HCTF GAME Week 1 解题报告

21 Jan 2017

突然被叫去参加了hctf的入门级ctf，第一周打得还不错，总分第一。虽然因为是入门级被我妈冷嘲热讽……

第二周大概不会再认真做了，除非有别的奖品。→_→

大概，按照我做的大类的顺序来。

Web

还不错，全部题目进前三做完的，两题第一，多亏我单身多年。不过说实话php是我的弱项。虽说作为一个web选手，但是php还是弱得要死，真不像话……

这TM是啥

http://115.28.78.16:13333/web/web1/

解题报告

呃，限于篇幅我就不发代码了。一眼看过去就是jsfuck的代码，所以直接去掉最后一对括号扔浏览器里跑。

结果卡死了我的浏览器，没法复制flag了……想着把这个flag手打一下提交上去吧，打错了……

最后js的代码是

```
function anonymous() {
  var f = "hctf{j5fuck_1z_m1233}";
  alert("Hack by LoRexxar, 你的flag被我拿走了")
}
```

我是谁我在哪？？？

http://115.28.78.16:13333/web/web2/index.php

解题报告

习惯性 `curl --include` 的我没有任何困难拿下了flag……其实一开始浏览器打开后什么都没有我是猜在header里，后来复制链接的时候注意到题目给的是index.php，结果浏览器地址栏显示index.html我就知道发生了什么。curl过来的结果是 `hctf{1t_iz_4_4mall_tr1ck}` ，也注意到有个 `location: index.html` 给我跳转了。

神奇的数字

```php
<?php
if(empty($_POST)){
    highlight_file(__FILE__);
    exit;
}
include_once("flag.php");

function is_palindrome_number($number) {
    $number = strval($number);
    $i = 0;
    $j = strlen($number) - 1;
    while($i < $j) {
        if($number[$i] !== $number[$j]) {
            return false;
        }
        $i++;
        $j--;
    }
    return true;
}
ini_set("display_error", false);
error_reporting(0);
$info = "";
$req = [];
foreach([$_GET, $_POST] as $global_var) {
    foreach($global_var as $key => $value) {
        $value = trim($value);
        is_string($value) && is_numeric($value) && $req[$key] = addslashes($value);
    }
}

$n1 = intval($req["number"]);
$n2 = intval(strrev($req["number"]));

if($n1 && $n2) {
    if ($req["number"] != intval($req["number"])) {
        $info = "number must be integer!";
    } elseif ($req["number"][0] == "+" || $req["number"][0] == "-") {
        $info = "no symbol";
    } elseif ($n1 != $n2) { //first check
        $info = "no, this is not a palindrome number!";
```

```
    } else { //second check
        if(is_palindrome_number($req["number"])) {
            $info = "nice! {$n1} is a palindrome number!";
        } else {
            $info = "find a strange dongxi: " . $FLAG;
        }
    }
} else {
    $info = "no number input~";
}
echo $info;
?>
```

解题报告

超烦的php源码审计……这题嘛，利用 `intval` 和 `is_numeric` 处理数字时的区别， `is_numeric` 会让小数也通过检测，而 `intval` 会在遇到小数（非数字字符）的时候截断只要整数部分，然后我们就可以搞事儿了。

对于php来说，如果一个整数比机器位数还长，那它就gg了。经过简单检测可以知道目标机为64位机，于是我们构造一个reverse前后都超大的数字即可（太大了也会被当作浮点数……所以还是用 `(1 << 63) - 1` 比较好）。

我们构造出来的数字是 `9223372036854775807.00000000000000000001` ，往上POST即可，拿到flag为 `hctf{go0d_job_intv4l_iz_g00d}` 。

不可能拿到的flag

```
<?php
if(empty($_POST)){
    highlight_file(__FILE__);
    exit;
}
include_once("flag.php");

if (isset($_POST['name']) and isset($_POST['password'])) {
    if ($_POST['name'] == $_POST['password']){
            print 'Your password can not be your name.';
        }else if (sha1($_POST['name']) === sha1($_POST['password'])){
        die('Flag: '.$flag);
    }else{
    print 'Invalid password';
    }
}
?>
```

