Pwnable.kr collision [Writeup]

原创

c01dkit 于 2021-02-25 20:34:30 发布 47 收藏

分类专栏： pwn 文章标签： pwn

更多writeup将更新于个人博客，随缘同步到CSDN，如有需要，请移步此处

题源

https://pwnable.kr/play.php

> Daddy told me about cool MD5 hash collision today.
> I wanna do something like that too!
> ssh col@pwnable.kr -p2222 (pw:guest)

题解

Pwnable.kr的第二道题，主要考察关于C语言基础知识。

关于ssh和pwn题目部署的关系，可以参考上一篇博文1，此处略去不表

查看程序源码如下：

```c
#include <stdio.h>
#include <string.h>
unsigned long hashcode = 0x21DD09EC;
unsigned long check_password(const char* p){
 int* ip = (int*)p;
 int i;
 int res=0;
 for(i=0; i<5; i++){
  res += ip[i];
 }
 return res;
}

int main(int argc, char* argv[]){
 if(argc<2){
  printf("usage : %s [passcode]\n", argv[0]);
  return 0;
 }
 if(strlen(argv[1]) != 20){
  printf("passcode length should be 20 bytes\n");
  return 0;
 }

 if(hashcode == check_password( argv[1] )){
  system("/bin/cat flag");
  return 0;
 }
 else
  printf("wrong passcode.\n");
 return 0;
}
```

直接看main函数，程序要求传入两个参数，且第二个参数需要有20个字节。当第二个参数按check_password函数执行后如果返回0x21DD09EC则得到flag。

分析check_password函数可知，该函数首先将传入的字符指针视作int类型指针，然后从此地址依次取5个int进行求和，最后返回相加结果。由于这里int是4字节，因此正好需要20个字节，和main的要求相契合。

因此，该程序要求简单来说就是：传入这样一个20字节的字符串，把它切割成5个int求和结果为0x21DD09EC。

该程序以**32bit小端序**运行（可以`file col`命令看一下），因此这里需要特别注意两点是：

- 字符串在内存上的排布是**起始地址在低地址，向高地址生长**，而int的解析方式是**高地址为高位，低地址为低位**。
- 每个字符占1字节，即8bits，4个字符占4字节，即32bits，即一个int。

举例而言，传入字符串`b'\x01\x02\x03\x04'`[2]将被解析为`0x4030201`。

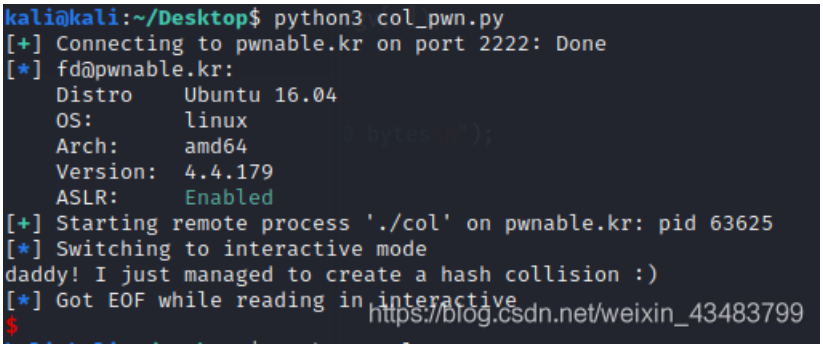
所以要凑成0x21DD09EC还是比较简单的。我也懒得算具体数值，直接看代码一目了然：

```python
from pwn import *
ssh = ssh('col','pwnable.kr',2222,'guest')
hashcode = 0x21DD09EC

# 基本思路是找平均数，如果有余数就都分给其中一个数就行了
payload = hashcode//5
remainder = hashcode%5 + payload

# 将数字转化为二进制字符串（p32函数默认返回数字的小端序二进制字符串）
payload = 4*p32(payload) + p32(remainder)
sh = ssh.process(['col',payload],'./col')
sh.interactive()
```

执行结果：



总结

由于还是pwnable.kr的入门题，所以这题也很简单。关于题目描述里的MD5哈希碰撞，其实和题目内容没有甚么关系3。至于MD5算法[4]，是一种将任意长度字符串映射为一个32字节字符串的单向散列函数。由于可以对任意长的数据进行加密，而映射结果是固定长度，因此存在着无穷多的碰撞可能性（不过和这个题目无关hhh）。通过这题应该了解到：

- C语言指针的基础用法、程序数据存放与解析的方式
- pwntools中p32函数的用法

Pwnable.kr fd [Writeup] ↩

这里采用python的形式表达比较方便，二进制表达是`0000 0001 0000 0002 0000 0003 0000 0004` ↩

此处向王小云院士致敬！ ↩

MD5简单介绍 ↩