代码规范：一般字符串就使用双引号吧，转义字符还多一点。

本地调试然后在web访问测试

php的设计与C不同，因为php是响应式的，每一次访问都会执行一次程序，而C是启动、关闭式在执行期间可以把数据存在内存中。所以web后端需要配合数据库来存储数据。

Linux下php-fpm默认限制的后缀名为php，如未取消限制，访问将出现access denied。测试想要查看执行结果，需修改php-fpm.conf中的security.limit\_extensions为空，即允许任意后缀名文件作为php解析。

CGI：Common Gateway Interface 是 Web Server 与 Web Application 之间数据交换的一种协议。cgi-bin是一个常见的目录。本质上来讲就是将web和本地程序连接起来，但是按照cgi的格式。早期基本是编译语言C、C++、Perl编写，实际上可以使用任何一门语言编写，因为都支持标准输入输出以及环境变量。提供地址空间保护（因为是编译出来的，操作系统给予支持，但是解释型语言没有支持），不会影响web Server。cgi不支持数据库连接池，同时存在不断创建进程的问题。反正在大量并发请求的情况下对资源的消耗有点恐怖。

FastCGI：同 CGI，是一种通信协议，但比 CGI 在效率上做了一些优化。同样，SCGI 协议与 FastCGI 类似。与nginx服务器通信使用这种技术，基于tcp:9000

PHP-CGI：是 PHP （Web Application）对 Web Server 提供的 CGI协议的接口程序。将PHP作为cgi，这时候作为一个独立的进程运行，有请求过来的时候就会创建一个进程进行响应。原本当php作为apache的一个模块时，php将作为apache的一个子进程存在，接受apache调用。

PHP-FPM：是 PHP（Web Application）对 Web Server 提供的 FastCGI 协议的接口程序，额外还提供了相对智能一些任务管理。

标准CGI变量：在所有Apache配置中的环境变量（另一个作用就是用在日志中）和由操作系统外壳传来的环境变量之外， 还有一组环境变量用以提供CGI脚本和SSI页面。用环境变量操作指令来覆盖或修改标准的CGI变量是不可能的。当用suexec来载入CGI脚本时， 环境变量将会被清除到在CGI脚本载入之前只剩一套安全变量。

详细介绍：CGI 脚本是任何运行在web服务器上的程序. CGI意思是Common Gateway Interface。CGI程序是通过标准输入（stdin）或系统环境变量来得到服务器的输 入信息，并通过标准输出（stdout）向服务器输出信息。CGI脚本是用下列两种方法使用的: 作为一个表单的ACTION 或 作为一个页中的直接link。和基本代码的行为一样，就是需要web服务器开启模块的支持，但就是能产生所有的web服务器都能理解的结果，你可以用你熟悉的任何语言编写CGI脚本, 只要你的脚本遵守下一节所陈列的规则即可，只要那个语言能在你的Web服务器系统上运行。

下面这个shell代码就是一种cgi脚本：

#!/bin/sh //表明身份

echo Content-type: text/plain //必须要有一个表明服务器传递内容类型的头部

echo /bin/date



奇怪的问题:

echo 255<124; PHP7.2没有输出,print系列函数也是，但是表达式为真时会输出1

推荐使用var\_dump

php -i | grep php.ini 快速找到php.ini的位置

Composer 是 PHP 的一个依赖管理工具，类似于Nodejs的npm。

PHPUnit 测试框架

常用模块：

xml json zip

数据库相关：mysqli pdo pdo\_mysql redis

图像处理：gd

网络支持（发包）：curl

字符串：mbstring iconv（编码转换，默认启用）

加密扩展库：Crack，Hash，Mcrypt，Mhash，OpenSSL，Sodium

扩展库安装：pecl

反转：strrev 但是对Unicode字符串无效，会乱码

切割：

explode(切割字符串，被切割字符串[, limit]) 最后产生一个数组，不会保留切割字符串

str\_split(被切割字符串[，切割长度]) 默认切割长度为1. 返回数组。

合并：

implode

字符串与数字类型直接使用比较符：

字符串类型会被转成数字类型然后进行比较，如果转型失败，就是数字0

参数传递： 一个个参数之间通过+号隔开

<A HREF="/cgi-bin/myscript?arg1+arg2+arg3">run my script</A>

对应shell就是${1}作为第一个参数, ${2}作为第二个, ${3}作为第三个. 在双引号中不需要花括号。

调用另一个页面进行响应：

Location: ../docs/final.html 这时候服务器会自己再用一边请求路由算法来包括这个文件

如果是shell就直接echo Location: ../docs/final.html

设置响应码：

Status: 204 No Response

echo Status: 204 No Response

对方法的接受：

通过环境变量REQUEST\_METHOD判断使用了哪种方法进行参数传递，其中的值就是字符串GET或者字符串POST

GET，它将数据打包放置在环境变量QUERY\_STRING中作为URL整体的一部分传递给服务器。所有的参数全都在一个变量中，所以上面的get格式与一般web应用的get格式相差很大。

POST做很多类似GET同样的事情, 不同的地方就是它是分离地传递数据给脚本. 你的脚本通过标准输入获取这些数据. (有些Web服务器是存储在临时文件中.) 这个QUERY\_STRING环境变量将不再设置. 通过环境变量CONTENT\_LENGTH获取长度，再从标准输入流中读取，其中的内容是经过URL编码的。

如果错误不开启，那么php代码错误访问就不会有页面也就是页面找不到，但是这不是服务器返回的，因为服务器认为php是存在的，所以返回的不是服务器配置中设定的404页面。

<?php

$a=array("Volvo"=>"XC90","BMW"=>"X5","Toyota"=>"Highlander");

print\_r(array\_keys($a));

?>

$array ={1,2,3};是错的，cars=array("Volvo","BMW","SAAB"); 才对

sizeof($arrays);不同于C\C++返回的就是元素个数而不是内存大小。

数据库模块：（主流mysqli，pdo）

yum安装php-mysql、php-gd没想到没有改配置文件么

mysql\_connect过时，把mysql改为mysqli，最后记得关闭数据库连接。如果是new出来的对象，直接$xxx->close();

没有数据返回就Notice: Trying to get property of non-object in /path/birth.php on line 8

查询完以后结果集要自己释放内存：

$stmt = $con->prepare("select question,answer from PE\_exam where question like ?");

//$stmt->bind\_param('s', "%".$\_POST['question']."%");

$question ="%".$\_POST['question']."%";

$stmt->bind\_param('s', $question);//不能传递常量只能是变量

$stmt->execute();

$result = $stmt->get\_result();

$row = $result->fetch\_array(MYSQLI\_NUM);

var\_dump($result);

$stmt->close(); #这就命令进行结果集的释放。But calling $stmt->free\_result() will not clear out the memory used by the prepared statement in which case you must use $stmt->close().

