PHP花式绕过大全

原创

置顶 fly夏天 于 2019-10-18 10:36:29 发布 6825 收藏 62
分类专栏： ctf 文章标签： PHP 函数绕过 ctf
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/xiayu729100940/article/details/102619255
版权


ctf专栏收录该内容
17 篇文章 1 订阅
订阅专栏

做了这么长的时间的ctf，现在总结一下自己做过的题，记录一下各种可能会存在绕过的php函数，持续跟新。

各位大佬可以一起交流♂交流。

1.preg_replace （$pattern , $replacement , $subject ）

其中 $pattern为正则表达式

$replacement为替换字符串

$subject 为要搜索替换的目标字符串或字符串数组

这个函数存在一些奇异的地方，正则表达式$pattern以/e结尾时$replacement的值会被作为php函数执行。

例如执行 preg_replace （'/test/e' , "phpinfo();" , "test" ）

“test”会被替换为phpinfo();并执行。

2.MD5加密绕过

2.1 MD5弱比较绕过

php中有一个提供MD5加密的函数md5()通常被用来进行密码验证之类的功能。

在ctf中常见如下的验证方式：

if(a == b && md5(a) == md5(b))

方法一：

这儿md5(a)/md5(b)两数如果满足科学计数法的形式的话，php会将其当作科学计数法所得的数字来进行比较。
例如：

```
md5（QNKCDZO）
0e830400451993494058024219903391
```

可以看见QNKCDZO的md5值是0e开头满足科学计数法的表示形式，而0e的值始终为0

因此，只要字符串经md5后满足科学计数法的0e开头，他们在==比较时就会被认定为相等。（只对==比较有效，不适用于===）

以下给出一些满足该要求的md5数

QNKCDZO
0e830400451993494058024219903391
s878926199a
0e545993274517709034328855841020
s155964671a
0e342768416822451524974117254469
s214587387a
0e848240448830537924465865611904
s214587387a
0e848240448830537924465865611904
s878926199a
0e545993274517709034328855841020
s1091221200a
0e940624217856561557816327384675

方法二：

md5()函数无法处理数组，如果传入的为数组，会返回NULL，所以两个数组经过加密后得到的都是NULL,也就是相等的。

也就是说a[]=1,2 b[]=2,3 该验证依然可以绕过。

2.2 MD5强碰撞

```
if((string)$_POST['param1']!==(string)$_POST['param2'] && md5($_POST['param1'])===md5($_POST['param2'])){
        die("success!);
}
```

Param1=%4d%c9%68%ff%0e%e3%5c%20%95%72%d4%77%7b%72%15%87%d3%6f%a7%b2%1b%dc%56
Param2=%4d%c9%68%ff%0e%e3%5c%20%95%72%d4%77%7b%72%15%87%d3%6f%a7%b2%1b%dc%56

2.3 双MD5碰撞绕过

```
$a != $b && md5($a) == md5(md5($b)
```

一些例子

```
MD5大全：

CbDLytmyGm2xQyaLNhWn

md5(CbDLytmyGm2xQyaLNhWn) => 0ec20b7c66cafbcc7d8e8481f0653d18

md5(md5(CbDLytmyGm2xQyaLNhWn)) => 0e3a5f2a80db371d4610b8f940d296af

770hQgrBOjrcqftrlaZk

md5(770hQgrBOjrcqftrlaZk) => 0e689b4f703bdc753be7e27b45cb3625

md5(md5(770hQgrBOjrcqftrlaZk)) => 0e2756da68ef740fd8f5a5c26cc45064

7r4lGXCH2Ksu2JNT3BYM

md5(7r4lGXCH2Ksu2JNT3BYM) => 0e269ab12da27d79a6626d91f34ae849

md5(md5(7r4lGXCH2Ksu2JNT3BYM)) => 0e48d320b2a97ab295f5c4694759889f
```

