

XDCTF 2016 官方 WriteUp

注：本次 WriteUp 提供部分热度较高题目的解题思路，余下赛题解题思路留待决赛结束后放出

一、Misc

签到 (50)

提供一段 base64 加密字符串：

WERGTEFHe1dlbGMwbWVfd2h5czBkaWZmaWN1bHR9

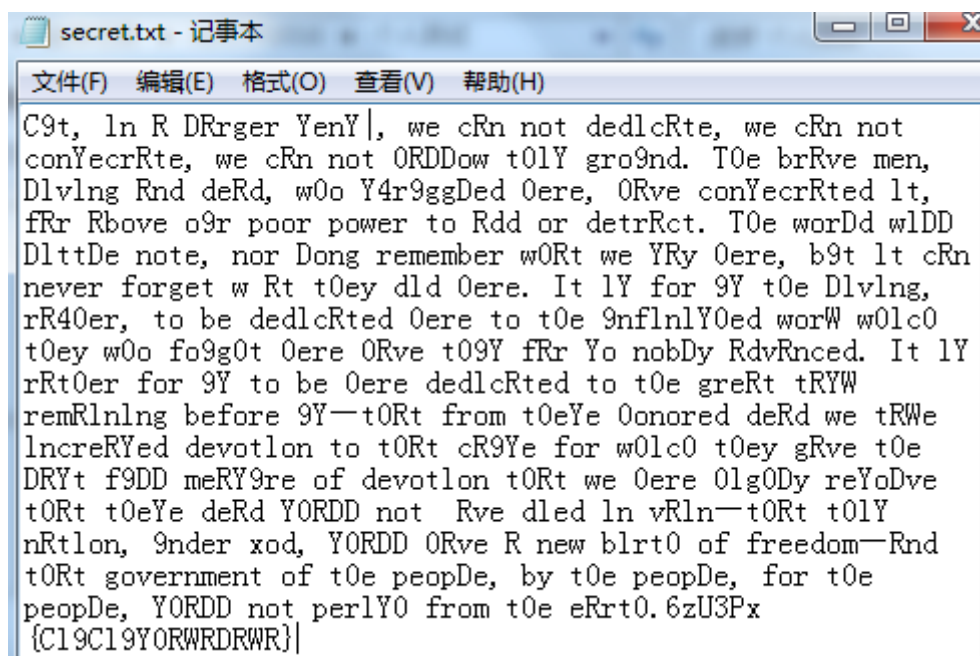
使用 python 处理即可。

```
Python 2.7.11 <v2.7.11:6d1b6a68f775, Dec 5 2015, 20:40:30> [MSC v.1500 64 bit <AMD64>] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import base64
>>> base64.b64decode("WERGTEFHe1dlbGMwbWVfd2h5czBkaWZmaWN1bHR9")
'XDFLAG{Welc0me_whys0difficult}'
>>>
```

得到 flag: XDFLAG{Welc0me_whys0difficult}

OWASP (150)

根据提供的网址，下载得到一个 pdf 文件，简单浏览和分析发现没有异样，猜测考察 pdf 隐写。接着使用 wbStego4.3open 软件分析这个 pdf 文件，因题目没有提示密码，因而尝试使用空密码，最终成功分离得到一个文本文件，文本内容如下：



C9t, ln R DRrger YenY|, we cRn not dedlcRte, we cRn not conYecrRte, we cRn not ORDDow t0lY gro9nd. TOe brRve men, DlYlng Rnd deRd, w0o Y4r9ggDed 0ere, 0Rve conYecrRted lt, fFR Rbove o9r poor power to Rdd or detrRct. TOe worDd w1DD DlttDe note, nor Dong remember w0Rt we YRy 0ere, b9t lt cRn never forget w Rt t0ey dld 0ere. It lY for 9Y t0e DlYlng, rR40er, to be dedlcRted 0ere to t0e 9nflnlY0ed worW w0lc0 t0ey w0o fo9g0t 0ere 0Rve t09Y fFR Yo nobDy RdvRnced. It lY rRt0er for 9Y to be 0ere dedlcRted to t0e greRt tRYW remRlnlng before 9Y—t0Rt from t0eYe 0onored deRd we tRWe lncreRYed devotlon to t0Rt cR9Ye for w0lc0 t0ey gRve t0e DRYt f9DD meRY9re of devotlon t0Rt we 0ere 0lg0Dy reYoDve t0Rt t0eYe deRd YORDD not Rve dld ln vRln—t0Rt t0lY nRtln, 9nder xod, YORDD 0Rve R new blrt0 of freedom—Rnd t0Rt government of t0e peopDe, by t0e peopDe, for t0e peopDe, YORDD not perly0 from t0e eRrt0.6zU3Px {C19C19Y0RWRDRWR}

留意到段落末尾的 6zU3Px{C19C19Y0RWRDRWR}，疑似 XDFLAG{} 的格式。简单还原文本文件里的英文语句，发现这是著名的《葛底斯堡公墓演讲》演讲稿，简单替换，即可得到 flag。flag 为：XDFLAG{BiuBiushakalaka}

美女配英雄 (250)

下载压缩包，发现被加密了。

flag.rar\flag3 - RAR 压缩文件, 解包大小为 12,013 字节						
名称	大小	压缩后大小	类型	修改时间	CRC32	
本地磁盘						
beauty.jpg *	11,833	11,696	JPEG 图像	2016/9/30 16...	3792C15F	
FLAG.txt *	165	160	文本文档	2016/9/28 13...	0AE98228	
key1.txt *	5	16	文本文档	2016/9/28 15...	53A610F4	
key2.txt *	5	16	文本文档	2016/9/28 14...	0F9C4B...	
key3.txt *	5	16	文本文档	2016/9/28 15...	0AAE6F9E	