检查输入字符串的SHA1是否相同，本来真还打算去找一对来着……

这个主要利用了php的一个漏洞，我们可以传入数组，而 `sha1` 在面对数组的时候返回的都是 `false` ，过关。

POST的是 `name[]=1&password[]=2` ，flag是 `hctf{o0k!!g3t_f14g_s0_ez}` 。

php真可怕我要回农村

```php
<?php
if(empty($_POST)){
    highlight_file(__FILE__);
    exit;
}
include_once("flag.php");

$a= "0.1";
$b= $_POST['b'];
if($b != ''){
    if(is_array($b)){
        echo "Something error!";
        exit;
    }
    else if(!is_numeric($b)){
        $c   = (int)(($a + $b) * 10);
        if($c == "8" && $b[10] == false){
            echo $flag;
        }
        else{
            echo "noflag";
            exit;
        }
    }
    else{
    echo "something error!";
    }
}
else{
    echo  "something error";
}
?>
```

解题报告

浮点数精度的问题，好像这锅不应该php来背……感觉是审计里最简单的一题，后缀一个字母可过 `is_numeric` 这关，然后b比0.7稍大一点就可以加起来等于8。

POST的是 `b=.75x` ，flag是 `hctf{wochubuxiaqule_over}` ，嗯，你出不下去了。

Misc

脑洞专辑Misc。

Explorer的图库之一二三

解题报告

一共有三题，三个flag。

第一个flag，10分，用 `strings misc.jpg` 即可获得flag `hctf{2e3e3}` 。不过……文件太大用 `strings` 一会儿就会把flag刷掉，所以要不加个 `grep` ，要不还是用文本编辑器打开稳。（我会说我一开始就傻傻地用 `strings` 然后疯狂戳 `Ctrl` `C` 吗……）

第二个flag，30分，用 `binwalk -e misc.jpg` 即可得到偏移量在B256的一个.tar.gz压缩包（可以用 `file` 来检查）， `tar -xf B256` 即可获得一个txt，打开后有个flag `hctf{nizh1dao_tuzh0ngm4}` 。

第三个flag，坑了我好久，因为JPG和TAR的信息已经被榨取完了，所以估计就在 `binwalk` 跑了显示的PNG里，手动抠出个PNG后怎么都找不到flag。一开始发现PNG已经被压缩过，我以为不太可能是LSB。不过还是试了一下，没有发现明显的flag。然后用StegSolve各种乱搞，去pixiv.net找了原图xor，没结果。

最后突然灵感大发，想到png就算被压缩过也可以写LSB，而且LSB里隐写的不一定就是明文……然后重新看了一下LSB，发现一处明显的base64痕迹…… `aGN0Znsxc2JfYWFiYmJfaXpfZXp6enp6en0=` ，解密得 `hctf{1sb_aabbb_iz_ezzzzzz}`

png在下面

后来题目给了hint直接暴露是LSB，感觉这题就已经不值100分了。

explore的奇怪番外1

朝121.42.25.113:20000发包。

```
import os
import random
num = random.randint(500,1000)
print "So do you ready to get flag?"

for i in range(num):
    ret = raw_input("ready?:")
    if ret != "yes":
        exit(0)

print "so you are ready to get flag"
print "now just say ready one times"
ret = raw_input(":")

if ret == "ready":
    for name in os.listdir('.'):
        if 'flag' in name:
            fp = open(name)
            print fp.read()
            fp.close()
            break
```

解题报告

题目已经说了是socket，那我简单写一个就好了。

```
#!/usr/local/bin/python3
import socket
```

```
import time
total = 0
s = socket.socket()
s.connect(('121.42.25.113', 20000))
time.sleep(1)
s.recv(64)
res = b'ready?:'
while res == b'ready?:':
    s.send(b'yes\n')
    total += 1
    print(total)
    res = s.recv(64)
s.send(b'ready\n')
total += 1
print(total)
res = s.recv(64)
print(res)
```

呃……有个测脸黑程度的小功能，别在意。最后拿到flag为 `hctf{pwnt0ols_1s_gr3aT}` ，然而我并没用pwntools……

Pentest

全是入门教学题，第四题编码题不知道被坑在哪里。

lightless的渗透教室入门篇（一）

同时给http://115.28.78.16:13333/pentest/01/发送POST和GET请求，内容分别为——见下。

解题报告

```
curl http://115.28.78.16:13333/pentest/01/?hacker=HelloGet --data "hacker=HelloPost"
```

flag是 `hctf{PostAndGetIsSoEasy_comeon!}`

lightless的渗透教室入门篇（二）

给http://115.28.78.16:13333/pentest/02/发送请求，请求头中包含来自google.com、xff来自本地、使用iOS 99访问本页面。