碰撞脚本

```
# -*- coding: utf-8 -*-
import multiprocessing
import hashlib
import random
import string
import sys
CHARS = string.letters + string.digits
def cmp_md5(substr, stop_event, str_len,. start=0, size=20):
    global CHARS
    while not stop_event.is_set():
        rnds = ''.join(random.choice(CHARS) for _ in range(size))
        md5 = hashlib.md5(rnds)
        value = md5.hexdigest()
        if value[start: start+str_len] == substr:
            print rnds
            stop_event.set()
            '''
            #碰撞双md5
            md5 = hashlib.md5(value)
            if md5.hexdigest()[start: start+str_len] == substr:
             print rnds+ "=>" + value+"=>"+ md5.hexdigest()  + "\n"
                stop_event.set()
            '''

if __name__ == '__main__':
    substr = sys.argv[1].strip()
    start_pos = int(sys.argv[2]) if len(sys.argv) > 1 else 0
    str_len = len(substr)
    cpus = multiprocessing.cpu_count()
    stop_event = multiprocessing.Event()
    processes = [multiprocessing.Process(target=cmp_md5, args=(substr,
                                        stop_event, str_len, start_pos))
                 for i in range(cpus)]
    for p in processes:
        p.start()
    for p in processes:
        p.join()
```

上面脚本注释部分是双MD5碰撞，取消注释然后注释掉16行即可。

使用方法：python md5Crack.py "你要碰撞的字符串" 字符串的起始位置

例如：python md5Crack.py “0e" 0

将产生MD5值为0e开头的字符串。

注：这个脚本我本地尝试好像有问题，具体问题有时间研究一下再补上。

3.MD4 绕过（来自强网杯2020）

```
$_GET["hash1"] != hash("md4", $_GET["hash1"])
```

例子 0e251288019

md4计算后为0e87495616364196127106940433240 9

参考网上的例子改了一个简易的python爆破脚本，就是效率真的低到离谱

y1ng神还有高级脚本，感兴趣的可以自己去看看2020第四届“强网杯”全国网络安全挑战赛初赛Writeup – 颖奇L'Amore

没研究透的我就不贴了。

```
import hashlib
for i in range(0,10**40):
    i='0e'+str(i)
    md4=hashlib.new('md4', i.encode()).hexdigest()
    if md4[:2]=='0e' and md4[2:].isdigit():
        print('num:{},md4：{} '.format(i,md4))
        break
#需要在python3环境下运行，不然无法执行
```

4.strip_tags()

这个函数用于去除字符串中的 HTML 标签，正常来说确实没啥问题。

但是却在ctf中一些绕过中存在。（出自roarctf2019 simple_upload）

thinkphp原生代码中的upload（）上传功能，会对上传的文件名执行该函数。

利用该方法可以进行前端绕过。

当php进行上传文件后缀验证，过滤.php时。

一般情况下诸如 1.php是无法上传上去的，但是此处可以通过 构造文件名为 1.<a>php绕过该验证。

文件上传后不满足.php绕过过滤，但是在tp的原生上传函数被调用是<a>被去除，文件就会被以php格式上传。

注：此处的html可以为任意html标签

5.assert（）

这个函数是php的断言函数，用来判断一个表达式是否成立。返回true or false。

注：assert执行的字符串包含的php语句如果有多条，只会执行第一条。

该表达式会被当做php函数来执行，相当于eval（）

在ctf中该函数也有一定的可利用性，如xctf中的一题

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

$file为用户传入的字符串，这里构造file=**')**.**phpinfo**()**;//**

此时语句变成了assert（"strpos（"）.phpinfo();// ===false") or die (" ")

相当于 assert（"strpos（"）.phpinfo();")

phpinfo()就被注入进去并执行了。

注：这里用到了php的一个特性，那就是 .连接符 ，

读者可以自行尝试assert("ehco(123);phpinfo();")和assert("ehco(123).phpinfo();")运行起来的区别。

6.反序列化相关知识点

1.__toString()

触发条件时对象被当作字符串调用时就会执行。

需要指出的是在 PHP 5.2.0 之前，__toString() 方法只有在直接使用于 echo 或 print 时才能生效。PHP 5.2.0 之后，则可以在任何字符串环境生效（例如通过 printf()，使用 %s 修饰符），但不能用于非字符串环境（如使用 %d 修饰符）。自 PHP 5.2.0 起，如果将一个未定义 __toString() 方法的对象转换为字符串，会产生 **E_RECOVERABLE_ERROR** 级别的错误。