留意到 key1、key2 和 key3 三个文件只有 5 个字节，再根据题面，可知压缩包的解压密码可能就藏在这三个文件里，考查的是 crc32 的碰撞。

题目提示到仅仅使用大写字母进行加密，这就节省了大家的时间，于是使用 python 脚本进行 crc32 爆破。

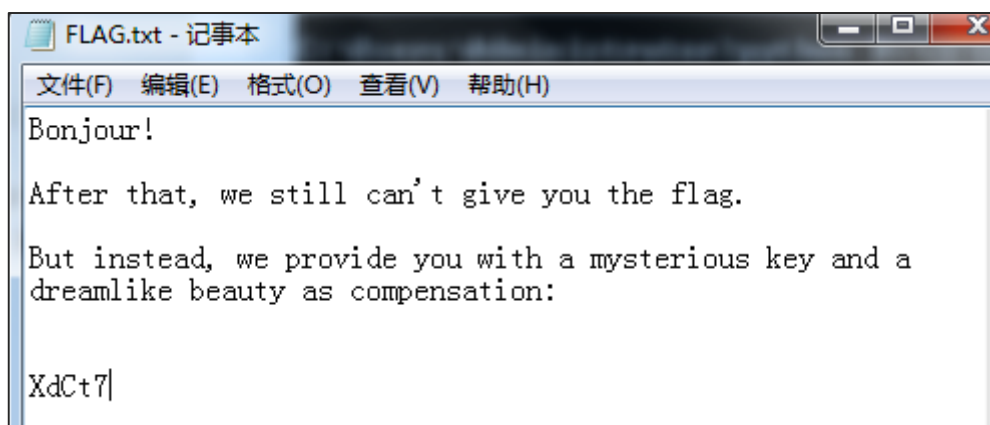
```
1  import binascii
2  import time
3
4  start = time.clock()
5  crc_num = set([0x53A610F4, 0x0F9C4BBD, 0x0AAE6F9E])
6  for i in range(65, 91):
7      for j in range(65, 91):
8          for k in range(65, 91):
9              for l in range(65, 91):
10                 for m in range(65, 91):
11                     txt = chr(i)+chr(j)+chr(k)+chr(l)+chr(m)
12                     if int(hex(binascii.crc32(txt)), 16) in crc_num:
13                         print txt
14 end = time.clock()
15 print "read: %f s" % (end - start)
```

爆破得到

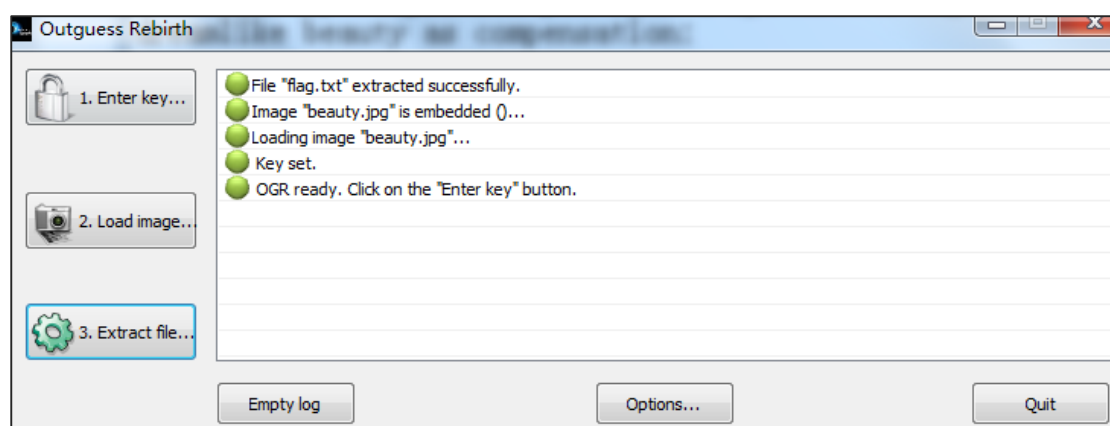
```
C:\Users\Administrator>python D:\安排任务\XDCTF2016\MISC\crc_collision\最终结果\
burp5.py
HLKID
SRTGE
WTJRS
read: 55.218593 s
```

于是拼接起来：HLKIDSRTGEWTJRS，尝试解压，发现没有成功，更改一下顺序，使用 HLKIDWTJRSSRTGE 即可破解压缩包密码。

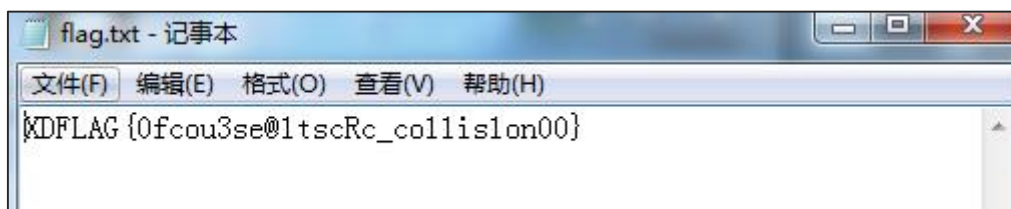
接着查看 FLAG.txt 得到



使用提供的图片，根据提示使用 outguess 进行了隐写，输入上面提供的密钥。



分离出 flag.txt 文件



得到 flag : XDFLAG{0fcou3se@1tscRc_co11is1on00}

二、Crypto

老宋的记事本 (150)

题目提供了两个字符串，并要求找到 Windows 服务器的登陆密码。

老宋的记事本

150

老宋平时注意隐私保护，一天他忘了自己Windows服务器的登陆密码，他翻开记事本，发现下面两个字符串：58220C460288D886和5422800000000000，一顿捣鼓之后，他就找到了登录密码，你知道他是怎么做到的吗？

FLAG形式：XDFLAG[登陆密码]

联想到这可能是考查 Windows 下的 Hash 生成方式。经过搜寻资料，可知提供的两个字符串正是 Windows 下 LM Hash 生成过程中的 8 字节编码字符串。接下来将这两个字符串分别作为 DES 加密密钥对魔术字符串“KGS!@#\$\$”进行加密。可使用工具实现。