解题报告

对不起这题太诡异，我已经忘了我POST的UA是什么了。但是前两个是Referer和X-Forwarded-For没问题。 `curl http://115.28.78.16:13333/pentest/02/ --include --referer "google.com" --head`

`"X-Forwarded-For: 127.0.0.1" --user-agent "抱歉我再也复现不出来了"` 加上 `--data "hint=hint"` 可以显示哪些是正确的。至于flag，我忘了……

lightless的渗透教室入门篇（三）

向http://115.28.78.16:13333/pentest/03/发送带有特定Cookie的请求。

解题报告

一开始页面上有个你不是管理员，不能看flag。，然后就想着GET一个带有'admin=true'的，结果GET之后直接响应头里就有一条 `cookiecontent: admin=1 and isLogin=true` ，按照说的再来一次就好了 `curl http://115.28.78.16:13333/pentest/03/ --include --cookie "admin=1;isLogin=true"` ，最后flag拿到是 `hctf{hao_hao_kan_zi_liao!!!}` 。假装看过资料（逃）。

Crypto

也算是教学题，被凯撒坑了。

密码学教室入门（一）

RSA加密，给出全部信息（……）和密文，解密。

```
p = 0x9a724c6747de9eadccd33f4d60ada91754b8be8c65590cafe66f69a2f4afbfd359e47ca6fd2dbde8948062dc116bc574f4313ab99b2bb6d8ae47beaa0c1ebedd
q = 0x8c1c81cc005ce3dd6d684ebb88151dc0c53b1cef8a29b1cb8121860fb57d93117bf449aac4300dc6103ac6211c6f8ae68987d99aff0dd8967a4afa00f2116873
e = 0x190a000845e9c8c2059242835432326369aaf8c7ca85e685bba968b386155a91f1f7ca1019ff23d119222e1f0dfdeb0915d2e97601ef94bf15ca6d9211e984e9038f263f4984355c397ed22d67c26da6d31acfc4d599c70cba80859bee099e5a2dc3ab23aecf58f73f44d07318f70985c623d9612efefb15bf8dab77d5d54e85
d = 0x28b95b7e3159a851cbf537e007ae49864b7dbb93fc370a5
c = 0x23091e42fa7609c73f1941b320fad6d2ff6e47be588d1623f970f1fee7abd221c9834b208f3c888902fe87ca76ec1e1363757d93c6e25c49f1c61c72b141c0b8848b54a117427d8e30eeab89694eb5f849cafecb0e5361b9b2b0e3f89e0fdbcc66a6aad4a1a4a85d828083a01a5d569b7eeb6f9151794453382b524aa52993f9
```

解题报告

这题特尴尬的就是出题人一开始把e和d搞反了，结果解密变成了加密……解密不成的我试了试加密就出来了……

很简单按套路来即可

```python
print(pow(c, d, p * q).to_bytes(24, 'big').decode())
```

密码学教室入门（二）

凯撒加密，`mlfrj{Hfjxfw_hnumjw_8x_ozxy_ktw_kzs}`。

解题报告

唯一的坑点在于数字也轮换了，而且还不是按照字母的方式轮的……flag

是`hgame{Caesar_cipher_1s_just_for_fun}`。

密码学教室入门（三）

维吉尼亚密码是最常见的分组密码。将明文对应的书名中的空格换成下划线，在并且加上hgame{}后作为flag

> ET. PESFVWI, AJGZII BU D LJSQ ISW IKV MRQTLWTOOHRY JP WLJ CCVXNMNH,
> XJTVLJNFU RR IBTQED'T DHLFMH DX MJU WVNBN. GEWOCB MX SGOIFTGG,
> SSMA WS GF CUVJTVHH FHCLR QBVHV YICW HFZ. C QIB UTLEQ CGJMST QQ
> XMF HRPQPYLRL ECB, YSEGU RJX EKEWHGV FWPWJLY CA WLJ EGIEWHGV ESE
> C WLNSF LRIJXLHZBN ZLT JU VSTO THZJBNHH FT FU. QFOGWXJ. IG KEI
> XTLXYFP DR FDERYSU QI LNT KPTWJURRRFPW EY UJH LFOFV SK ECURFZ'U
> IEYIGU ESE JLHIFP LX NO JLW HFNO; HJGCUKJ GQXRI JV ZLNMG VIFSEKMSH VKI
> HFNO HZSKQK YIG VXTSOLRL PH WLJ CCVXNMNH.

