【pwnable.kr】collision

原创

搓雪小怪兽 于 2019-04-12 13:01:44 发布 120 收藏 1

分类专栏： PWN 文章标签： PWN CTF

本文链接：https://blog.csdn.net/qq_33892117/article/details/89233300

ssh登录，源码如下：

```
#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;
}
```

这道题的代码也比较简单，输入20个字符，然后将这二十个字符每四个转换为一个int，相加之后与存储的hash比较。每四个char转换为int的公式为

`int = char[0]+char[1]*256+char[2]*256*256+char[3]*256*256*256`

由于每个char是8bit所以可以看成是一个256进制的数来计算int。

由于一开始没有想到可以通过python向col程序传入不可打印的字符，所以真的在遍历哪些可见字符能符合条件。但是后来发现即使是值最小的可见字符串加起来也要比给定的hash大很多。
经过别人的writeup的提醒，才知道可以将指令的运行结果作为程序的命令行参数。
于是构造payload如下即可：

```
col@ubuntu:~$ ./col `python -c "print '\x01'*16+'\xE8\x05\xD9\x1d'"`
daddy! I just managed to create a hash collision :)
```

使用 `\x01` 进行填充是因为如果用 `\x00` 会发生截断。还要注意一下小端序的问题(低位放在低地址)。

补充：其他的关于将指令运行结果作为程序输入的方法：

```
#Use command output as an argument
./vulnerable `your_command_here`
./vulnerable $(your_command_here)
#Use command as input
your_command_here | ./vulnerable
#Write command output to file
your_command_here > filename
#Use file as input
./vulnerable < filename
```