- 本题是一个综合性练习题，打开环境后会发现只有一句话，显然我们需要寻找题目入口点，通过扫描或者直接访问都可以发现网站存在 `robots.txt`

```
User-agent: *
Disallow: 1nd3x.php
```

访问目标页面得到源码

```
 <?php
highlight_file(__FILE__);
error_reporting(0);

if($_REQUEST['a'] === 'hello'){
    die('no');
}

if($_GET['a'] != 'hello') {
    die('no');
}
$d = intval($_GET['data']);
if ($d < 1000 || chr($d) !== 'A') {
    die('no');
}

$a = $_GET['p'];
$b = $_POST['s'];

if ( sha1($a) === sha1($b) && $a != $b){
    echo getenv('FLAG');
}
?>
```

第一个要求参数a不能是hello，第二个if要求GET参数得是hello，如果我们直接传输GET参数a=hello的话，会发现返回no，这是被第一个条件所拦截了，因为在REQUEST会取得 `GET` 、 `POST` 、 `COOKIE` 内的参数，那么我们该如何绕过这个判断呢？这里我们考虑如果我们同时在GET和POST中都传输了一个a参数，那么REQUEST最终会保留谁的值呢？

- 在PHP中，默认情况下REQUEST的排序为 `GPC` ，新值会覆盖旧值，也就是说同名的情况下会有限保留COOKIE，没有则保留POST中的值。这个设置可以通过改变php.ini（PHP配置文件）文件中的 `request_order` 来改变。
- 第二关需要我们传递一个值，再经过intval > 1000，显然这里我们不能使用一些绕过intval的方法来做了，因为后续的操作都是使用intval之后的数据而不是 `$_GET['data']` 本身，还要求这个数字经过 `chr` 函数后得到 `A` ，我们先看一下chr函数的介绍

```
Generate a single-byte string from a number
```

通过数字生成一个单字节字符串，其实这就是一个将ASCII码转换为字符（应该叫做单字节字符串，因为PHP实际上没有字符的概念）的函数。对于A来说，其ASCII码为65，但是前面又规定了

输入要大于1000，这里我们可以考虑本地先测试一下chr的输出

```php
<?php
echo ord(chr(1000));
```

执行本代码我们发现输出竟然是 `232` ，（ord是一个将字符转换为ASCII码的函数，这里使用它是因为输出是一个不可见字符，转换为数字方便我们操作）
你可以再尝试一些其他的输入，最终你会发现 `chr` 是会对输入进行取模256的操作的。

- 其实到这里我们了解这个技巧就够了，但是如果你想继续研究其原因是什么？或者是害怕在后续学习途中遇到类似的问题，知道一些技巧但别人问你为什么却答不出？我们可以继续往下研究
在PHP手册中我们可以看到官方的答案

```
chr(int $codepoint): string
```

返回单字符字符串，包含将 **codepoint** 作为无符号整数解释的指定字符。

这可用于在一种单字节编码（像是 ASCII、ISO-8859 或 Windows 1252）中通过传递想要的字符在编码映射表中的位置来创建单字符字符串。但是，注意此函数并不清楚任何字符串编码，特别是无法通过传递一个 Unicode 码位值来生成多字节编码（像是 UTF-8 或 UTF-16）字符串。

此函数与 ord() 互补。

参数

codepoint
一个介于 0 与 255 之间的整数。

超过有效范围（0..255）的值将和 255“按位与”，与以下算法等效：

```
while ($bytevalue < 0) {
$bytevalue += 256;
}
$bytevalue %= 256;
```

已经说的很明白的，但我们还可以继续查看PHP的底层源代码
下载PHP源代码，在 `ext/standard/string.c` 中2491行你可以找到 `chr` 函数的源码

```
/* {{{ Converts ASCII code to a character
Warning: This function is special-cased by zend_compile.c and so is bypassed for constant ir
PHP_FUNCTION(chr)
{
    zend_long c;

    ZEND_PARSE_PARAMETERS_START(1, 1)
        Z_PARAM_LONG(c)
    ZEND_PARSE_PARAMETERS_END();

    c &= 0xff;
    RETURN_CHAR(c);
}
/* }}} */
```

其中c便是输入的参数，显然我们可以看到这里有个与操作，只保留低8位，也就是等于取模256。之后对于其他的函数，你依然可以通过搜寻 `PHP_FUNCTION(xxx)` 找到它们在源码中的位置，从而了解其隐藏的技巧。说不定哪天能挖到一个PHP源码级CVE呢：）

- 回到题目，最后一关就显得很简单了，我们直接使用数组绕过即可，最终完成payload如下

```
<?php
highlight_file(__FILE__);
error_reporting(0);

if($_REQUEST['a'] === 'hello'){
        die('no');
}

if($_GET['a'] != 'hello') {
        die('no');
}
$d = intval($_GET['data']);
if ($d < 1000 || chr($d) !== 'A') {
        die('no');
}

$a = $_GET['p'];
$b = $_POST['s'];

if ( sha1($a) === sha1($b) && $a != $b){
        echo getenv('FLAG');
}
?>  NSSCTF{1fbebc69-f75e-48d4-85b8-00e1a7d73a26}
```