解题报告

略坑，一开始给出的题目空格和标点也计入，我以为不算，结果求公约数怎么都是1，放弃了。后来题目改了一下（上面那段是改了之后的）不计空格和标点了，我在B站浪了一整天后开始做这题。先分析分组长度，观察到一对`\w{2}\. \w{7}`类型的和的`'\w`类型，前者初步猜测是`MR. 人名`，后者估计是`'S`，留着后面可以帮助分析密钥。

首先看到三个`WLJ`、一对`CCVXNMNH`、一对`ESE`，估计都是同一个，中间距离分别是150、195、345、110，最大公约数是5，大约就是了。中间还有一对`C`和`HFNO`，`C`太短懒得计入（虽然结果不变），`HFNO`和其它互质所以估计是巧合，于是估计密钥长度5。

然后先估计一下，如果那两个`'T`和`'U`都是`'S`变过来的话，应该分别对应是密钥第一个和第二个，于是第一个字符和第二个字符应该分别是`BC`。这样的话，那个`ET.`和`FU.`都对上了`DR.`，看起来没问题。

接下去继续看 `ET. PESFVWI` 和 `FU. QFOGWXJ` ，他们之间如果按密文加密顺序，在位置上只差1。所以……呃， `E` 对着 `F` ， `T` 对着 `U` ， `P` 对着 `Q` ， `E` 对着 `F` ……这说明了什么，这说明了我词频分析白做了……也就是密钥的后一位的ASCII=前一位的ASCII+1，所以我们得到密钥是 `BCDEF` ，虽然不一定正确，我们来解一下密就知道了。

召唤JPK，我才懒得重造轮子……

> dr. manette, viewed as a hero for his imprisonment in the bastille, testifies on darnay's behalf at his trial. darnay is released, only to be arrested again later that day. a new trial begins on the following day, under new charges brought by the defarges and a third individual who is soon revealed as dr. manette. he had written an account of his imprisonment at the hands of darnay's father and hidden it in his cell; defarge found it while searching the cell during the storming of the bastille.

耶好通顺。貌似是讲巴士底狱的。

搜了一下是著名的《双城记》的，看书太少……

于是flag为 `hgame{A_Tale_of_Two_Cities}`

（这段好长……是因为边做边写的wp的原因吗？）

密码学教室入门（四）

又是RSA

```
n = 0x81cfc71c44c83faf3c5242fa81ae2e533fc945f3bef30bc13323ea4a55b3debc11301c6a9ecb8f7ef92fa169b157435af728a145497f2cdf75b3007b9732da4c47d67683f09ae1edc8f698f5ec7549593d9f1d06adafae4ad09514928bf0367a2719f7c171580318690dafc6a3d5385b3516b769f529c0a055ce25e68bc21395
e = 0x01
c = 0x6867616d657b7273615f31735f737469316c5f653473795f6e6f77217d
```

解题报告

一眼就能注意到很特殊的e=1，e=1说明什么，其关于(p-1)(q-1)的乘法逆元也是1，所以解密就是加密。

上面zz了，何止解密就是加密，一个数的一次方就是本身，所以实际上密文就是明文。

```
print(c.to_bytes(29, 'big').decode())
```