DES 计算器

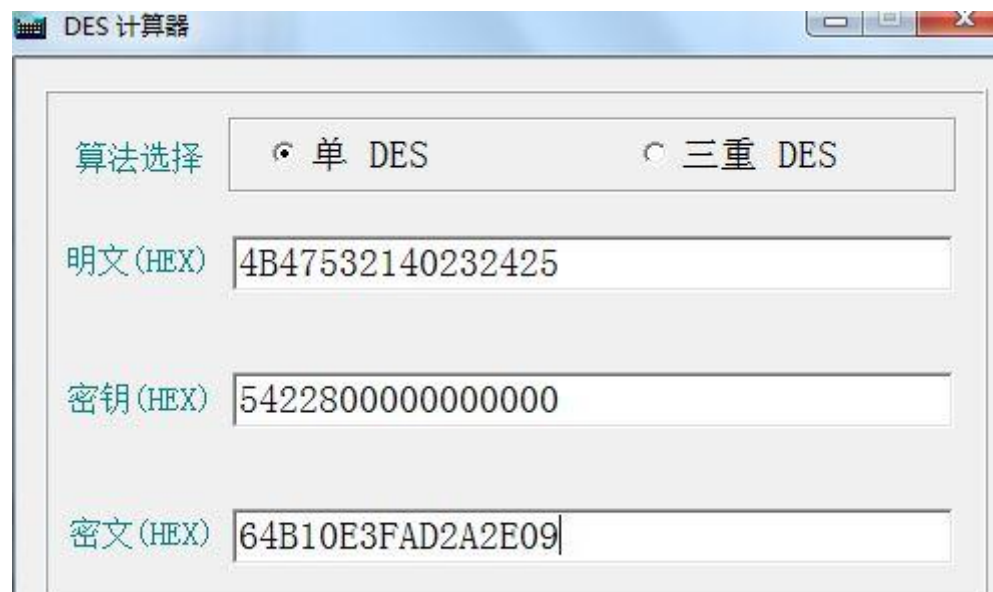
算法选择 ☒ 单 DES ☐ 三重 DES

明文 (HEX) 4B47532140232425

密钥 (HEX) 58220C460288D886

密文 (HEX) 90FC6637DCD12562

加密运算 解密运算 退出



DES 计算器

算法选择 ☒ 单 DES ☐ 三重 DES

明文 (HEX) 4B47532140232425

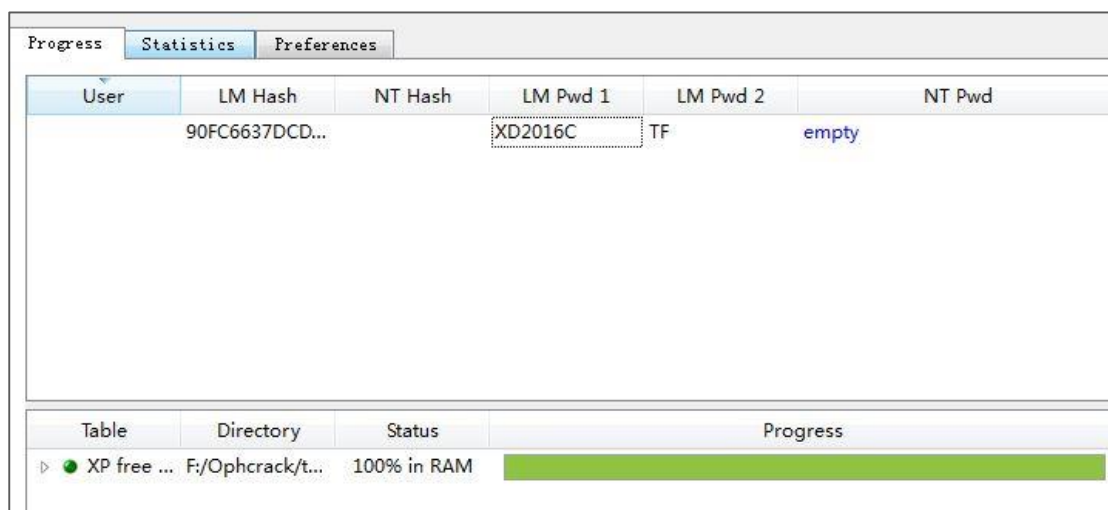
密钥 (HEX) 5422800000000000

密文 (HEX) 64B10E3FAD2A2E09

这样就得到了最终的 LM Hash 值:

90FC6637DCD1256264B10E3FAD2A2E09

将得到的 LM Hash 值传入 Ophcrack，并调用彩虹表破解即可。



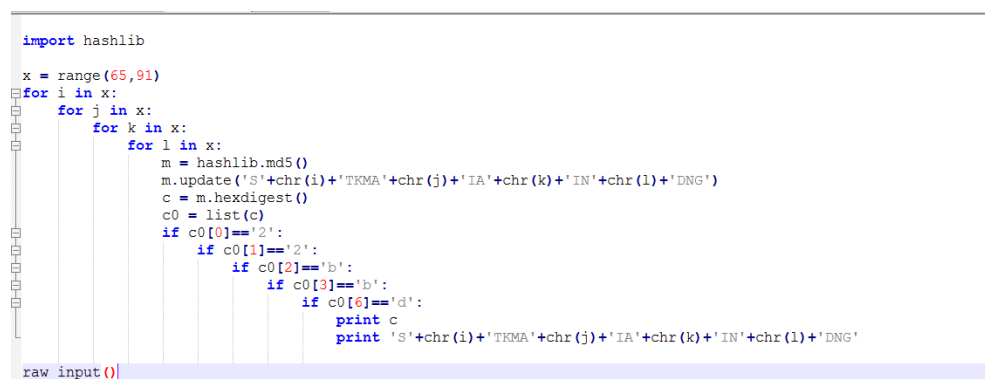
得到登陆密码: XD2016CTF

另外此题可以根据提供的字符串反向还原登陆密码，大家自行脑补。

根据 flag 提交格式，得到 flag: XDFLAG{XD2016CTF}

跨时代的刚蹭（250）

依据题目提示，是要根据提供的破损字符串和其破损 md5 值将破损字符串还原。使用脚本暴力猜解即可，提供选手的 python 爆破脚本作为参考。



爆破得到原始字符串



按照题目提示，将得到的字符串再做处理，经过测试发现是 4 栏的栅栏加密



得到 flag: XDFLAG{SMARTANDTHINKING}

心里话 (360)