直接执行数据库语句的函数适用于安全的且语句短的情况，bindValue适用于不安全的或者配合switch实现很多参数的长语句。

$dbhost = 'localhost:3307';

$dbuser = 'root';

$dbpass = '';

$conn = mysqli\_connect($dbhost, $dbuser, $dbpass); //端口指明可以放在第一个参数也可以放在第五个参数

mysqli\_select\_db($conn ,'RUNOOB'); //数据库的选择可以作为连接函数的第四个参数

这是结果集，还要使用fetch\_系列函数来提取结果。

object(mysqli\_result)#2 (5) {

["current\_field"]=>

int(0)

["field\_count"]=>

int(6)

["lengths"]=>

NULL

["num\_rows"]=>

int(5)

["type"]=>

int(0)

}

$row = $result->fetch\_array(MYSQLI\_NUM); //查询到多个也只会先返回一个。MYSQLI\_NUM意味着索引使用数字。

array(6) {

[0]=>

string(76) "判断题：身体素质的发展与形态机能的发育是无关的.（）"

[1]=>

string(1) "0"

[2]=>

string(1) "1"

[3]=>

NULL

[4]=>

NULL

[5]=>

string(9) "不正确"

}

$row = $result->fetch\_all();//相当于一次性全部输出的fetch\_array(MYSQLI\_NUM) 子数组数据格式和其一样

select count(id) from ITclass where name=?。对于这句话num\_rows一定是1。

Warning: mysql\_connect(): [2002] No such file or directory

If you have /tmp/mysql.sock but no /var/mysql/mysql.sock then...

cd /var

sudo mkdir mysql

sudo chmod 755 mysql

cd mysql

sudo ln -s /tmp/mysql.sock mysql.sock

If you have /var/mysql/mysql.sock but no /tmp/mysql.sock then

cd /tmp

ln -s /var/mysql/mysql.sock mysql.sock

Centos在/var/lib/mysql/mysql.sock

Parse error: syntax error, unexpected '$con'可能分号没了

json：

<?php

$arr = array ('a'=>1,'b'=>2,'c'=>3,'d'=>4,'e'=>5);

echo json\_encode($arr);

?>

以上例程会输出：

{"a":1,"b":2,"c":3,"d":4,"e":5}

<?php

error\_reporting(E\_ALL & ~E\_NOTICE);#没有jsoncallback传入就是报个Notice。一行行解析下来所以要放在开头。

$jsoncallback = htmlspecialchars($\_REQUEST ['jsoncallback']);

$limit\_num =3;

$con=mysqli\_connect("localhost",'guest','123456','collage');

if($con->connect\_error) die('Could not connect: '.$con->connect\_error);

$result =$con->query('select \* from birth where birthday>'.date('md').' order by birthday limit '.$limit\_num);

while($row =$result->fetch\_assoc() ) {

$a[$row['name']]=$row['birthday'];

}

if( $result->num\_rows<$limit\_num ){

$need = $result->num\_rows-$limit\_num;

$result =$con->query('select \* from birth where birthday>0 order by birthday limit '.$need);

while($row =$result->fetch\_assoc() ) {

$a[$row['name']]=$row['birthday'];

}

}

echo $jsoncallback.'('.json\_encode($a,JSON\_UNESCAPED\_UNICODE).')'; #中文转码

?>

<?php

//Record Ip将ip记录和jsonp合在一起做。

echo "$limit\_num"; #可以输出3

if($\_SERVER['HTTP\_CLIENT\_IP'] !=0) { #如果有这个头（代理服务器可能会有这个头）

$out\_ip=$\_SERVER['HTTP\_CLIENT\_IP'] ;

}else {

$out\_ip= $\_SERVER['REMOTE\_ADDR'];

}

$result =$con->query('select \* from ip where ip=\''.$out\_ip.'\'');

if( $result->num\_rows==0 ) {

$con->query('insert into ip(ip,time) values(\''.$out\_ip.'\',0)');

}else {

$con->query('update ip set time='.(($result->fetch\_row())[time]+1).' where ip=\''.$out\_ip.'\''); #这里看起来在$out\_ip这一部分可能有注入漏洞

}#fetch\_row()是获取一行

$con->close();

?>

同时发现多拨访问ip不是固定的。

$link = mysqli\_connect($url,$usr,$paw,$database)

or die("Error " . mysqli\_error($link));//不错的写法 。这种写法还出现在msql\_query函数之后。但是最后的生产环境中是将这个要去掉，因为将报错显示在外。die（exit函数的别名）只能用于调试。直接结束的话建议使用return哦。

php防注入：

使用mysqli支持的参数化语句

$stmt = $mysqli->prepare("INSERT INTO table\_name (name) VALUES (?)"); //$mysqli是连接数据库的对象

$stmt->bind\_param('s', $postedName);//stmt名字取得好，虽然我也不知道什么意思

$stmt->execute();

在 prepare 函数里面把参数用 ‘?’ 来替代，然后使用 bind\_param 绑定参数。在 bind\_param 中，第一个参数 's' 代表了参数的类型与个数（此处为一个字符串类型）。如官方的例子

$stmt = $mysqli->prepare("INSERT INTO CountryLanguage VALUES (?, ?, ?, ?)");

$stmt->bind\_param('sssd', $code, $language, $official, $percent);

就是绑定了四个参数，三个为字符串，一个为数字。非常简便。

或者使用PDO（PHP Database Object）

是个额外的扩展。觉得没有mysqli方便。不过怕是要了解点。

面向对象风格

bool mysqli\_stmt::bind\_param ( string $types , mixed &$var1 [, mixed &$... ] )

过程化风格

bool mysqli\_stmt\_bind\_param ( mysqli\_stmt $stmt , string $types , mixed &$var1 [, mixed &$... ] )

This code did not work:

<?php

$test = $sql->prepare("SELECT name FROM names WHERE name LIKE %?%");

$test->bind\_param("s", $myname);

?>

The solution is:

<?php

$test = $sql->prepare("SELECT name FROM names WHERE name LIKE ?");

$param = "%" . $myname . "%";

$test->bind\_param("s", $param);

?>

数据查询的两种姿势：直接执行还是嵌入参数

<http://php.net/manual/en/mysqli.query.php> 或者

<?php

//$\_POST['question']="";

error\_reporting(~E\_ALL);

if (!isset($\_POST['question']) || $\_POST['question']=="") {

echo "<script>alert('你需要更加详细的题目输入');history.go(-1);</script>";

exit(1);

}

//write into databases

$con=new mysqli("127.0.0.1",'guest','123456','collage');

if(mysqli\_connect\_errno()) die('Could not connect: '.mysqli\_connect\_error());

mysqli\_query($con, 'set names utf8');

//query(defense SQL injection)

$stmt = $con->prepare("select question,answer from PE\_exam where question like ?");

//$stmt->bind\_param('s', "%".$\_POST['question']."%");

$question ="%".$\_POST['question']."%";

$stmt->bind\_param('s', $question);

$stmt->execute();

$result = $stmt->get\_result();

$row = $result->fetch\_array(MYSQLI\_NUM);

$stmt->close();

//id already exits will update

if( $result->num\_rows==0 ) {

echo "<script>alert('你使用的姿势有问题，请联系电池改正！');</script>";

}else if($result->num\_rows==1){

echo "<script>alert('".$row[0]."的回答是：".$row[1]."');history.go(-1);</script>";

}else {

echo "<script>alert('你需要更加详细的题目输入');history.go(-1);</script>";

}

?>

如果没有MYSQLI\_NUM参数：

array(2) {

[0]=>

string(6) "正确"

["answer"]=>

string(6) "正确"

}

而有的话

array(1) {

[0]=>

string(6) "正确"

}

$result=mysqli\_query($con,$sql);//得到结果集

$xx =mysqli\_fetch\_all($result,MYSQLI\_ASSOC);

mysqli\_free\_result($result); // 释放结果集

var\_dump($xx);

array(3) {

[0]=>

array(2) {

["id"]=>

string(13) "2015141462001"

["info"]=>

string(19) "04.09,下午,江安"

