file_include

[攻防世界 (xctf.org.cn)]

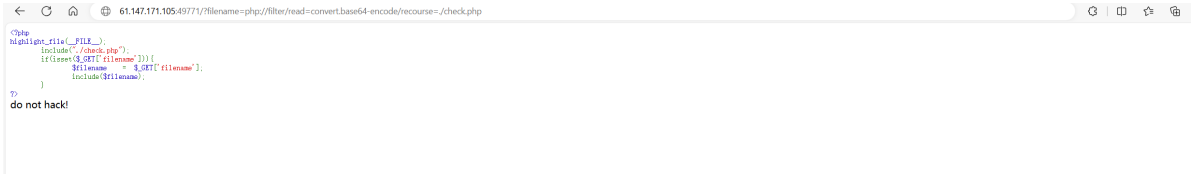
总结：对于fillter过滤器的更加深刻的认知，是一个好题

1，初见

看到就是一个check.php的过滤，使用昨天学的中间流过滤来完成

php://filter/read=convert.base64-encode/recourse=./check.php

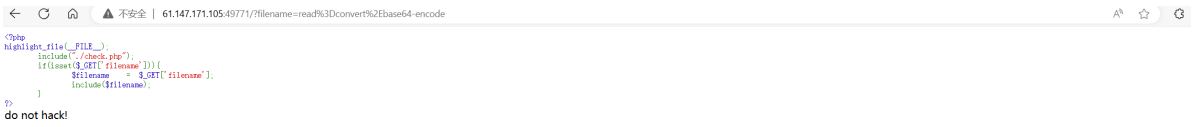
可以看到对于这样是过滤了



尝试使用url编码也无法绕过：

“read=convert.base64-encode”的URL编码是：

`%72%65%61%64%3D%63%6F%6E%76%65%72%74%2E%62%61%73%65%36%34%2D%65%6E%63%6F%64%65`



是什么过滤了吗，在尝试后发现

base64，read，encode，这三个全部过滤了

那就无法利用了吗？在之前的学习中了解到read是可以省略了，省去后

convert.base64的过滤器只是一个基础操作，通过对下面的学习

[php://filter的各种过滤器_php过滤器列表-CSDN博客]

fillter过滤器的完整使用

1，string的字符过滤器，比如rot13，strip_tags等

利用姿势：

```
php://filter/string.strip_tags/resource=flag.php
php://filter/string.rot13/resource=flag.ph
```

但是一旦匹配string就无法使用了

2，convert的转换过滤器

除了传统的base64加密外，还有

convert.quoted-printable-encode：`convert.quoted-printable-encode` 是 PHP 中的一个过滤器，用于将数据编码为可打印字符引用编码（Quoted-Printable Encoding）。这种编码方式主要用于将二进制数据或包含特殊字符的数据转换为仅包含可打印 ASCII 字符的格式

convert.iconv.*
这个过滤器需要 php 支持 iconv，而 iconv 是默认编译的。使用convert.iconv.*过滤器等同于用iconv()函数处理所有的流数据。

```
convert.iconv.<input-encoding>.<output-encoding>
or
convert.iconv.<input-encoding>/<output-encoding>
```

编码方式有几种

```
UCS-4* UCS-4BE UCS-4LE* UCS-2 UCS-2BE UCS-2LE UTF-32* UTF-32BE* UTF-32LE* UTF-16*
UTF-16BE* UTF-16LE* UTF-7 UTF7-IMAP UTF-8* ASCII*
```

3，Compression Filters（压缩过滤器）

zlib.deflate（压缩）和 zlib.inflate（解压）

4，Encryption Filters（加密过滤器）//在7.1版本移除

mcrypt.*和 mdecrypt.*

这种不常见，暂时不研究

continue

针对上面的4种形式，发现string，zlib，quoted都被过滤了，只有iconv了

既然在同一个过滤规则下，那能读到check.php,那么flag.php也能读到呗

我们可以编写一个脚本来爆破一下允许什么编码，也可以一个一个试

运气不错

```
filename=php://filter/convert.iconv.UTF-8.UTF-7/resource=flag.php
```

试出来了

```
+ADw?php +ACQ-flag+AD0'cyberpeace+AHs-72aab16b38c98186b9a3d38eec9d8ee1+AH0'+ADs
```

写一个python脚本解密：

<?php $flag='cyberpeace{72aab16b38c98186b9a3d38eec9d8ee1}';

然而存在疑问是

```
filename=php://filter/convert.iconv.UTF-8/UTF-7/resource=flag.php
```

这样为什么反而不行，这样写和最开始的构造语句是等价的，但为什么没有回现？