP chain to using shellcode. Much of the time, it's actually easier! You can see the binary and other stuff I used on [this github repo](#).

It might make sense to read [a post I made in 2013](#) about a level in PlaidCTF called ropasaurusrex. But it's not really going to explain the same stuff again with two years more experience!

What is ROP?

Most modern systems have DEP - [data execution prevention](#) - enabled. That means that when trying to run arbitrary code, it has to be in memory that's executable. Typically, when a process is running, all memory segments are either writable (+w) - not both. That's sometimes called "[W^X](#)", but it seems more appropriate to just call it common sense.

Return-oriented programming

From Wikipedia, the free encyclopedia

Return-oriented programming (ROP) is a [computer security exploit](#) technique that allows an attacker to execute arbitrary code without the need for a [buffer overflow](#) or [code signing](#).^[1]

In this technique, an attacker gains control of the [call stack](#) to hijack program [control flow](#) and then execute arbitrary code. The code typically ends in a [return instruction](#) and is located in a [subroutine](#) within the existing program and/or shared library. ROP is often used to bypass defenses on a machine employing defenses that thwart simpler attacks.

Contents [\[hide\]](#)

- 1 [Background](#)
 - 1.1 [Return-into-library technique](#)
 - 1.2 [Borrowed code chunks](#)
- 2 [Attacks](#)
 - 2.1 [x86 architecture](#)
- 3 [Defenses](#)
- 4 [See also](#)
- 5 [References](#)
- 6 [External links](#)

Background [\[edit\]](#)

PWN学习案例-ROP

根据Writeup和学到的内容自己动手调试一下，尽量自己手动重写exploit

重写exploit成功：基本掌握了ROP

```
(gdb) b *0x555555554000+0xe53
Breakpoint 1 at 0x555555554e53
(gdb) c
Continuing.
Stopped due to shared library event:
  Inferior loaded /lib/x86_64-linux-gnu/libdl.so.2
  /lib/x86_64-linux-gnu/libc.so.6
(gdb)
Continuing.

Welcome to an easy Return Oriented Programming challenge...
Menu:
Stopped due to shared library event (no libraries added or removed)
(gdb)
Continuing.
1) Get libc address
2) Get address of a libc function
3) Nom nom rop buffer to stack
4) Exit
: 3
Enter bytes to send (max 1024): 8
AAAAAAA

Breakpoint 1, 0x0000555555554e53 in ?? ()
(gdb) x/12wx $rsp
0x7fffffffef140: 0xf7814d28      0x00007fff      0xf7ff79b0      0x00007fff
0x7fffffffef150: 0x41414141      0x0a414141      0x01f25bc2      0x00000000
0x7fffffffef160: 0xfffffe270     0x00007fff      0xf7de4991      0x00007fff
(gdb) x/4x $rbp
0x7fffffffef590: 0x41414141      0x0a414141      0xf7832ec5      0x00007fff
```

The image features a white background with two large yellow geometric shapes in the corners. The shape in the top-left corner is a triangle pointing towards the center. The shape in the bottom-right corner is a more complex polygon, also pointing towards the center, with a smaller yellow triangle nested within its lower right portion.

The End