}

[1]=>

array(2) {

["id"]=>

string(13) "2015141462002"

["info"]=>

string(19) "04.09,下午,江安"

}

[2]=>

array(2) {

["id"]=>

string(13) "2015141462002"

["info"]=>

string(19) "04.09,下午,江安"

}

}

与html混合：

写了条<?php echo 'test';?>结果被注释了。即使我把php放到<html></html>外面，审查元素时注释还是出现在里面。哦我知道了，想到用户连接访问以后服务器整个流程（寻找文件，根据文件类型和文件是否存在和权限，应用是否正常等作出反应）。我需要把后缀给为php才会给php解析。

还有种就是类似于js输出html，使用php输出html。echo<<<eot...eot

使用前端模板引擎如Smart

访问http://182.254.217.40/birth.php要是没有php就会不能访问，因为服务器那里只做了index的允许省略。

sso：登录一个账号 关联那几个都登录了

php session登录（同一用户多次登录会挤掉原有session）

<?php

session\_start();

//ini\_set('session.auto\_start', 0); //关闭session自动启动

//ini\_set('session.cookie\_lifetime', 0);//设置session在浏览器关闭时失效

//ini\_set('session.gc\_maxlifetime', 3600); //session在浏览器未关闭时的持续存活时间

//防止同一用户的重复登录

$server = '127.0.0.1'; //不要是localhost，否则可能会报错mysqli\_real\_connect(): (HY000/2002): No such file or directory。我也是很无语，用了一段时间才知道

$username = 'root';

$password = '';

$conn = mysql\_connect($server, $username, $password);

$e = mysql\_select\_db('test',$conn);

mysql\_query("SET NAMES UTF8");

//访问url，http://www.test.com/single\_login.php?name=zx&pwd=zx

header("Content-type:text/html;charset=utf-8");

$username =$\_GET['name'];

$password =$\_GET['pwd'];

//$ipdress = $\_SERVER['REMOTE\_ADDR'];

//

//$login\_time = time();//登陆时间更新

//

//$session\_id = session\_id();

//

//$\_SESSION['name']=$\_GET['name'];

//var\_dump($session\_id);

//$sql = "INSERT INTO tongji VALUES ('NULL','$data\_id','$time','$ipdress','$session\_id')";

//$result =mysql\_query($sql);

$sql = "select \* from slogin where username = '$username' and password = '$password'";

//var\_dump($sql);

$result =mysql\_query($sql);

$s = mysql\_fetch\_array($result);

//权限验证的时候一定要判断数据库写入session\_id是不是和$\_session里面的session\_id是不是一致，如果不一致就退出，这个防止同一个账户被多个人同时登陆的实现方法就是基于这个session\_id

//一定要设置session的存活时间，和session关掉页面就session消除

//如果有其他人登陆的时候，会写入新的session\_id，这样旧的用户就会被权限验证不通过，但是不能做到实时效果，这样就挤掉了旧用户，这里有个缺陷，就是每次验证权限的时候就需要查询数据库，如果可以吧这个数据放在redis就比较好

//其实其他的验证方法，也类似使用这种方法，有个验证的凭证，只不过这种方法需要查询数据库，但是放在redis之后就比较好

//或者统一管理session的时候，比如放在数据库就更好了，记住吧session\_id也存入数据库，或者放在memcash，redis就比较方便，比如在redis进行管理的时候，就可以直接更新掉存在session\_id

if($s){

// var\_dump($s);

// echo '<br>';

// echo session\_id();

if($s['session\_id'] === session\_id()){

//unset($s['session\_id']);

} else {

$ipdress = $\_SERVER['REMOTE\_ADDR'];

$login\_time = time();

$session\_id = session\_id();//重新赋予一个session\_id

$sql = "update slogin set ip = '$ipdress',login\_time= '$login\_time',session\_id = '$session\_id'";

$result1 =mysql\_query($sql);

// $ss = mysql\_fetch\_array($result);

if($result1){

echo 'OK';

} else {

echo 'F';

}

}

}else {

echo 'FFFFFFFFFFFFFFFFFFFFFFF';

}

/\* If we have to retrieve large amount of data we use MYSQLI\_USE\_RESULT \*/

$result = $mysqli->query("SELECT \* FROM City", MYSQLI\_USE\_RESULT)这时候才会用到第二个参数，一般默认第二个参数是 MYSQLI\_STORE\_RESULT 。这时候就需要自己释放结果集。

php调用可执行文件：

system() 输出并返回最后一行shell结果。

exec() 不输出结果，返回最后一行shell结果，所有结果可以保存到一个返回的数组里面。

passthru() 只调用命令，把命令的运行结果原样地直接输出到标准输出设备上。

相同点：都可以获得命令执行的状态码

php防止DDOS：

对了,顺便提一下,我在代码中用文件来记录来访者ip和时间,最好是别用数据库(也不要自作聪明地存入session中),另外最好是把这个文件放入SSD硬盘上,原因嘛我就不说了,估计大家都知道

用php来支持文件的下载。之前只知道用<a>。如果你在URL或者POST中有个参数给专门下载的php页面那么一定要做好对"."的过滤准备。

对于XML，php使用SimpleXML插件。

异常：Exception类

getMessgae()

getCode()

getFile()

getLine()

getPrivious 获得一个嵌套的异常对象

getTrace() 获得一个数组，高数组追踪导致异常的方法调用（方法的各个信息）

getTraceAsString()

\_\_toString()将对象转为字符串类型时自动调用。

echo 自带输出后换行

php发包之curl：需要开启curl模块

安装curl，yum和RPM安装或源码编译安装 ,三选一即可

yum install curl

rpm -ivh http://mirror.city-fan.org/ftp/contrib/sysutils/Mirroring/curl-7.42.1-1.0.cf.rhel6.x86\_64.rpm

##源码编译安装curl：

安装cURL

wget http://curl.haxx.se/download/curl-7.42.1.tar.gz

tar -zxf curl-7.42.1.tar.gz

./configure --prefix=/usr/local/curl（可选，默认安装在/usr/local/bin目录下）

make & make install

##两种方式安装php扩展：1.源码编译安装php，加 --with-curl=/usr/local/curl（curl目录）；2.编译安装php扩展curl.so

还有种phpize

yum install php53\_devel （ps：请注意自己的版本)

安装完毕后。会生成phpize命令

# cd /usr/local/src/php-5.2.6/ext/pcntl

# phpize

# ./configure --with-php-config=/usr/local/php/bin/php-config

# make && make install

这四种都是已经安装好php后的增加模块，所以需要你去改动php的配置文件/etc/php.ini(phpinfo来查看)来开启php对模块的支持。

输入phpize，错误提示如下：

Can't find PHP headers in /usr/include/php

The php-devel package is required for use of this command.

yum install php-pear

然后再安装php-devel：

yum install php54w-dev（我是7.2.1,没想到应该直接yum install php-devel）

**cURL（网络发包库）：**

具体执行函数有curl\_exec

但是封装成为curl\_fetch:

function curl\_fetch($ch, $args=null) {

if($args['method']=='POST'){

curl\_setopt($ch, CURLOPT\_POST, TRUE);

curl\_setopt($ch, CURLOPT\_POSTFIELDS, $args['post\_data']);

}

curl\_setopt ($ch, CURLOPT\_CONNECTTIMEOUT, isset($args['timeout'])?$args['timeout']:5);

curl\_setopt($ch, CURLOPT\_HEADER, isset($args['headers\_display'])?$args['headers\_display']==true:false);

