ctf writeup之程序员密码

转载

weixin_30347009 于 2018-04-09 23:32:00 发布 154 收藏
原文链接：http://www.cnblogs.com/cc11001100/p/8764153.html

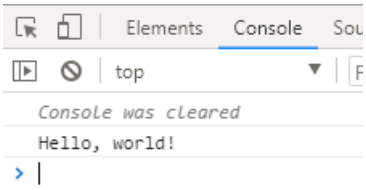
起因

在v2ex上看到有人发了一篇帖子，说做了一个程序员小游戏，遂试玩了一下。

游戏的地址在这里： http://www.bettertomissthantomeet.com/pages/level.html

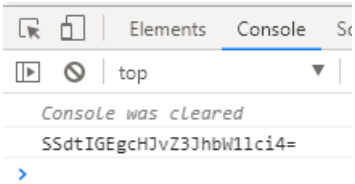
第零关

控制台打印的字符“Hello, world!”，复制粘贴通过。



第一关

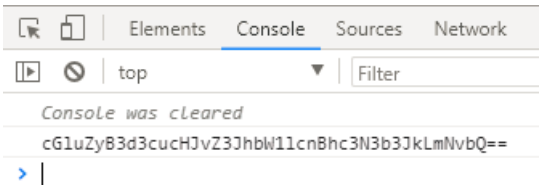
控制台又打印了一串字符：



看着像base64，拿去解码，得到“I'm a programmer.”，复制粘贴通过。

第二关

为毛又是控制台打印...



拿去base64解码，得到“ping www.programmerpassword.com”，遂ping了一下：

将ip地址“47.104.152.148”复制粘贴通过。

感觉这个设计得不太好，因为一般ping的话是为了看ttl延时，少数情况是为了解析到ip地址，不过好歹也能够联想到。

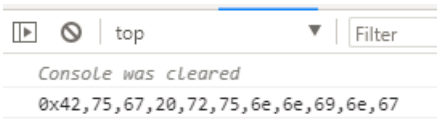
第三关

这次控制台没有输出了，并且页面上也没有啥提示，于是ctrl+shift+c选了下提交按钮，然后看到了

这是一个透明元素，所以页面上看不到，复制“123456”粘贴没通过....需要完整的复制“密码是123456”粘贴提交才能通过，作者很顽皮啊。

第四关

这次还是老的套路，控制台打印了一串十六进制数：



拿去解码（关键词十六进制转字符串），将“4275672072756e6e696e67”转为字符串“Bug running”，复制粘贴通过。

第五关

选一下这个井号：

```
▼<div class="container center" id="info-box">
    <span style="color:rgb(47, 69, 132)!important; font-size:8em;">#</span> == $0
  </div>
```

看到有一个样式是rgb(r,g,b)这种格式指定的，然后页面上又是一个大大的井号，遂将“rgb(47, 69, 132)”拿去rgb转hex，得到“#2F4584”，但是很悲剧的事情是作者的判断正确的方法是将我们提交得的答案md5和正确答案的md5进行比对，所以就没办法兼容大小写这种情况，这个是我想吐槽的第一点。然后我想吐槽的第二点是没有办法兼容大小写就算了标准答案还是非主流的小写…hex的A-F很少有用小写的，所以就导致大多数人都会被坑一下…

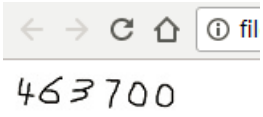
提交“#2f4584”通过。

第六关

控制台打印了一堆东西，很明确表示这是base64。然后观察base64的开头和img的src base64很像，遂拿去放到一个html中：

```
<!DOCTYPE html>
<html>
<head>
 <title></title>
</head>
<body>
<img src="https://img-blog.csdnimg.cn/20220106213821183l3.png">

</body>
</html>
```