下载 2.txt 文件，得到一串 01 编码

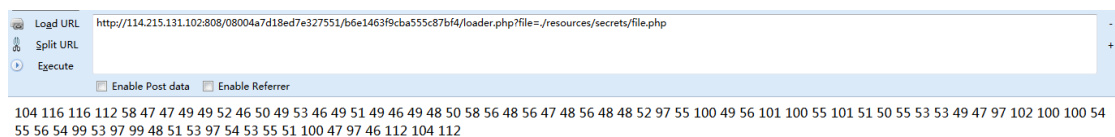
[illegible]


```

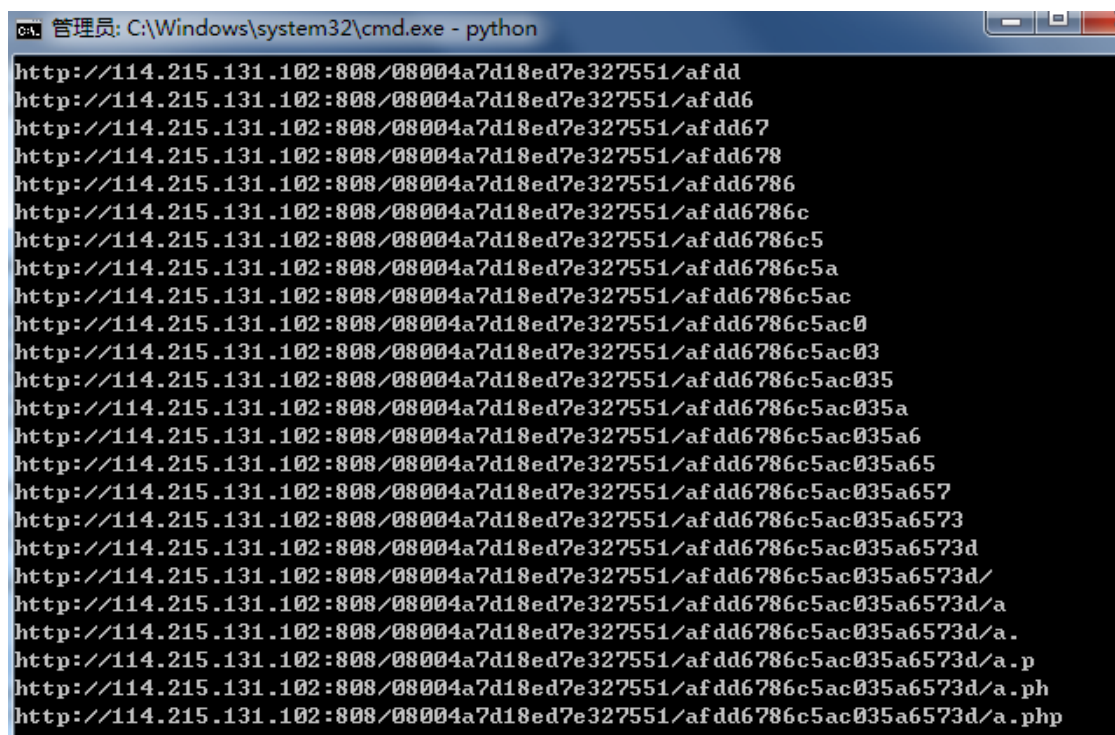
<div class="container">
  <div class="row">
    <div class="col-md-12">
      <?php
        $config_file = fopen("./resources/config/mode.config", "r");
        if (fgets($config_file) == "true") {
          $file = fopen("./resources/secrets/file.php", "r");
          echo fgets($file);
          fclose($file);
        }
        fclose($config_file);
      ?>
    </div>
  </div>
  <span style="margin-left: 750px; margin-top: 10px; font-size: 19px; line-height: 21px; height: 60px;"><h2>西安电子科技大学</h2> </span>
  <center>
    
    
    <br><br>
    
    
    <br>
  </center>
</div>
<a href="index.php">index.php source code</a>

```

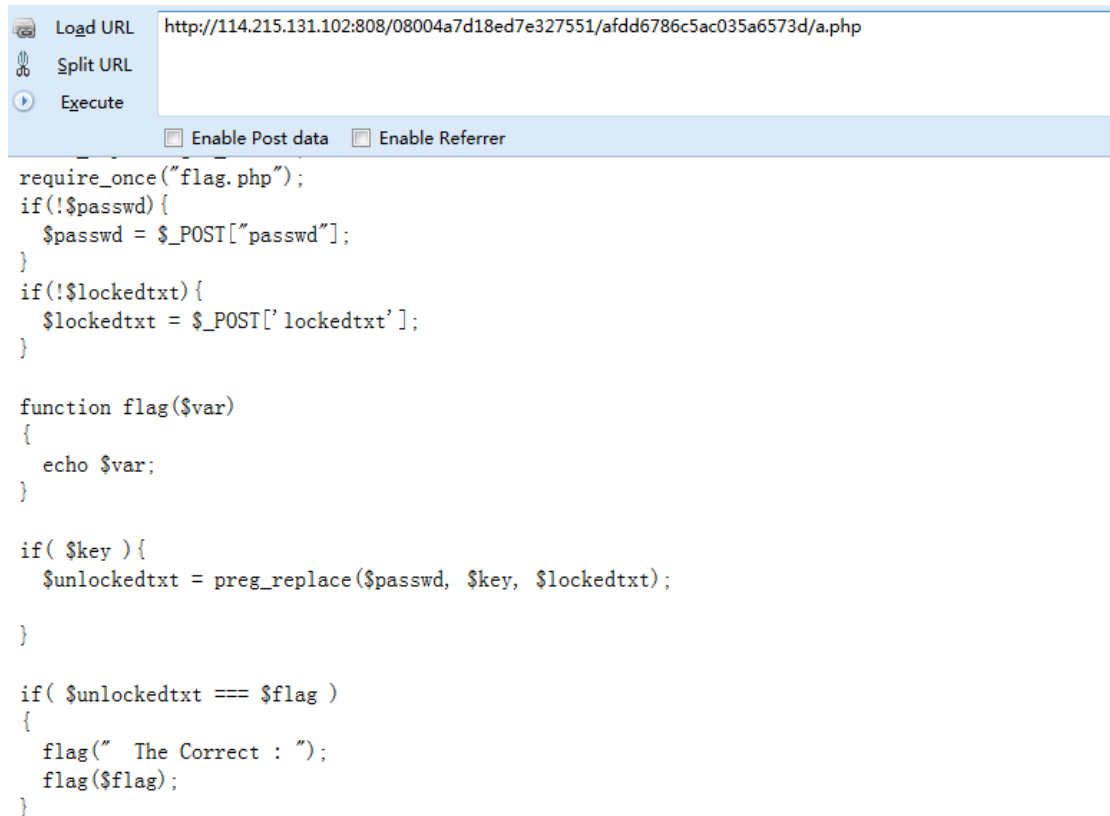
图片使用 loader.php 文件 GET 获取，于是尝试读取 resources/secrets/file.php 的内容。