$file\_contents = curl\_exec($ch);

if($args['method']=='POST')

curl\_setopt($ch, CURLOPT\_POST, false);

return $file\_contents;

}

设置curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, TRUE); 则不会输出curl到的页面，参数为TRUE表示我们希望结果作为一个变量存储，所以记得存储curl\_exec的返回值。

CURLOPT\_POSTFIELDS可以给Array（这时候会用boundary方式隔开），也可以给字符串。

curl\_setopt($ch, CURLOPT\_HEADER, TRUE); 记录响应头

curl\_setopt($ch, CURLOPT\_HEADER\_OUT, TRUE); 记录请求头

echo curl\_getinfo($ch, CURLINFO\_HEADER\_OUT); 获取请求头，如果是POST会看不到发送的内容  
curl\_setopt($curl, CURLOPT\_COOKIEFILE, ""); 空文件名表示将使用内存来存储，也就是自动维护过程中的cookie。获取：foreach(curl\_getinfo($curl\_a, CURLINFO\_COOKIELIST) as $cookie\_line)。清空cookie：curl\_setopt($ch, CURLOPT\_COOKIESESSION, TRUE);

**PHP发包之IO读写：**

function getcookie($config,$zjh,$mm)

{

// if ($config['ip'] == "202.204.208.73" && $config['port'] == "8033") $mm = md5($mm,false);

$link = fsockopen($config['ip'],$config['port'],$errno,$errstr,30);

if (!$link) exit ( $errstr . " ==> " . $errno);

$seekcookie = 0;

$cookie = 0;

$out = "GET " . $config['dictionary'] . "loginAction.do?zjh=" . $zjh . "&mm=" . $mm . " HTTP/1.1\r\n";

$out .= "Host: " . $config['ip'] . ":" . $config['port'] . "\r\n";

$out .= "Connection: close\r\n\r\n";

fwrite($link, $out);

while (!feof($link))

{

$seekcookie = fgets($link, 128);

$seekcookie = explode(" ",$seekcookie);

if (strncasecmp($seekcookie[1],"JSESSIONID",10) == 0)

{

$cookie = $seekcookie[1];

// $cookie = str\_replace(";","",$cookie);

$cookie = chop($cookie,";");

// echo $cookie;

break;

}

}

fclose($link);

return $cookie;

}

在yum中加入php高版本的源：

rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

rpm -Uvh https://mirror.webtatic.com/yum/el7/webtatic-release.rpm

阿里云找不到php71w

yum install php-curl

设置url：

$ch =curl\_init($url);

curl\_setopt($ch, CURLOPT\_URL, $url);

设置方法：默认GET

if($method='POST' && $post\_data != ''){

curl\_setopt($ch, CURLOPT\_POST, TRUE);//使用了POST那么下面这条就必须要有。即使$post\_data=''否则会报错

curl\_setopt($ch, CURLOPT\_POSTFIELDS, $post\_data);

}

curl\_setopt($ch, CURLOPT\_PUT, TRUE);

设置http 请求头：

curl\_setopt($ch,CURLOPT\_HTTPHEADER,$this\_header);

设置超时时间：

curl\_setopt ($ch, CURLOPT\_CONNECTTIMEOUT, $timeout);

curl\_exec返回字符串形式：这样才能有转码什么的

curl\_setopt ($ch, CURLOPT\_RETURNTRANSFER, TRUE);

显示http 响应头：

curl\_setopt($ch, CURLOPT\_VERBOSE, 1); will output response http headers

curl\_setopt($ch, CURLOPT\_HEADER, true); //这句设置为false，上面那句也能起到效果

$header\_size = curl\_getinfo($ch, CURLINFO\_HEADER\_SIZE);

$header = substr($response, 0, $header\_size);

$body = substr($response, $header\_size);

自动记录cookie：三句都有保存cookie的功能

curl\_setopt($ch, CURLOPT\_COOKIESESSION, true );//仅限cookie中的SESSION

curl\_setopt($ch, CURLOPT\_COOKIEFILE, $args["cookiefile"]);

curl\_setopt($ch, CURLOPT\_COOKIEJAR, $args["cookiefile"]); //同时记录一份给$args["cookiefile"]

URL编码函数：

curl\_escape — URL encodes the given string 要求版本是5.5以上才会有这个函数。

curl\_unescape — Decodes the given URL encoded string

curl库版本信息：

curl\_version — Gets cURL version information

对于curl获取包中文乱码问题，有以下三种解决方法：

0.特殊：如果全是乱码说得到的流被编码了，很可能是gzip编码，使用gzdecode($res);

1.

$this\_header = array(

"content-type: application/x-www-form-urlencoded;

charset=UTF-8"

);

curl\_setopt($ch,CURLOPT\_HTTPHEADER,$this\_header);

2.

extension=php\_mbstring.dll

string mb\_convert\_encoding ( string str, string to\_encoding [, mixed from\_encoding] )

mb\_convert\_encoding($str, 'utf-8', 'GBK,UTF-8,ASCII');#but undefined.排列的顺序ASCII永远要排在后面。

一般情况下用 iconv，只有当遇到无法确定原编码是何种编码，或者iconv转化后无法正常显示时才用mb\_convert\_encoding 函数.

3.

iconv

iconv命令是运行于linux平台的文件编码装换工具。当我们在linux系统shell下通过curl命令或者wget命令获取一个网页的源代码，当网页的编码与当前操作系统坏境的设置的编码不同时，就会发现网页中有很多乱码。如在网页"meta"标签"charset"属性值设置为"gb2312"的http://www.baidu.com百度首页，在系统坏境变量"$LANG"值为"en\_US.UTF-8"的linux系统即会产生中文乱码现象。这时我们可以尝试使用iconv命令进行编码装换，让中文不在是乱码。如下命令是处理百度在系统坏境变量"$LANG"值为"en\_US.UTF-8"的linux系统乱码的问题的解决方案之一：

curl http://www.baidu.com|iconv -f gb2312 -t utf-8

curl http://pead.scu.edu.cn/stu/|iconv -f gb2312 -t utf-8 #gbk

coding htmlGB2312是GBK的子集

iconv("gb2312","utf-8//IGNORE",$content);

iconv("utf-8","gb2312//IGNORE",$content);

html 解析：

echo strpos($str, '<label>'); //echo strstr会打印出找到位置开始的字符串

strpos没有找到会返回FALSE，需要用===进行判断

感觉这个比正则快一点或者差不多

preg\_match('/\d{5,6}/', $str, $num\_matchs); //preg\_match\_all

$verify\_code =$num\_matchs[0];不过正则的好处在于方便

$str =iconv('gbk', 'utf-8//ignore', curl\_exec($ch)); 转码后就可以直接匹配中文

<http://www.php.cn/php-weizijiaocheng-374177.html> 说明正则表达式的\w不等于[0-9a-zA-Z\_]。\w不能匹配中文。

Linux下匹配一个中文：preg\_match('/.{3}/', "目标场次：20181", $result);

使用正则匹配手机号，因为php的类型可以方便地自动转换，所以可以直接除以1e11，然后结果是不是1 。这种已经能够验证大部分情况，但是浮点数等输入会被排除。不过一般人应该想不到偏偏输入这个特例。

正则的使用还有在preg\_split，preg\_replace（preg\_replace对表达式起始结束符支持好几种的，/与#与|）

[\s\S]\*? 借助?做到最小匹配

与正则相关的安全漏洞：



<?php 在这里随意加一些字符不带分号

正常php代码.....

