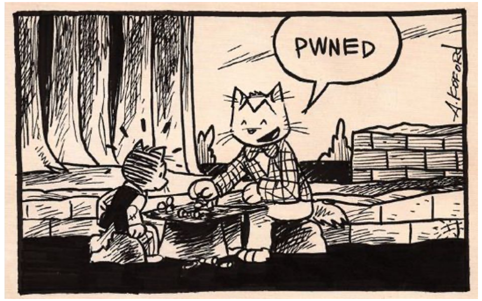
Attention

- 课程 PPT 以及调试材料已经上传学在浙大
- 作业已布置在课程网站
- 今晚有答疑，欢迎来 Q&A

PWN Basics

PWN基础-1 @ f0rm2l1n



Outline

- PWN 引言
- 代码注入漏洞
- 栈上缓冲区溢出漏洞

PWN 引言

PWN 是什么？

PWN = Find the Bugs + Exploit them

Bug Definition

A Software Bug is **a failure or flaw** in a program that produces **undesired or incorrect results**. It’s an error that prevents the application from **functioning as it should**.

Digression: it’s just so hard to define “BUG”

beliefs are facts about the system implied by the code, can flag **belief contradictions** as errors

- Bugs as deviant behavior: A general approach to inferring errors in systems code [SIGOPS2021]

CTF PWN Bugs

- C/C++ language
  ↳ **memory corruption bugs**
- Clear exploitation aim
  ↳ **code execution**
- Naive program
  ↳ **usually terminal program**

CTF PWN Bugs cont.

- C/C++ language
  ↳ **memory corruption bugs** ⇨ **other complex language**
  ↳ logic bugs
- Clear exploitation aim
  ↳ **code execution**
- Naive program
  ↳ **usually terminal program** ⇨ **other complex target**
  ↳ iot, httpd, kernel, browser, ...

Talk is less ... example-nocrash

- 参见课堂网站的 *[lec1] pwn nocrash*
- 阅读源代码，找到程序的漏洞
- 本地运行并触发该 bug
- 与远端交互、触发 bug 并获取 flag

nocrash explain

Talk is less ... login_me

- 参见课堂网站中的 *[lec1] pwn login_me*
- 尝试逆向找到其包含的 bug (其实也给了源码
- 本地运行并利用该 bug
- 与远端交互、利用 bug 并获取 flag

login_me explain

Notes

pwn 赛题结构

- 赛题文件
  - 往往需要逆向
  - 漏洞描述 (diff)

- 赛题环境
  - libc and ld
  - Dockerfile
  - "good challenge should issue everything you needed to run and test it"

- 赛题远程

赛题远程

- xinetd: http://www.xinetd.org/
- 快速将命令行文本程序搭建为 TCP 服务
- （别在 host 上直接跑服务）

Notes cont.

I’d love it: system(“/bin/sh”);

- backdoor
- execve
- *sometime not necessarily a shell*

Quick Review

- PWN = find bugs + exploit them
- PWN Challenge 010

一个测试工程师走进一家酒吧，要了一杯啤酒

一个测试工程师走进一家酒吧，要了一杯咖啡

一个测试工程师走进一家酒吧，要了0.7杯啤酒

一个测试工程师走进一家酒吧，要了-1杯啤酒

一个测试工程师走进一家酒吧，要了2^32杯啤酒

一个测试工程师走进一家酒吧，要了一杯洗脚水

一个测试工程师走进一家酒吧，要了一杯蜥蜴

一个测试工程师走进一家酒吧，要了一份asdfQwer@24dg!&*(@

一个测试工程师走进一家酒吧，什么也没要

一个测试工程师走进一家酒吧，又走出去又从窗户进来又从后门出去从下水道钻进来

一个测试工程师走进一家酒吧，要了一杯烫烫烫的锟斤拷

测试工程师们满意地离开了酒吧。然后一名顾客点了一份炒饭，酒吧炸了

代码注入漏洞

代码注入 (Code Injection)

“An attacker introduces (or "injects") code into the program and changes the course of its execution. “

- 原始 + 直接的漏洞与攻击
- “相对容易”检测 - 特征函数
- “相对容易”防御 - 白名单/黑名单

代码注入 (Code Injection cont.)