读到一段 ascii 码字符串，使用脚本解码。得到一个 url 地址。



查看源码



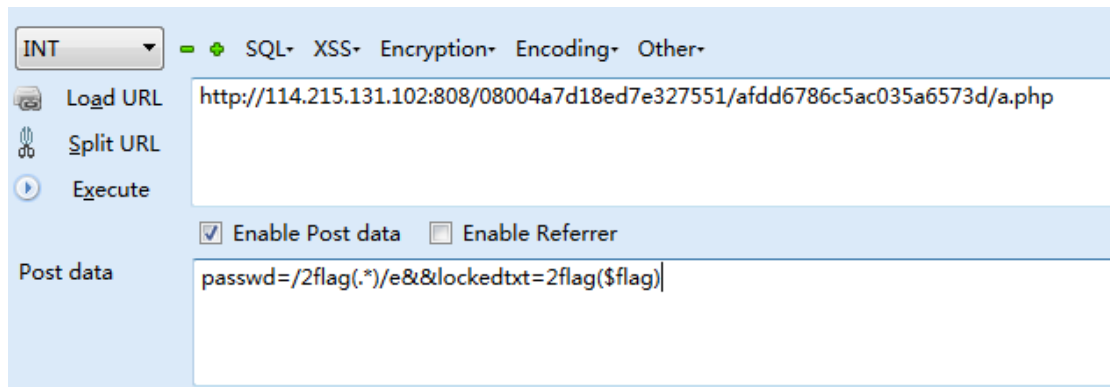
```
require_once("flag.php");
if(!$passwd){
    $passwd = $_POST["passwd"];
}
if(!$lockedtxt){
    $lockedtxt = $_POST['lockedtxt'];
}

function flag($var)
{
    echo $var;
}

if( $key ){
    $unlockedtxt = preg_replace($passwd, $key, $lockedtxt);
}

if( $unlockedtxt === $flag )
{
    flag(" The Correct : ");
    flag($flag);
}
```

考点在 preg_replace()的/e 修正符, 对于\$unlockedtxt 这个变量, 如果合理设置\$passwd 和\$lockedtxt, 就会返回\$key, 然后如果再利用上面的 flag (\$flag) 函数, 正好是返回\$flag, 这样就能返回 flag。



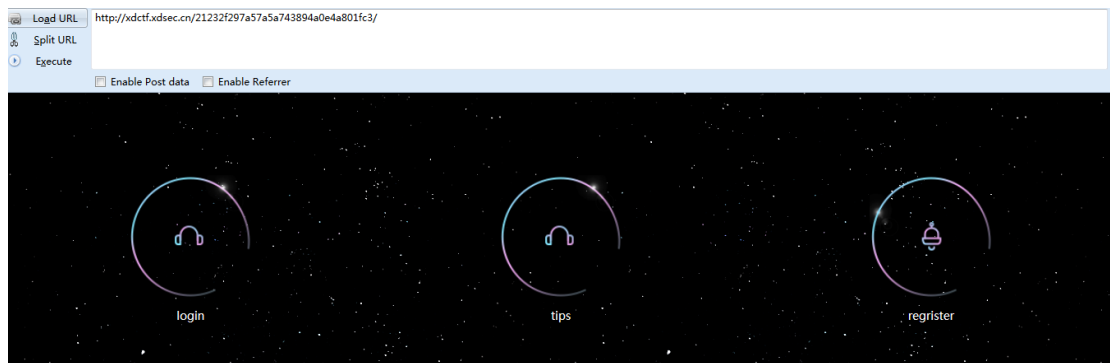
The Correct : XDFLAG{ssdfecfrtgnjdfaqswwjiklopuik}
[a.php source code](#)

同样本题也可以使用 preg_replace()的/e 的问题, 来直接使\$unlockedtxt 返回\$flag, 构造 payload 如下:

```
passwd=/(.*)/e&&lockedtxt=$flag
得到 flag: XDFLAG{ssdfecfrtgnjdfaqswwjiklopuik}
```

登陆的技巧 (150)

答题页面如下图：



尝试登陆，发现没有什么突破点，点击查看 tips

Code Snippet

```
$user_name=$_POST['user_name'];

$user_pwd=$_POST['user_pwd'];

$secret="??";

$msg=$user_name.$secret;

if ($user_name=='admin'){

    if ($cookie['user_name']==crc32($msg)){

        ?? }

    else echo"密码错误";}

else{??? }
```

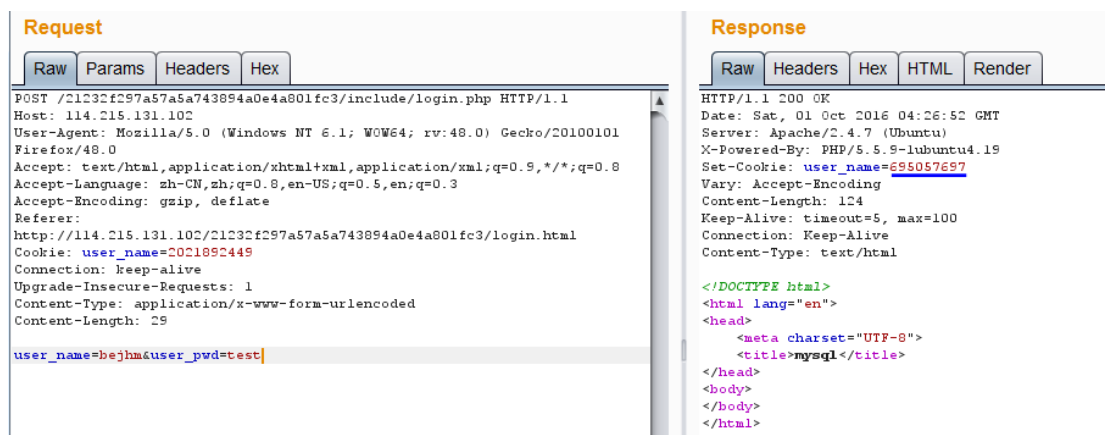
分析代码，尝试使用 admin 用户登录，并抓包。从 cookie 位置可以看到 cookie 被设置为 crc32 函数处理后的字符串。因为我们对于 \$secret 的字符串无从知晓，根据 crc32 的仿射特性，所以我们注册几个账号并抓包来异或绕过。