?>

然后这个php页面的源码就会被显示

在其他地方加就会是报错

<?php

$str ="123456";

echo strlen($str)."\n";

echo substr($str, -6, -1)."\n";

echo substr($str, -7, -1)."\n";

?>

6

12345

12345

$\_GET这类是一个下划线

常量和方法倒是两个下划线

PHP数据类型有三种转换方式：

在要转换的变量之前加上用括号括起来的目标类型

使用3个具体类型的转换函数，intval()、floatval()、strval()

使用通用类型转换函数settype(mixed var,string type)

第一种转换方式： (int) (bool) (float) (string) (array) (object)

<?php

$num1=3.14;

$num2=(int)$num1;

var\_dump($num1); //输出float(3.14)

var\_dump($num2); //输出int(3)

?>

第二种转换方式： intval() floatval() strval(

<?php

$str="123.9abc";

$int=intval($str); //转换后数值：123

$float=floatval($str); //转换后数值：123.9

$str=strval($float); //转换后字符串："123.9"

?>

第三种转换方式： settype();

<?php

$num4=12.8;

$flg=settype($num4,"int");

var\_dump($flg); //输出bool(true)

var\_dump($num4); //输出int(12)

?>

<?php

$a =array(1,2);

unset($a[0]);

echo $a[0];

echo $a[1];

?>

PHP Notice: Undefined offsen /root/Desktop/test.php on line 4

2

<?php

$test ='1212312312312';

echo $test+4;

?>

1212312312316

print("$(date(\"H:i a\"))"); $没有转义却会如转义般输出

$(date("H:i a"))

php文件禁止被访问：

从服务器角度直接设定为403 Forbidden

location ~/PE/pe\_scan.php {

deny all;

}

在php文件中检查：

!$\_SERVER['HTTP\_REFERER'] or die('forbidden!'); 这个是设定不能从url直接访问（而不是不能访问）但是可以用html编辑器在网页上加一个<a></a>或者自己修改http包头

ignore\_user\_abort(TRUE) 函数设置与客户机断开是否会终止脚本的执行。可以用于触发隔一段时间做什么的行为。

在shell中后台运行php隔时扫描脚本：

php PE\_scan.php&

雅黑PHP探针,可以实时查看服务器硬盘资源、内存占用、网卡流量、系统负载、服务器时间等信息。开一个外网访问。

<?php

date\_default\_timezone\_set('Asia/Shanghai');

echo date("Y-m-d H-i-s");

?>我把系统时间改了但是php的date还是没改回来，直到我设置了时区。

当对使用date() 等函数时，如果 timezone 设置不正确，在每一次调用时间函数时，都会产生 E\_NOTICE 或者 E\_WARNING 信息

php的date函数返回字符串，所以date("m")会返回的是04而不是4。

<?php

echo "今天:".date("Y-m-d")."<br>";

echo "昨天:".date("Y-m-d",strtotime("-1 day")), "<br>";

echo "明天:".date("Y-m-d",strtotime("+1 day")). "<br>";

echo "一周后:".date("Y-m-d",strtotime("+1 week")). "<br>";

echo "一周零两天四小时两秒后:".date("Y-m-d G:H:s",strtotime("+1 week 2 days 4 hours 2 seconds")). "<br>";

echo "下个星期四:".date("Y-m-d",strtotime("next Thursday")). "<br>";

echo "上个周一:".date("Y-m-d",strtotime("last Monday"))."<br>";

echo "一个月前:".date("Y-m-d",strtotime("last month"))."<br>";

echo "一个月后:".date("Y-m-d",strtotime("+1 month"))."<br>";

echo "十年后:".date("Y-m-d",strtotime("+10 year"))."<br>";

//strtotime()函数的作用是将日期时间描述解析为 Unix 时间戳

//int strtotime ( string time [, int now] )

?>真好，我不需要处理日期的加减进位了！

<?php

echo time()."\n";

echo strtotime(date("Y-m-d H:i:s", strtotime("14:00")));

?>差值用于进行计时不是很精准的sleep()

time()没有参数，如果给没有参数的函数传递参数是无效且不会有错的。

<?php

echo strtotime("2017-04-11 0:0:0");

echo strtotime("2017-04-11");

?>//两者是相等的

你可以把任何一个html文件改成.php,用于测试是否服务器已经可以支持显示php文件。你也可以把最外面套上<?php与?>

从可读性来讲，人最适合阅读的是二维数组。三维开始就变得糟糕了。

把数据接收页面和界面显示页面做到一起，比如说表单提交给自己，我认为这样就需要在开头来个if分支，我见过做的糟糕的就是体育学院改密码，它是靠当前session访问字数来判断是提交数据还是访问页面。

<?php

$data =Array(); //这句话可以没有，但不能是new Array();

$date['江安']=Array(Array(318,401),Array(405,419));

$date['望江']=Array(Array(421,428),Array(502,513));

var\_dump($date);

?>

遍历数组的几种特殊方法：

第一、foreach()//不嗯能够对字符串使用，它不是可遍历的

foreach (array\_expression as $value) #$value是个临时变量

statement

foreach (array\_expression as $key => $value)

statement

<?php

$urls= array('aaa','bbb','ccc','ddd');

foreach ($urls as $url){

echo "This Site url is $url! <br />";

}

?>

数组最后一个元素的 $value 引用在 foreach 循环之后仍会保留。建议使用 [unset()](http://php.net/manual/zh/function.unset.php" \t "_blank) 来将其销毁。<http://m.blog.csdn.net/lengyue1084/article/details/43022403>

第二、while() 和 list()，each()配合使用。

<?php

$urls= array('aaa','bbb','ccc','ddd');

while(list($key,$val)= each($urls)) {

echo "This Site url is $val.<br />";

}

?>

在php和js当中(1<2)+(3<4);true+true;都会得到2而不是true。

php中的函数名，大小写也都随意。所以变量前面都要求有个$。

如果用户提交表单不对，我们用js：

echo "<script>alert('dfsf');history.go(-1);</script>";

echo intval('123-afd123');只会输出开头的123，可以防注入。我前端也限制了input name="id" type="number" required>。

echo intval('-');是个0。

php的进程控制函数需要在编译的时候--enable-pcntl来打开模块

或者进源码目录ext/pcntl目录来用源码编译安装pcntl.so

运行

phpize

[root@localhost pcntl]# ./configure --with-php-config=/usr/local/php/bin/php-config && make && make install

把生成的文件放到默认的目录里：

[root@localhost pcntl]# cp /usr/local/php/lib/php/extensions/no-debug-non-zts-20090626/pcntl.so /usr/lib/php/modules/

然后在php.ini里加上这个extension

extension=pcntl.so

php守护进程：

在UNIX 系统中，一个进程结束了，父进程还在执行所以不是孤儿进程也就不会被init这1号进程接管，但是他的父进程没有等待(调用wait / waitpid)他， 那么他将变成一个僵尸进程。 但是如果该进程的父进程已经先结束了，那么该进程就不会变成僵尸进程， 因为每个进程结束的时候，系统都会扫描当前系统中所运行的所有进程， 看有没有哪个进程是刚刚结束的这个进程的子进程，如果是的话，就由Init 来接管他，成为他的父进程

一个进程在调用exit命令结束自己的生命的时候，其实它并没有真正的被销毁， 而是留下一个称为僵尸进程（Zombie）的数据结构（系统调用exit，它的作用是 使进程退出，但也仅仅限于将一个正常的进程变成一个僵尸进程，并不能将其完全销毁）