比如stristr函数。（来自强网杯的怨念）

2.__wakeup绕过

该函数会在反序列化时执行。当成员属性数目大于实际数目时可绕过wakeup方法(CVE-2016-7124)

例如：一个对象序列化结果如下

O:6:"jungle":1:{s:4:"name";s:1:"1";}

如果想要绕过类中的__wakeup函数，只需要将jungle数量对应的1改成2既可。

注：这里的字符串数字比如O:6，写成O:+6也是可以，利用这个可以绕过一些过滤。

3.S和s的区别

依然已上文的序列化字符串举例

O:6:"jungle":1:{s:4:"name";s:1:"1";}

这里的s表示的是属性的内容是字符串，因为这里的变量属性是public，如果是protected，则需要在属性名前加chr(0)*chr(0)

也就是s:7:"chr(0)*chr(0)name",这个是无法打出来的。

通常情况下我们可以这样表示：s:7:"%00*%00name"，用%00来代替这个chr（0）字符

如果用S的话，就可以这样写：S:7:"\00*\00name" ，简单来说就是在S的模式下，\+字符的ascii码的16进制表示就可以被识别为字符，这里的s:7:"%00*%00name" 甚至可以直接表示成 S:7:"\00*\00\6e\61\6d\65" ,通过这种方法可以绕过很多过滤（来自强网杯的怨念）,具体原理可以参考https://www.neatstudio.com/show-161-1.shtml

4.反序列化逃逸

这个好复杂啊，不想写懒狗选择放弃，大家可以百度安恒月赛反序列化逃逸来找例子参考。

https://www.cnblogs.com/BOHB-yunying/p/12774297.html

这个师傅就写的很好，可以参考一下。

7.弱比较问题

7.1 数字与字符串的比较

字符串与数字比较的时候，会自动提取字符串中的数字。

例如

1=="1" 可以通过

1=="1a" 可以通过

注：这里从字符串中提取数字的操作是从首位开始，如果首位不是数字就不会提取数字，1=="aa1" 不可以通过。

8.PHP标签的几种写法

1. <?php ?>
常规写法

2. <? ?>

注：

- 利用短标签写法可以绕过一些对php字符的过滤
- Windows环境中短标签默认是打开的，Linux下 默认是关闭的。
- 控制参数： php支持短标签，需要我们把short_open_tag 设置为On.

3. <% %>

注：需要配置php.ini文件。在配置文件中找到asp_tags=off ,将off改为on。改动配置文件后需要重启apache。

4. <script language=”php”> </script>

9.php中的命令执行函数总结

10.disabled_function绕过

11.PHP下的模板注入TWIG

如果目标网站没有对模板进行过滤，即可通过一下方法执行命令。

```
{{_self.env.registerUndefinedFilterCallback("exec")}}{{_self.env.getFilter("shellcode")}};
```

12.PHP伪协议

1.filter协议 ，利用该协议可以读取文件

```
php://filter/resource=1.php  //该情况下php代码会被执行
php://filter/read=convert.base64-encode/resource=1.php //通过转义可以避免被执行，直接读取源码
php://filter/read=string.rot13/resource=1.php
```

有几种常见的编码转换参数

string.rot13

string.toupper

string.tolower

string.strip_tags //去除 HTML 、 PHP 标记 、空字符，php标签里所有东西都会被去除，html只有标签会被去除，里面的文字不会删除。

convert.base64-encode

convert.base64-decode

zlib.deflate　//压缩

zlib.inflate　//解压

注：该协议可以嵌套使用，php://filter/resource=test/../1.php或者php://filter/resource=test/1.php,test文件夹不存在的情况下这两种写法下仍然可以读取1.php文件内容。

2.file协议，该文件也是利用来直接读取文件

file:/etc/passwd

13.PHP变量覆盖漏洞

PHP<5.4.0版本下，可能存在该漏洞。

参见大佬的文章，写的很好

PHP变量覆盖漏洞小结 [Mi1k7ea]

暂时就写到这儿如果后续还有其他说话，会继续更下来，以后如果有时间会写一个php一些伪协议的总结。