于是注册两个账号，分别是 12345 和 23456，密码都是 test，在登陆页面登陆并记录包里返回的 crc32 处理的字符串，分别是 2021892449，395092601

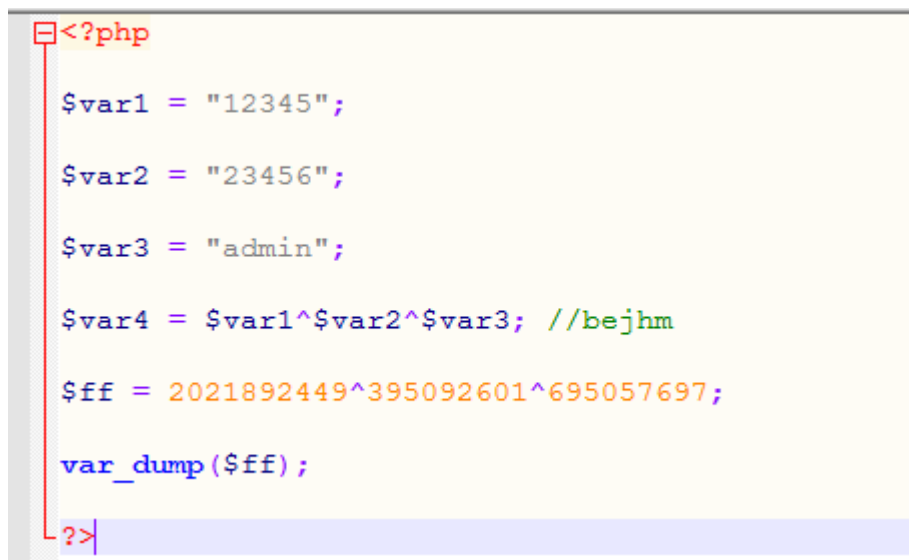
Request				Response				
Raw	Params	Headers	Hex	Raw	Headers	Hex	HTML	Render
<pre>POST /21232f297a57a5a743894a0e4a801fc3/include/login.php HTTP/1.1 Host: 114.215.131.102 User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:48.0) Gecko/20100101 Firefox/48.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3 Accept-Encoding: gzip, deflate Referer: http://114.215.131.102/21232f297a57a5a743894a0e4a801fc3/login.html Connection: keep-alive Upgrade-Insecure-Requests: 1 Content-Type: application/x-www-form-urlencoded Content-Length: 29 user_name=12345&user_pwd=test</pre>				<pre>HTTP/1.1 200 OK Date: Sat, 01 Oct 2016 04:23:15 GMT Server: Apache/2.4.7 (Ubuntu) X-Powered-By: PHP/5.5.9-1ubuntu4.19 Set-Cookie: user_name=2021892449 Vary: Accept-Encoding Content-Length: 124 Keep-Alive: timeout=5, max=100 Connection: Keep-Alive Content-Type: text/html </DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>mysql</title> </head> <body> </body> </html></pre>				



将 12345、23456 和 admin 异或，得到 bejhm，注册然后登陆抓包，得到 crc32 字符串：695057697

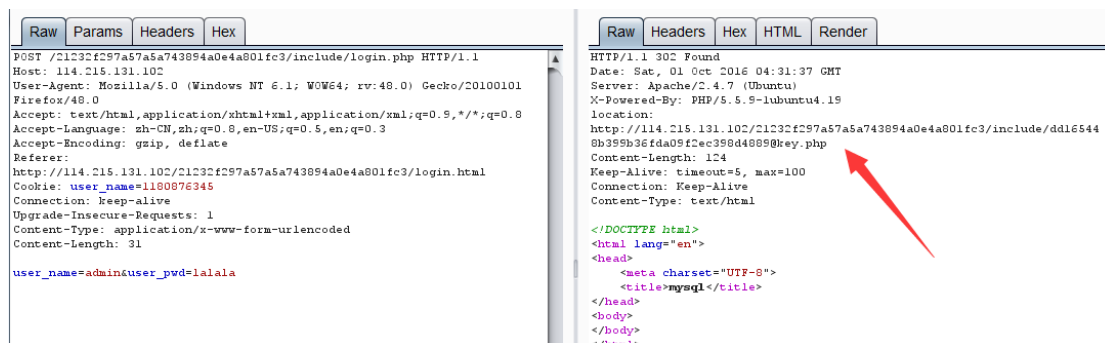


基于此，使用 php 脚本来异或处理，内容如下



得到异或后的字符串为 1180876345。这样就得到了 admin 用户的 crc32 处理后的字符串，在 burp 里处理，将 Cookie 设置为 user_name=1180876345，用户名填 admin，密码随意。就能在响应包里看到一个跳转的 url 地址：

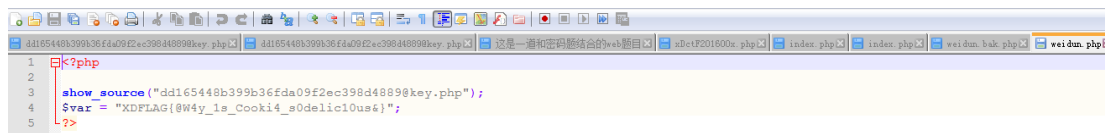
http://114.215.131.102/21232f297a57a5a743894a0e4a801fc3/include/dd165448b399b36fd a09f2ec398d4889@key.php



使用浏览器访问，发现是使用威顿加密的 php 文件

```
<?php // This file is protected by copyright law & provided under license.Welcome to xidian 2016.
$000000000=__FILE__.$000000000=urldecode('%74%68%36%73%62%65%68%71%6c%61%34%63%6f%5f%73%61%64%66%70%6e%72');$000000000=:
>
kr9NHenNHenNHe1zfukgFMaXdoyjcUImb19oUAxyb18mRtvmwJ4LT09NHr8XTzEXRjwmwJXPkr9NTzEXHenNHtILTO8XT08Xhr8XhtONTznNTzEXHr8Pkr8j
```

对应解密，得到 flag: XDFLAG{@W4y_1s_Cooki4_s0delic10us&}



迷途 (260)

题面如下：

迷途
260

我们都在不断赶路，忘记了出路

<http://114.215.131.102/297aae72cc4d0d068f46a9158469e34d/>

TIPS: The PATH to glory is always rugged.