- 命令注入
    - 直接

- shellcode 注入
    - 间接
    - 搭配控制流劫持的利用方式

命令注入 inject1

- 参见随堂材料中的 *inject1*
- 逆向并找到代码注入漏洞

inject1 explain

shellcode 注入 inject_me

- 参见课堂网站的 *[lec1] pwn nolocrash*
- 尝试逆向找到其包含的 bug
- 本地运行并触发该 bug
- 与远端交互、触发 bug 并获取 flag

Notes and Explaination

- 函数指针与 indirect call
- Segmentation fault (revisit)
- mmap and munmap
    - mprotect

Quick Review

- Never trust user-provided Code
  - malicious commands
  - shellcode (RWX)

- Defense Strategy
  - privilege checking & sandboxing
  - verifier
  - command and data

Stack Buffer Overflow

stack 是什么？

- 先进后出 FILO (First-in-last-out) 的数据结构
  - visualization
- 由内核申请的，进程特殊的内存区域
  - example
- 存放临时变量
  - 作用域

stack 的实现

往往基于两个指针

- stack pointer
- frame pointer

往往提供两个原语

- push
- pop

Stack-Based Virtual Machine, like EVM

Example by Debugging

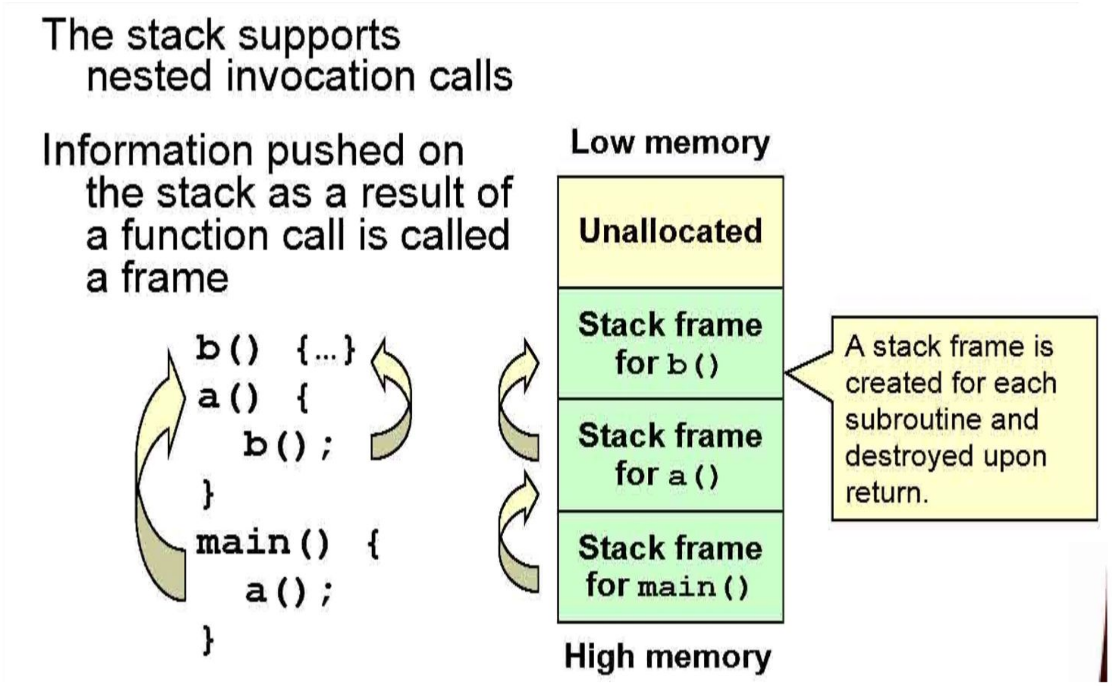
(static analysis also enough)

- debug login_me

- draw stack graph

- many other stack challenges
  - Protostar

CallStack and Backtrace



The stack supports nested invocation calls

Information pushed on the stack as a result of a function call is called a frame

```
b()  {…}
a()  {
  b();
}
main()  {
  a();
}
```

A stack frame is created for each subroutine and destroyed upon return.