打开这个html：

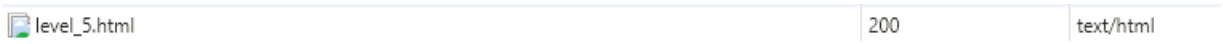


提交“463700”通过。

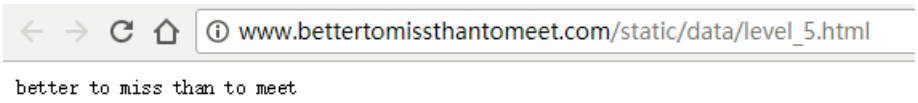
第七关

有个挂掉的图片：

然后切换到network看一下：



发现这明明是个doc类型的好吧，然后手动打开：



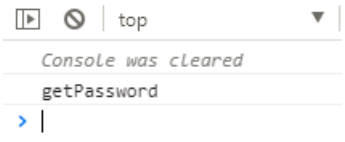
提交“better to miss than to meet”通过。

第八关

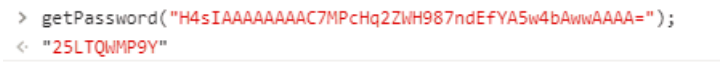
页面上显示了一行字：

```
H4sIAAAAAAAAC7MPcHq2ZWH987ndEfYA5w4bAwwAAAA=
```

控制台也有输出：



遂在控制台使用getPassword()方法解密（别问我怎么知道的，因为我读了这个js...）：



提交“25LTQWMP9Y”通过。

第九关

这关是设计得还比较好的，因为所使用的技术的特点体现的非常好。

页面上就这么一张图片：

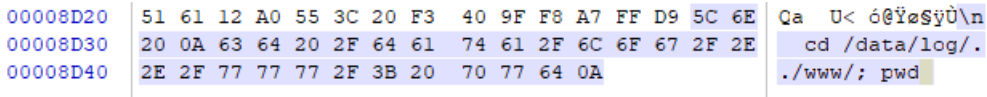
```
<?php
try {
  $dbh = new PDO('odbc:SAMPLE', 'db2inst1', 'ibmdb2',
      array(PDO::ATTR_PERSISTENT => true));
  echo "Connected\n";
} catch (Exception $e) {
  die("Unable to connect: " . $e->getMessage());
}

try {
  $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  $dbh->beginTransaction();
  $dbh->exec("insert into staff (id, first, last) values (23, 'Joe', 'Bloggs')");
  $dbh->exec("insert into salarychange (id, amount, changedate)
      values (23, 50000, NOW())");
  $dbh->commit();

} catch (Exception $e) {
  $dbh->rollBack();
  echo "Failed: " . $e->getMessage();
}
?>
```

看了半天，以为玄机在图片上的代码里，还去看了PDO的相关资料....代码成功的吸引了猿类的注意力，如果是一张风景图之类的恐怕就能够秒想到图片隐写术。

右键把图片保存到本地，然后使用winhex打开，搜索FFD9找jpg图片的结尾。

jpg文件有一个特点就是以FFD9为结束符，结束符之后的内容就直接无视，利用这个特性可以往尾部藏一些东西。

下图中高亮的部分都是会被无视的内容：



这是两行命令，最有可能的答案就是这两行命令的输出了，来分析一下可能会输出什么：

```
cd /data/log/../www/
pwd
```

log会被..约掉，所以是cd /data/www/然后用pwd输出当前路径，Linux下pwd是不会带最后的斜线的，所以最终的输出就是“/data/www”，提交通过。

不足

核心的判断逻辑在这个js中：http://www.bettertomissthantomeet.com/static/js/app/level.js，将判断放在前端还很齐齐整整，人家只要阅读一遍这个js就大概了解了，这样大大降低了难度不太好。

不过最后的隐写术真的很6，成功误导我在PDO的方向上越走越远，花了一大会儿时间才想到….

.

转载于:https://www.cnblogs.com/cc11001100/p/8764153.html