进入答题页面是一个登陆表单，尝试常规注入等方式都没有效果，使用 test 作为用户名登陆，可以发现页面返回 administrator，但是没有什么用。根据提示猜测本题考查 XPATH 注入。

根据 XPATH 注入的特点，fuzz 多次后，最终在账号处测试得到下面的 payload:

```
1']//*[s[
```

登陆得到 flag

114.215.131.102/297aae72cc4d0d068f46a9158469e34d/

站长工具 - 站长之家 PHP Sessions HackFu 2016 Writeu

nosec
foryou
lazy
xdctf_1
administrator
xdctf_3
crazy
xdctf_5
xpath

XDFLAG{Y0u_Cr@cke9_xPa7h66}

得到 flag: XDFLAG{Y0u_Cr@cke9_xPa7h66}

四、Android Mobile

埃及神战 (100)

下载该文件，发现是 dex 格式，然后根据反编译出来的代码中的种种提示分析文件，发现丢失了一个函数，那么简单推出该函数的偏移地址，修改二进制值，如下图：

• struct uleb128 direct_methods_size	0x5
• ubyte val	5
• struct uleb128 virtual_methods_size	0x0
• struct encoded_method_list direct_methods	5 methods
• int s	5
• int i	5
• int methodid	11
• struct encoded_method method[0]	public constructor void com.nbrc.dessert.LostFunc.<init>()
• struct encoded_method method[1]	public static java.lang.String com.nbrc.dessert.LostFunc.welcometoxdctfA(ja
• struct encoded_method method[2]	public static java.lang.String com.nbrc.dessert.LostFunc.welcometoxdctfB(ja
• struct encoded_method method[3]	public static java.lang.String com.nbrc.dessert.LostFunc.welcometoxdctfC(ja
• struct encoded_method method[4]	public static java.lang.String com.nbrc.dessert.LostFunc.welcometoxdctfD()
• int p	10
• struct uleb128 method_idx_diff	0x1
• struct uleb128 access_flags	(0x9) ACC_PUBLIC ACC_STATIC
• struct uleb128 code_off	0x8CC
• ubyte val[0]	204
• ubyte val[1]	17

然后重新反编译该 dex 文件，就可以看到该函数。该函数的加密很简单，反推就可以得到钥匙。反编译出的加密算法如下：

```

public static String welcometoxdctfD() {
    int i;
    int i2 = 0;
    String str = "75b49434e484471493d5c6e3c774c3d7c3f405f427";
    int length = str.length();
    byte[] bArr = new byte[((length / 2) + 1)];
    for (i = 0; i < length; i += 2) {
        bArr[i / 2] = (byte) ((Character.digit(str.charAt(i), 16) << 4) + Character.digit(str.charAt(i + 1), 16));
    }
    byte[] bArr2 = new byte[(str.length() / 2)];
    i = 0;
    int i3 = 0;
    while (i3 < bArr.length - 1) {
        bArr2[i] = (byte) (bArr[i3] ^ 240);
        i3++;
        i++;
    }
    i3 = bArr2.length;
    String str2 = new String();
    while (i2 < i3) {
        str2 = new StringBuilder(String.valueOf(new StringBuilder(String.valueOf(str2)).append(Integer.toHexString((bArr2[i2]
        i2++;
    }
    return str2;
}

```

最终的 flag: XDFLAG{N6Rc1sH3r3@_@}

穿针引线 (150)

打开反编译的源码后，发现调用 `checkPass(String str)` 进行密码校验，在 `so` 中找到 `checkPass` 方法，发现该方法调用 `java` 层的 `Functions` 类 `func1` 函数对 "wearethebest" 字符串进行取 MD5 值，调用 `func2`，对密码进行比较。

```

v5 = (*v3)->NewStringUTF(v4, "wearethebest");
v6 = ((int (__fastcall *)(JNIEnv *, const char *))(*v3)->FindClass)(v3, "com/xidian/crackme009/Functions");
v7 = v6;
v8 = ((int (__fastcall *)(JNIEnv *, int, const char *, const char *))(*v3)->GetStaticMethodID)(
    v3,
    v6,
    "func1",
    "(Ljava/lang/String;)Ljava/lang/String;");
((void (__fastcall *)(JNIEnv *, int, int, jstring))(*v3)->CallStaticObjectMethod)(v3, v7, v8, v5);
v9 = ((int (__fastcall *)(JNIEnv *, int, const char *, const char *))(*v3)->GetStaticMethodID)(
    v3,
    v7,
    "func2",
    "(Ljava/lang/String;Ljava/lang/String;)I");
if ( ((int (__fastcall *)(JNIEnv *, int, int, int))(*v3)->CallStaticIntMethod)(v3, v7, v9, v4) )
{
    v10 = v3;
    v11 = "a";
}
else
{
    v10 = v3;
    v11 = "b";
}

```

对 `func2` 进行分析，发现将 `flag` 值的每个字符下标值往后循环移 5 位再取字符与 MD5 进行比较，故将 md5 每个字符循环左移 5 位可得到 `flag` 值。

根据 `flag` 形式，得到最终 `flag` 值: XDFLAG{a4373d6fdb0c73917fde04123a6e474b}

```

public static int func2(String data1, String data2) {
    StringBuilder mString = new StringBuilder();
    for (int i = 0; i < data1.length(); i++) {
        mString.append(hexString.charAt((hexString.indexOf(data1.charAt(i)) + 5) % 16));
    }
    if (mString.toString().equals(data2)) {
        return 1;
    }
    return 0;
}