在每个进程退出的时候，内核释放该进程所有的资源，包括打开的文件，占用的内存等。但是仍然为其保留一定的信息（包括进程号the process ID，退出状态the termination status of the process，运行时间the amount of CPU time taken by the process等）。直到父进程通过wait / waitpid来取时才释放. 但这样就导致了问题，如果进程不调用wait / waitpid的话，那么保留的那段信息就不会释放，其进程号就会一直被占用，但是系统所能使用的进程号是有限的，如果大量的产生僵死进程，将因为没有可用的进程号而导致系统不能产生新的进程. 此即为僵尸进程的危害，应当避免。

守护进程的父进程就是用户态的init（1号进程）

在PHP中的函数是pcntl\_fork(). PCNTL是Process Control的缩写，PHP中与进程有关的函数都以此开头。

系统中的每个进程都有对应的父进程，进程的ppid标识符即为其父进程的pid。当子进程执行完毕之后，父进程应该通过wait请求这些信息，否则内核会一直保留子进程的状态信息。如果父进程没有调用wait来等待它，子进程就会变成僵尸进程。如果父进程先于子进程结束，子进程通常并不会受影响，而会一直执行下去。父进程结束时系统会扫描其子进程，并将其ppid置为1，成为init进程的子进程。

当终端收到终止信号(Ctrl-C)时，会转发给进程组中的所有进程。进程组是一个相关进程的集合，通常是父进程与其子进程。

会话组是进程组的集合，当在shell中执行命令：git log | grep shipped | less时，每个命令都有一个进程组，三个进程组属于同一个会话组。当终端收到Ctrl-C时，发送给会话leader的信号会被转发给该会话组的每个进程组，然后再被转发到进程组中的所有进程。

setsid系统调用会使衍生进程成为一个新会话组leader（同时也是新进程组leader），并返回新建的会话组id。在PHP中对应的函数是posix\_setsid();

posix\_setsid();

php创建守护进程fork了两次，原因是虽然设置新的会话组已经断了与当前终端的联系，但是这时候fork第一次还是由于调用了setid那么也就成了新会话组的leader，如果后期打开终端那么他的输出与输入是会到这个终端上的，C里面的解决方案是dup2重定向了输入输出，php里面则是fork两次（第一次setid，第二次就不是leader，不过父进程要记得等待子进程退出pcntl\_wait($status);否则就是僵尸进程了）

之所以要setid脱离原会话组是因为原会话组是会传递C-C，或者说是与一开始调用的终端断开关系。如果进程已经是会话组长时，setid()调用将失败，所以需要在fork后的子进程中调用setid。

别忘了除了setid还有umask和chdir。如果从父进程那里有文件操作符继承过来的话需要关闭。

umask(0)

linux中的 umask 函数主要用于：在创建新文件或目录时 屏蔽掉新文件或目录不应有的访问允许权限。文件的访问允许权限共有9种，分别是：r w x r w x r w x（它们分别代表：用户读 用户写 用户执行 组读 组写 组执行 其它读 其它写 其它执行）。

其实这个函数的作用，就是设置允许当前进程创建文件或者目录最大可操作的权限，比如这里设置为0，它的意思就是0取反再创建文件时权限相与，也就是：(~0) & mode 等于八进制的值0777 & mode了，这样就是给后面的代码调用函数mkdir给出最大的权限，避免了创建目录或文件的权限不确定性。

我们希望程序在非超级用户运行，这样一旦由于程序出现漏洞被骇客控制，攻击者只能继承运行权限，而无法获得超级用户权限。所以有些人用root启动web服务器，一旦攻破就直接有权限了。不过nginx强制要求不能用root账户来启动服务器。

创建守护进程的步骤：

（1）调用umask将文件模式创建屏蔽字设置为0

（2）父进程fork出子进程，然后子进程调用setsid，父进程直接退出。（断开终端联系，立即退出父进程这种写法是为了把子进程变成孤儿进程）

（3）调用setsid创建一个新的会话（调用成功会使调用进程成为新会话的首进程，并且成为一个进程组的组长进程，调用进程没有控制终端）

（3.5）之后不要再打开终端（不安全的写法），不过最好你重定向一下几个流或者让孙进程成为守护进程他也就不是是会话组组长。

（4）将当前工作目录更改为根目录

（5）关闭不在需要的文件描述符（继承自父进程）

\*忽略SIGCHLD信号，已经脱离了终端所以没有必要

如果你从shell里启动一个进程,当终端exit时,终端的控制进程也就是shell会给本会话期的所有进程发送SIGHUP信号,这样你不忽略该信号的话,只要你一断开终端你的进程就会退出.所以守护进程可以通过忽略SIGHUP来达到.

当然你也可以通过脱离控制终端加入另一个全新的全话期来达到,因为此时原来终端的控制进程已经不会给你发SIGHUP信号了. 这种不脱离，算取巧。

php设置与信号处理函数的连接：

pcntl\_signal(SIGTERM, "sig\_handler");

pcntl\_signal(SIGHUP, "sig\_handler");

pcntl\_signal(SIGINT, "sig\_handler");

pcntl\_signal(SIGQUIT, "sig\_handler");

pcntl\_signal(SIGILL, "sig\_handler");

pcntl\_signal(SIGPIPE, "sig\_handler");

pcntl\_signal(SIGALRM, "sig\_handler");

php比较高的版本ps -aux|grep php都可以看到PHP-FPM，似乎用于CPU、内存管理。

突然遇到一次php写入数据库编码乱码的问题，之前还好好的，我调试的时候是直接用utf-8文本中中文字符串写入的，而且数据库中能设置成utf8都已经确定设置了，但是就是乱码。最后的解决是在获取数据库连接以后调用了mysqli\_query($con, "set names utf8");

评价：其实按道理来讲是不应该的，因为输入和接受都已经设置了是utf8，我也无法解释为什么会出现这种问题。

存文件比存数据库安全

对数据库的操作我直接delete，而并没有下查询数据是否存在，而这样的使用相当好，并没有遇到什么错误。本以为是担心数据库的错误显示到界面上，不过其实到目前为止我似乎都还没遇到数据库显示到页面上的。

php体侧后台扫描程序：

<?php //method timeout headers\_display

function curl\_fetch($ch, $args=null) {

if($args['method']=='POST'){

curl\_setopt($ch, CURLOPT\_POST, TRUE);

curl\_setopt($ch, CURLOPT\_POSTFIELDS, $args['post\_data']);

}

//curl\_setopt ($ch, CURLOPT\_CONNECTTIMEOUT, isset($args['timeout'])?$args['timeout']:10);

curl\_setopt($ch, CURLOPT\_HEADER, isset($args['headers\_display'])?$args['headers\_display']==true:false);

$file\_contents = curl\_exec($ch);

return $file\_contents;

}

function login($ch, $headers,$password) {

$url ='http://pead.scu.edu.cn/jncx/';

curl\_setopt($ch, CURLOPT\_URL, $url.'logins.asp');

$headers[0]='Referer: '.$url.'left.asp';

curl\_setopt($ch, CURLOPT\_HTTPHEADER, $headers);

$args['method'] ='POST';

$args['post\_data'] ='xh='.'2015141462109'.'&xm='.$password.'&Submit=%CC%E1%BD%BB';

curl\_fetch($ch,$args);

}

function logout($ch) {

$url ='http://pead.scu.edu.cn/jncx/out.asp';

curl\_setopt($ch, CURLOPT\_URL, $url);

curl\_fetch($ch);