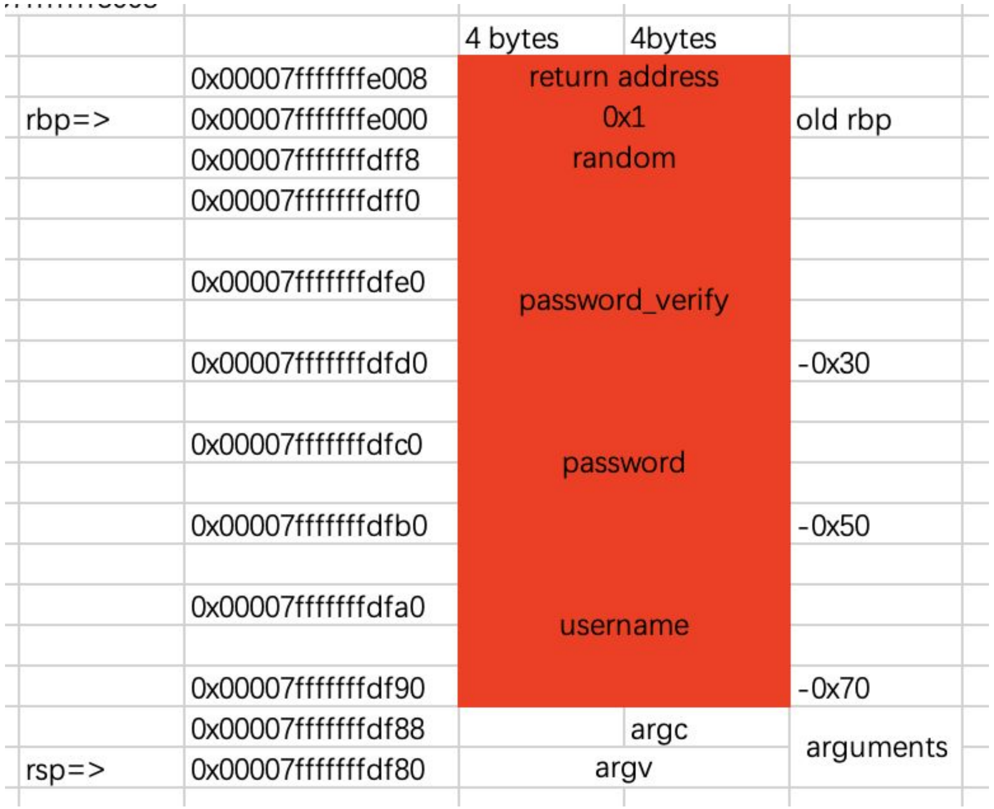
*Slide by Robert Seacord

Question:

有寄存器为什么还需要栈来存储临时变量?

stack overflow 是什么

蠕虫病毒



		4 bytes	4bytes	
	0x00007fffffffe008	return address		
rbp=>	0x00007fffffffe000	0x1		old rbp
	0x00007fffffffdff8	random		
	0x00007fffffffdff0			
	0x00007fffffffdfe0			
		password_verify		
	0x00007fffffffdfd0			-0x30
	0x00007fffffffdfc0			
		password		
	0x00007fffffffdfb0			-0x50
	0x00007fffffffdfa0			
		username		
	0x00007fffffffdf90			-0x70
	0x00007fffffffdf88		argc	arguments
rsp=>	0x00007fffffffdf80	argv		

stack overflow 能力

- 溢出破坏局部变量
- 溢出破坏存储的栈帧指针
- 溢出破坏存储的返回地址

stack overflow 能力

● 溢出破坏局部变量	⇨	● **破坏数据流**
● 溢出破坏存储的栈帧指针	⇨	● **栈迁移**
● 溢出破坏存储的返回地址	⇨	● **控制流劫持**

sbof example - sbof1

Notes

- data-only 的攻击是难以防御的
- data-only 的攻击的利用非常复杂
    - 和程序逻辑息息相关
- 溢出长度的计算

sbof example - sbof2

Notes

- PIE（Position-Independent Executable）”保护”
- 静态链接、动态链接
- 后门检测

sbof example - sbof3

sbof + shellcode

Code Injection is dead?

- 如果不是 mmap，没有 PROT_EXEC
- W^X 保护
- 代码复用攻击

Takeaway

- pwn 赛题基础(本地/远程)
- 代码注入及其利用
- 栈上缓冲区溢出及其利用