Wargama-leviathan Writeup

转载

weixin_34034670 于 2018-03-08 10:55:51 发布 64 收藏
文章标签： python 运维 shell
原文链接：https://juejin.im/post/5aa116b7f265da238d504b11

> litao3rd · 2015/03/24 11:51

level 0

这一关就是一个简单的游戏介绍，当然，还有明文的账号密码。打开网页就能看到，不再叙述。

level 0 -> 1

使用上一关得到的账号和密码，ssh 登陆到目标机器。

在主目录下面，我们可以发现一个.backup文件夹，进入文件夹，发现了一个html文件，看样子是浏览器收藏夹文件。我猜测密码应该是在文件中以明文形式出现，用关键字leviathan进行搜索，立刻就能得到密码。the password for leviathan1 is rioGegei8m。这道题应该就是让你熟悉命令的。

level 1 -> 2

使用上一个level得到的密码，ssh到下一个level。在这关的home目录里面，我们看到一个名为check的set-uid程序，执行程序，看看程序执行结果。

程序要求我们输入一个密码，好吧。我不知道该输入什么，猜想程序应该是将我们的输入和某个密码做比较，然后再执行。随便输入一个，观察结果。

打开GDB神器，看看它内部到底是个什么东西。

从图中我们可以看看出来，程序三次调用getchar()，猜想，password应该是三个字符。strcmp()函数的一个参数是我们输入的字符的存储起始地址，另一个地址是0x18(%esp)，程序一开始在这里存储了一个值0x786573，然后将两个地址作为实参送入strcmp()，所以猜想，这里0x18(%esp)就是我们需要的密码。这个值正好是sex的十六进制格式，于是，得到密码是sex。输入程序，得到下一个level的密码。

level 2 -> 3

在这个level的home目录里面，依然是一个set-uid的程序，执行程序得到如下结果，还是要使用神器gdb。

使用gdb打开程序，这个程序主函数反汇编出来的结果有点长，不过貌似没有看到子函数调用，只有一个主函数。

这个程序首先进行了参数检查，如果没有命令行参数输入，则打印出错信息，然后退出。之后使用access()进行文件权限检查，如果没有权限，则打印出错信息，然后退出。然后再使用system()函数执行/bin/cat fileinput这个命令。

- 根据access()的手册，access() checks whether the calling process can access the file pathname. If **pathname** is a symbolic link, it is dereferenced.

这就意味着我们不能利用符号链接来过这个权限检查。

从gdb的反汇编代码，我们可以得到，程序使用输入的命令行参数构造了/bin/cat argv[1]这个字符串，然后送入system()这个函数执行。于是我猜想到构造这么一个字符串/bin/cat < /etc/leviathan_pass/leviathan2。这样就可以通过权限检查，然后得到密码了。

- 构造一个文件，名为<file
- 构造一个符号链接， file->/etc/leviathan_pass/leviathan2

这样，系统进行权限检查的时候，则检查的是<file这个文件的权限，但是执行命令的时候却是/bin/cat < file这个命令。

level 3 -> 4

还是一个set-uid程序，老流程走起，打开程序，看看它是那个小怪。

还是要gdb走起啊。

这里的strcmp()函数为什么调用我没有理解，或许是为了干扰吧，在main()函数中有一个do_stuff()函数调用，整个程序的逻辑应该在这个函数里面。

可以看到，在do_stuff()函数中有strcmp()和system()等比较，strcmp()的其中一个实参来自于fgets()函数的返回结果，另一个参数存储在-0x117(%ebp)，从图中可以看出来，存储在-0x117(%ebp)的参数就是待匹配的密码。

将密码输入到程序中，就可以得到一个**shell**，进而得到下一个level的密码。

level 4 -> 5

好吧，这一个level的文件在.trash文件夹中了，依旧是熟悉的流程。

从程序输出看，应该是密码被转换成二进制输出了，或者是加密之后转换成二进制输出，这个还是得要祭起gdb神器来帮忙了。

从这个循环过程中，我们可以看到，程序对密码的每个字符，从最高位（符号位）开始计算，每次输出一个字符0或者字符1，所以可以得到，该bin程序是将密文的每个字符转换成二进制格式输出。单步调试的时候，程序在调用fopen("/etc/leviathan_pass/leviathan5", "r")的时候返回错误，导致调试无法进行。看来，直接查看内存获得密码的方法是行不通了，只能写脚本，处理程序的输出了。

```
#!python
#!/usr/bin/env python
#coding=utf-8

if __name__ == "__main__":
        try:
                fp = open("log.txt", "r")
        except:
                print "file open error"
        for i in range(12):
                byte = fp.read(9).strip()
                value = (int(byte, 2)) & 0xFF
                #print value
                char = chr(value)
                print char,
```
复制代码

> **python代码写得不是一丁点的丑，实在是因为用得不多！**

把密码中的空格剔除就是正确的密码了。

level 5 -> 6

还是熟悉的流程，熟悉的味道。

似乎`/tmp/file.log`这个路径被硬编码到了程序中了，还是要看看这个程序到底做了什么事情。

确实是被硬编码到了程序中去了，尝试着在该路径下建立一个指向密码文件的**符号链接**文件，发现居然成功了。

好吧，这一个level实在是有点开玩笑啊。。。。

level 6 -> 7

哈哈，这个level似乎有不一样的味道，虽然还是熟悉的流程。

这里，这个`set-uid`程序需要输入一个`4 digit code`，懒得再去用gdb反汇编了，我估计也不太好反，做了混淆应该。干脆就来个爆破吧。

```
#!python
#!/usr/bin/env python
#coding=utf-8

import os

for pincode in range(10000):
        cmd = "~/leviathan6 %04d" %(pincode)
        # for debug
        #print "test %s" %(cmd)
        ret = os.popen(cmd).read()
        if ret.find("Wrong") == -1:
                print "maybe success in %s" %(cmd)
```
复制代码

程序在7123那里停止了，我猜想是跑到了shell里面去了。尝试了一下，果然是这样的。

level 7

---

ssh登陆到level 7

好吧，我纯粹是为了混一个邀请码，才写这个writeup的。要不然我肯定遵守这个规则！

end

Bingo！这个wargame就这样结束了，题目确实挺简单的，完全是为了新手学习，知道怎么玩wargames。

更多关于wargames的信息，请参考这里!