```

此地无银六百两（330）

此题的开发方式采用 NDK 开发，该应用就是一个输入密码的程序。其中一个 flag 隐藏在 so 文件中。另外一个隐藏在背景图片中，因此得名“此地无银六百两”。

题目提示本题有两个 flag。其中一个 flag 隐藏在 so 文件中。使用 so 文件对字符串进行处理和对输入的字符串进行判断。该文件编辑如下：

```

1 #include <jni.h>
2 #include <com_example_hellondk02_GetString.h>
3 #include <malloc.h>
4 #include <string.h>
5 #include <stdio.h>
6 #include <stdlib.h>
7
8 #define LOG_TAG    "HelloJni"
9
10 JNIEXPORT jboolean JNICALL Java_com_example_hellondk02_GetString_encrypt
11 (JNIEnv *env, jclass, jstring string01){
12
13 //  jstring string04="bcNOUV8vwyInSTpqrsHk1m2EFtuJKLzMS67CDefghijAB09XYZadoPQRGx134W";
14 //  jstring string03="Welc0meT0XiAn"; //5ucce55fulJ0b
15
16     jstring string04=env->NewStringUTF("bcNOUV8vwyInSTpqrsHk1m2EFtuJKLzMS67CDefghijAB09XYZadoPQR
17     jstring string03=env->NewStringUTF("Welc0meT0XDCTF"); //5ucces5fulW0RK
18
19     jstring string02=string01;
20     const char* tmp = env->GetStringUTFChars(string03, NULL);
21     const char* tmp02=env->GetStringUTFChars(string04, NULL);
22     const char* tmp03=env->GetStringUTFChars(string02, NULL);
23     int len = strlen(tmp) + 1;

```

string 03 中定义了一段字符串“Welc0meT0XDCTF”。紧接着，后面会对这个字符串进行一系列变换。将其转化为注释中的字符串“5ucces5fulWORK”。为此，专门定义了一个字符串 string 04 作为查询表。

然后就开始进行替换

```

int a[14]={32,25,3,-2,41,12,38,31,34,11,71,34,67,15};
strcpy(mac, tmp);
strcpy(mac02,tmp02);
strcpy(mac03,tmp03);
for (int var = 0; var < strlen(tmp); ++var) {
    if (var%2==0) {
        mac[var]=mac02[a[var]-var];
    } else {
        mac[var]=mac02[a[var]+var];
    }
}
for (int i = 0; i < 14; i++) {
    if(mac03[i]!=mac[i]) {

        return false;
    }
}

```


并且按照逐一对输入的字符和变换后的字符进行检测。返回 True 或则 False。

另一个 flag 隐藏在背景图片里，使用 stegsolve 打开图片，切换像素通道，存在 LSB 隐写，发现二维码



反色处理后扫描得到下面的字符串:

```
&#65;&#49;&#121;&#48;&#117;&#33;&#78;&#105;&#72;&#101;&#110;&#66;&#117;&#67;&#117;&#48;
```

使用 unicode 编码，解码得到这部分 flag: A1y0u!NiHenBuCu0

最终组合得到本题 flag: XDFLAG{A1y0u!NiHenBuCu05ucces5fulW0RK}

五、Reverse

伏羲剑 (150)

先动态 od 或者工具解除程序所加壳，否则很难分析。很头疼然后绕过一些反调试和错误坑。如果静态 ida 查看则不需要解除，只要分清楚程序流程和逻辑。

该程序既是 GUI 界面程序，又是命令行输入程序。正常参数必须通过命令行传入，不传入程序也正常运行，但是会进入错误逻辑运算。

比如界面点击按钮输入位置，是错误算法。

真正在程序点击右上角关闭程序之后 destroy 函数响应之后，才有正常加密算法逻辑开始处理。

算法是顺序是 base64 编码，再接着编码后，顺序按照字节异或 1, 2, 3, 4, 5, 6, 7, 8。

然后异或的结果进入 des 算法，但又不是完全是，此处的密钥是 13579 字符串，而长度是八字节，默认实现 des 都是至少 128 比特长度，得注意区别。最后和密文比较，逆向解出即可。

最终得到 flag: XDFLAG{this_1s_a_small_di8h_for_y0u_today!}

黄金周福利 (220)

仍然是先动态 od 或者工具解除程序所加得壳，然后绕过一些反调试和错误坑，如果静态 ida 查看则不需要解除，就要分清楚程序流程和逻辑。

此题算法并不难，命令行接收两个参数，一个账号一个密码，算法是先分开对其编码，然后异或几个不同的值。很多人估计能拿到用户名和密码，到这步不是答案，但距离答案近了一步，后面需要。

然后在密码账号获取的情况下，别被逻辑误导。注意到有个释放文件函数，写入一个文件到用户临时目录。这个文件看头部，像压缩文件，修复头部。得到一个压缩文件，但需要密码，输入刚才用户名和密码拼接字符串，解开文件。发现是一个，十六进制编辑器打开，貌似无规律。实际上文件是个二进制流倒写。倒着看发现文件头 ID，推测 mp3 音频文件，修复好。

最后倒着二进制流保存，播放即可。音频会告诉你答案。

最终 flag 值: XDFLAG{A1B2C3SUCCESD4F5G6}

致敬经典 (240)

题目内容如下:

致敬经典

240

还记得那些令你印象深刻的红极一时的手机品牌么。

HTC? 诺基亚? 还是。。

PS:程序包含部分保护代码，建议在虚拟机下运行

程序分为三个部分，第一部分要求输入用户名，然后解密第二部分的代码执行

第二部分的会加密第一部分的代码然后检验用户名的 base64 是否为 fmgzMSFPX20wVDAq，是的话解密第三部分的代码

第三部分加密第二部分的代码，然后对输入的密码进行 base64 加密和 YnkzYnkzX20wVDAq 比较

当程序运行中检测到输入不正确时会直接关机。

最终 flag 值: XDFLAG{~h31!O_m0T0*by3by3_m0T0*}