拿到flag是 `hgame{rsa_1s_sti1l_e4sy_now!}`

密码学教室番外篇

昨天看大家凯撒密码疯狂试flag，所以 `yxrdv{uxwupytip19954902180//+/%}`

解题报告

拉丁字母用JPK来跑，符号显然是不改的，数字暴力跑出十个一个一个提交过去……flag忘了。

Reverse

虽然不是二进制选手，但是仗着自己还是有一些基础知识做了一下二进制（妈呀不做二进制我就被追上了……），还好，毕竟入门不是很坑。

re？

一个jar包，Java的(dú)逆(dài)向(mǎ)题。

解题报告

上次和学长们去厦门掌握了一点apk反编译技能，用京东鬼(jd-gui)反编译出java代码，是AES加密，因为Cipher自带解密，直接复制代码，unzip解压一下可以拿到资源文件（又是教练我想打CTF.jpg），稍微改动一下，把 `cipher.init` 的第一个参数改成Cipher.DECRYPT_MODE（其实就是2），然后再去掉原来的 `cipher.doFinal` ，在最后加上 `cipher.doFinal(tFlag)` ，输出即可，代码如下。

```
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.util.Arrays;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class re
{
  public static void main(String[] args)
    throws Exception
  {
    new re().m();
  }
```

```
  private void m() throws Exception {
    InputStream is = getClass().getResourceAsStream("./ctf.jpg");
    int len = is.available();
    if (len < 20008) {
      throw new Exception("res error");
    }

    long r = 10000L;
    while (r != 0L) {
      r -= is.skip(r);
    }
    byte[] key = new byte[16];
    int l = 0;
    while (l != 16) {
      l += is.read(key, l, 16 - l);
    }

    r = 10000L;
    while (r != 0L) {
      r -= is.skip(r);
    }
    byte[] iv = new byte[16];
    l = 0;
    while (l != 16) {
      l += is.read(iv, l, 16 - l);
    }

    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    SecretKeySpec skey = new SecretKeySpec(key, "AES");
    IvParameterSpec ivSpec = new IvParameterSpec(iv);
    cipher.init(2, skey, ivSpec);
    byte[] tFlag = { 69, -101, 74, -127, -13, 110, 17, -103, 112, -111, -87, 87, 45, -110,
38, -11 };
    String de = new String(cipher.doFinal(tFlag));
    System.out.print("hgame{" + de + "}");
  }
}
```

直接编译运行即可获得flag `hgame{AES_1s_b3TteR}` 。

你看看，逆向多简单！

链接: 百度网盘 密码: YnhtYg==

解题报告

用记事本打开exe，搜索hctf，得到flag `hctf{It_ls_t0o_ea5y!}` 。

这告诉我们私密的字符串不能明文储存……

蛤，这是啥？

链接：百度网盘 密码：aGN3bg==

解题报告

给了个压缩包，里面有极像base64的一段 `nBRxIZT3mJQxgZK7gmZCC7I=` 和一个蜜汁文件。

一开始看没有后缀名想是elf可执行文件，拖到IDA里没结果，直接运行，报错。无奈之下用 `file Re50` 命令检查知道了是python 2.7 byte-compiled，于是 `python Re50` 运行测试，就是个加密用的，稍作几次测试，可以得出是5字符进行hash。

当时并不知道是base32是什么玩意儿，但是知道base64后一位对前一位的影响不是特别大（顶多改后面那一位），后一组对前一组完全没有影响，于是就自己写了个暴力脚本逐一跑每一位。

这个只是其中一步的例子，但是大致上都是这样做出来的。

```
chr() {
  printf \\$(printf '%03o' $1)
}

for i in {33..127}; do
  chr $i
  echo -n ": "
  GUESS="b"$(chr $i)
  echo $GUESS \
    | python Re50 \
    2> /dev/null
done | grep "mJQ"
```

`chr` 是ascii码转字符。下面那个for循环是枚举字符用的，最后加一个管道通给 `grep` ，筛选出可能的字符。另外题目给出的程序有个神奇的bug，就是刚好整组五的倍数个字符输入的时候会蜜汁崩溃……

可以得到flag是 `hctf{base_32!}` 。

奇怪的代码

http://ojwp3ihl4.bkt.clouddn.com/re.exe

解题报告

这题是真·逆向，用IDA看了一晚上才看出来的，这题的混淆有点强……

每次打开exe的各个函数、变量偏移量好像都不同了，我这就以自己为准了。

首先找到main函数，F5后还是可以看的，主要是那个5个参数的函数，形式如下：

```
v5 = encrypt(
  v8,
  msg[4 * v8],
  msg[4 * v8 + 1],
  msg[4 * v8 + 2],
  msg[4 * v8 + 3]
);
```