}

function scan($ch, $headers) {

$url ='http://pead.scu.edu.cn/jncx/';

curl\_setopt($ch, CURLOPT\_URL, $url.'xzyy.asp');

$headers[0]='Referer: '.$url;

curl\_setopt($ch, CURLOPT\_HTTPHEADER, $headers);

$str =iconv('gbk', 'utf-8//ignore', curl\_fetch($ch));

preg\_match\_all('/(\d{3,4},){3}.{33}/', $str, $arr);

//write into maradb

$con=mysqli\_connect("127.0.0.1",'guest','123456','collage');

if($con->connect\_error) die('Could not connect: '.$con->connect\_error);

mysqli\_query($con, "set names utf8");

foreach( $arr[0] as $value )

if( $con->query("select \* from pe\_scan where post='$value'")->num\_rows==0 )

$con->query("insert into pe\_scan(time, post) values('".date("H:i a")."','$value')");

$con->close();

}

function run() {

$ch =curl\_init();

curl\_setopt ($ch, CURLOPT\_RETURNTRANSFER, TRUE);

curl\_setopt($ch, CURLOPT\_COOKIEFILE, null);

$headers=array(

'Referer: http://pead.scu.edu.cn/stu/',

'User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:45.0) Gecko/20100101 Firefox/45.0'

);

$password ='ICL1LA';

//$password='RJ5NXC';2015141462163

while(TRUE) {

login($ch, $headers,$password);

scan($ch, $headers);

logout($ch);

sleep(5\*60);

}

login($ch, $headers,$password);

scan($ch, $headers);

logout($ch);

curl\_close($ch);

}

set\_time\_limit(0); // 取消脚本运行时间的超时上限

error\_reporting(~E\_ALL);

date\_default\_timezone\_set('Asia/Shanghai');

run()

?>

之前一直会隔一段时间就退出，怕是因为我没有加上set\_time\_limit(0);

ignore\_user\_abort() 函数设置与客户机断开是否会终止脚本的执行。这个适用于通过web访问来做到在后台开启一直运行的进程。

<?php

$str ='123';

echo strlen($str);

echo substr($str,1);

echo substr($str,0,1); //和py里面的分片一样，[1]不会被输出

?>和C“兼容”的php

数据库操纵：(推荐使用php的类进行操作, 使用函数已经是过去式了)

php的Mysql/Mariadb数据库操作从mysql\_变成了mysqli\_开头。

学习网站: <http://php.net/manual/zh/book.mysqli.php>

不论是查询还是更新数据库操作, 都是使用query 方法.

不能将"关闭autocommit"作为缺省设置，否则在 innodb 表上执行的查询操作也将因为没有执行 commit 或者 rollback 而一直锁表（记录锁）！因此只能在需要时局部关闭 autocommit，并在操作完成后开启 autocommit！

出现这个问题就说明你在对NULL变量操作：

Notice: Trying to get property of non-object

获取ip：

1. HTTP\_CLIENT\_IP头是有的，只是未成标准，不一定服务器都实现了。

2. HTTP\_X\_FORWARDED\_FOR 是有标准定义，用来识别经过HTTP代理后的客户端IP地址，格式：clientip,proxy1,proxy2。详细解释见 http://zh.wikipedia.org/wiki/X-Forwarded-For

3. REMOTE\_ADDR 是可靠的， 它是最后一个跟你的服务器握手的IP，可能是用户的代理服务器，也可能是自己的反向代理。

关于伪造： HTTP\_\*头都很容易伪造。例如使用火狐插件伪造x-forwarded\_forIP为 8.8.8.8

分隔函数：

list($header,$body) = explode("\r\n\r\n", $html)；

<?php

$str = 'one,two,three,four';

print\_r(explode(',', $str));

// 零 limit

print\_r(explode(',',$str,0));

// 正的 limit

print\_r(explode(',',$str,2));

// 负的 limit

print\_r(explode(',',$str,-1));

?>

Array

(

[0] => one

[1] => two

[2] => three

[3] => four

)

Array

(

[0] => one,two,three,four

)

Array

(

[0] => one

[1] => two,three,four

)

Array

(

[0] => one

[1] => two

[2] => three

)

explode第三个参数limit：

大于 0 - 返回包含最多 limit 个元素的数组

小于 0 - 返回包含除了最后的 -limit 个元素以外的所有元素的数组

0 - 返回包含一个元素的数组

退出函数：

exit(status)

<=>die(status) status是一个描述字符串或者状态号，如果是字符串会输出到stdout。

退出状态的值在 0 至 254 之间。退出状态 255 由 PHP 保留，不会被使用。

删除数组中的某一个元素：

<?php

$arr = array('apple','banana','cat','dog');

unset($arr[2]);

print\_r($arr);

?>

程序运行结果：

复制代码 代码如下:

Array ( [0] => apple [1] => banana [3] => dog )

但是这种方法的最大缺点是没有重建数组索引，就是说，数组的第三个元素没了。

经过查资料后，原来PHP提供了这个功能，只不过很间接。这个函数是array\_splice()。

为了使用方便，我封装成了一个函数，方便大家使用：

复制代码 代码如下:

<?php

function array\_remove(&$arr, $offset)

{

array\_splice($arr, $offset, 1);

}

$arr = array('apple','banana','cat','dog');

array\_remove($arr, 2);

print\_r($arr);

?>

文件操作：

删除文件：bool unlink ( string $filename [, resource $context ] )

删除 filename。和 Unix C 的 unlink() 函数相似。 发生错误时会产生一个 E\_WARNING 级别的错误。

判断常量是否存在：

if (defined('CONST\_NAME')) {

//do something

}

变量检测则是使用isset，注意变量未声明或声明时赋值为NULL，isset均返回FALSE，如：

if (isset($var\_name)) {

//do something

}

函数检测用function\_exists，注意待检测的函数名也需要使用引号，如：

if (function\_exists('fun\_name')) {

fun\_name();

}

<?php

$a =test();

if (isset($a)) {

echo "isset";

}else {

echo "isn't set";

}

function test() {

return ;

}

?>//isn't set

<?php

$a =test();

if (isset($a)): #虽然有脚本形式的语法但是if后面还是要圆括号的

echo "isset";

elseif(!isset($a)): //或者直接else

echo "isn't set";

endif;

function test() {

return ;

}

?>//isn't set

<?php

if($test=test())

echo 1;

else

echo 0;

function test(){

return;

}

?>//0

$ids =[];

if(isset($ids)):

echo '1';

endif;

?>//1

打开文本处于追加模式对于写日志挺有用的。

echo false;不会输出0，但是echo true;会输出1.

urlencode($str)

urldecode($str)

curl\_escape($ch, $str) 这是POST里面，不同于url，加号也是要被编码的。

<?php

$ch =curl\_init();

echo curl\_escape($ch, "+");

echo urldecode(curl\_escape($ch,"+"));

?>//%2B +

urldecode($str)可以解码curl\_escape得到的编码

调试的时候把死循环的while(1)改为if(1)或者加一个break;后者更好，因为你可能使用了break;和continue;在某些位置。

echo 一个数组会输出“Array”。

PHP Warning: sleep(): Number of seconds must be greater than or equal to 0

<?php

print\_r(test()[0]); #a feasible grammer

function test() {

return Array(Array(1,2,3),Array(1,2,3));