显然是加密用的，其中看上下文可以明白，v8其实是循环变量，msg是输入字符串，按照4个一组扔进去加密，进去研究一下，加密函数内部是这样的

```
int __cdecl encrypt(int offset, unsigned char m0, unsigned char m1, unsigned char m2, unsig
ned char m3)
{
  int v5; // eax@1
  int v6; // edx@1
  int v7; // edx@1
  int result; // eax@1

  c0 = shift[*(_DWORD *)(&t1[4 * t256[t17[0][offset]]] + t4[m0])];
  c1 = shift[*(_DWORD *)(&t1[4 * t256[c0]] + t4[m1])];
  c2 = shift[*(_DWORD *)(&t1[4 * t256[c1]] + t4[m2])];
  c3 = shift[*(_DWORD *)(&t1[4 * t256[c2]] + t4[m3])];
  *(&tmp1 + c0) = 1;
  v5 = t256[*(&tmp1 + enc[t4[offset]])];
  tmp2[c1] = 1;
  v6 = *(_DWORD *)&t1[4 * t4[offset] + 4];
  LOBYTE(v5) = tmp2[enc[v6]];
  dword_B33B80 = v5;
  *(&tmp + c2) = 1;
  v7 = *(_DWORD *)&t1[4 * v6 + 4];
  result = t256[*(&tmp1 + enc[v7])];
  *(&tmp1 + c3) = 1;
  LOBYTE(result) = *(&tmp1 + enc[*(_DWORD *)&t1[4 * v7 + 4]]);
  return result;
}
```

上面的已经改过变量名和变量类型。大概能看出来，重点在于c0c1c2c3这部分，来检查一下到底这么多的数组是些什么东西。

中间略去无数心酸过程。

其实这些数组应该是故意打表来混淆的，第一个shift数组，可以看出来是个65536(256*256)大小的数组，其中的值其实是这样和下标映射上的（讲道理这个规律还真不好找……位运算真是太美了） `shift[i] = i / 256 ^ i % 256` ……，而几个tx数组，其实里面的值就是下标乘以x……比如说 `t256[i] = i * 256` 。

我们简化上述代码为：

```
c0 = 0x11 * (offset + 1) ^ c0;
c1 = b0 ^ c1;
c2 = b1 ^ c2;
c3 = b2 ^ c3;
```

为什么不加上后面的呢……因为后面的已经无关痛痒了。我们稍微调试一下可以看出来，如果前几个输入的是hgame{}，hgam进去加密是可以通过第一次循环的，而hgam的加密结果恰好就和上面enc数组里的值相同，后面的所有的过程都是用来校验的，只不过是被稍微混淆过而已，所以只需要反向解密enc的值即可，我从二进制文件中搜出了enc数组的所有值，然后写了个脚本解密，如下

```
enc = (
    0x79, 0x1e, 0x7f, 0x12, 0x47, 0x3c, 0x55, 0x26,
    0x6c, 0x05, 0x71, 0x2e, 0x2d, 0x43, 0x37, 0x52,
    0x27, 0x54, 0x20, 0x49, 0x08, 0x6f, 0x30, 0x44,
    0x18, 0x47, 0x2a, 0x45, 0xe7, 0x91, 0xae, 0xd3,
)

def decrypt(offset):
    c0, c1, c2, c3 = enc[offset << 2 : (offset << 2) + 4]
    m0 = chr(c0 ^ 17 * (offset + 1))
    m1 = chr(c1 ^ c0)
    m2 = chr(c2 ^ c1)
    m3 = chr(c3 ^ c2)
    print(m0, m1, m2, m3, sep='', end='')

for i in range(8): decrypt(i)
```

运行即可得结果 `hgame{is_it_intersting_to_moov?}`

逆向做出来之后说说简单，只是做的过程实在是心累……这也许也是我没有选择二进制的原因之一吧。

Pwn

pwn step0

nc 121.42.25.113 10000 binary:http://7xn9bv.dl1.z0.glb.clouddn.com/pwn0.zip

解题报告

不想说了，很简单，0x61616161就是'aaaa'，所以输入一大堆的a，只要没有覆盖不该覆盖的（只读段啊什么的）就是答案。具体个数IDA打开在栈里数一下char开始到arg_0的长度就好了。

最后

前十三全部都是WEB+MISC嘿嘿嘿。