}

?>

如果include()不是放在开头，我放即将使用的前面结果导致后面的代码都无效。放过去，再放回来就好了，简直傻逼。

被include的文件如果不是<?php ?>包裹就会被当做文本文件，然后在执行的时候被显示出来。

数组的写法：

$result =Array

(

'valid' => Array

(

0 => Array(),

1 => Array()

),

'all' => Array

(

0 => Array

(

0 => '2017.04.13,星期四,上午,江安体测点',

1=> '2017.04.13,星期四,下午,江安体测点',

2 => '2017.04.14,星期五,上午,江安体测点',

3 =>' 2017.04.14,星期五,下午,江安体测点'

)

)

);

PHP Warning: mysqli\_connect(): (HY000/2002): No such file or directory in /path/PE/test.php on line 2

1.使用TCP/IP代替Unix socket。即在连接的时候将localhost换成127.0.0.1。

2.修改MySQL的配置文件my.cnf，指定mysql.socket的位置：

/var/lib/mysql/mysql.sock (你的mysql.socket路径)。

3.直接在php建立连接的时候指定my.socket的位置（官方文档：mysqli\_connect）。比如：

$db = new MySQLi('localhost', 'root', 'root', 'my\_db', '3306', '/var/run/mysqld/mysqld.sock')

php 默认相对路径都是以，被访问页面所在路径为准的。无论一个入口页面，里面包含多少文件，相对路径，都是以这个页面为准。

php的类型查看函数是gettype。typeof是js，typedef是C语言。

重定向：

<?php

header("Location : http://xx")

?>

现在 PHP也能支持 动态水印视频。我见过有人使用ffmpeg配合php使用。

php里面也是有fprintf、printf函数

php认为$\_GET到的内容都是字符串，虽然它是弱类型，之后怎么解释（解释成什么类型），还要看你怎么使用。

'0'是FALSE。

''不等于'0'，但是他们都可以和0相等。

'3'等于'3.0' ，但是不完全等于

md5('1')===md5(1) 会自动转化为字符串类型，对于md5，如果指明第二个参数是TRUE，那么就就会返回16位的md5，默认FLASE是32位。

PHP在处理哈希字符串时，会利用”!=”或”==”来对哈希值进行比较，它把每一个以”0E”开头的哈希值都解释为0，所以如果两个不同的密码经过哈希以后，其哈希值都是以”0E”开头的，那么PHP将会认为他们相同，都是0。但是===还是不一样的，估计和涉及的转换有关系。

http://119.29.16.200:5010/index.php?index=a1.html 很明显可能存在一个php代码执行漏洞，如果是直接include(HTTP方法中的file\_path)来返回文件的话。破解的话就是http://119.29.16.200:5010/index.php?index=然后在post数据中加上php的执行就好。可以用system("")来执行系统命令。也可以是自己php脚本直接上。事实上，在include函数的使用上，经常会造成任意文件读取漏洞，而file\_get\_contents()和file\_put\_contents()这样函数下，常常会构成getshell等更严重的漏洞。

php输入流php://input可以读取没有处理过的POST数据。相较于$HTTP\_RAW\_POST\_DATA（$\_POST这个内置变量）而言，它给内存带来的压力较小，并且不需要特殊的php.ini设置。php://input不能用于enctype=multipart/form-data

$raw\_post\_data = file\_get\_contents('php://input', 'r');

一下代码存在漏洞：

<?php

$ha = $\_GET['ha'];

$s1 = $\_GET['1s'];

echo gettype($\_GET['ha']);

echo gettype($\_GET['1s']).'<br>';

if ($ha and $s1) {

if ($ha == $s1) {

echo 'string is equal';

}else if (md5($ha) === md5($s1)) {

echo '+1s<br>';

echo 'hacked';

}else {

echo 'md5 not euqal';

}

}

?>

1.ha和1s可能是数组然后就导致md5返回FALSE

http://118.89.107.162/test.php?ha[0]=1&1s[0]=2

http://118.89.107.162/test.php?ha[]=1&1s[]=2

2.md5已经不安全了，对于本题目是可以获取flag的。尤其是php里面的md5实现还有问题：

MD5运算结果0e开头的全部相等(==判断)，比如240610708 和 QNKCDZO。似乎说这个科学计数法有关。

int ereg(string pattern, string string, array [regs]);这个函数接受的也是字符串，但是可以使用数组进行绕过，%00可以截断eregi。本函数以 pattern 的规则来解析比对字符串 string。比对结果返回的值放在数组参数 regs 之中，regs[0] 内容就是原字符串 string、regs[1] 为第一个合乎规则的字符串、regs[2] 就是第二个合乎规则的字符串，余类推。

int strcmp ( string $str1 , string $str2 )、strpos

if (isset($\_GET['password'])) {

if (strcmp($\_GET['password'], $flag) == 0)

die('Flag: '.$flag);

else

print 'Invalid password';

}

sha1()也是一样的

接收参数中不能出现某一字符，但下面又必须使用可以 php://伪协议绕过

php伪协议：

php://input标准输入

php://outut标准输出

php://fd 允许直接访问指定的文件描述符

php://filter

resource=<要过滤的数据流> 这个参数是必须的。它指定了你要筛选过滤的数据流。

read=<读链的筛选列表> 该参数可选。可以设定一个或多个过滤器名称，以管道符（|）分隔。

write=<写链的筛选列表> 该参数可选。可以设定一个或多个过滤器名称，以管道符（|）分隔。

任何没有以 read= 或 write= 作前缀 的筛选器列表会视情况应用于读或写链。

使用举例：php://filter/read=convert.base64-encode/resource=upload.php

数字的转化：

不允许出现数字1和9却要进行数字的比较，所以就是用0xdeadc0de，考虑进制的转化就好

function noother\_says\_correct($number)

{

$one = ord('1');

$nine = ord('9');

// Check all the input characters!

for ($i = 0; $i < strlen($number); $i++)

{

// Disallow all the digits!

$digit = ord($number{$i});

if ( ($digit >= $one) && ($digit <= $nine) )

{

// Aha, digit not allowed!

return false;

}

}

return $number == "3735929054";

}

如果之前先进行了字符串的比较=="42"，42.00e+0000000000，10的00000次方。。等于42.000000.... 结果其实就是42。

is\_numeric 空格、\t、\n、\r、\v、\f、+、-能够出现在参数开头，“点”能够在参数任何位置，E、e只能出现在参数中间。绕过 采用16进制编码可以绕过，不过之后的php版本已经修复。

stripslashes 函数会删除反斜杠

mysql\_real\_escape\_string 是一个可以通过编码绕过的函数

PHP弱类型的一个特性，当一个整形和一个其他类型行比较的时候，会先把其他类型intval再比。intval('ctf')是0。

$user[1]=1445502008; 如果直接这样子那么数组的[0]是未设置的。

PHP Version: 5.5.26 索引里面是4字节也就是32位。

var\_dump([0 => 0] === [0x100000000 => 0]); // 7.0.16已经没有了

magic\_quotes\_gpc=on会将一些字符比如说单引号进行转义

拼接变量的地方就要注意漏洞

$usrname = $\_POST [ 'username' ];

$passwd = $\_POST [ 'password' ];

这是怎么实现的呢？这后面的东西就是Hash Map啊，然后被hash collision DOS了

有的时候为了防止全局变量覆盖，会出现这样的代码，unset释放变量

//if register globals = on, undo var overwrites

foreach(array('\_GET','\_POST','\_COOKIE') as $method){

foreach($$method as $key=>$value){

unset($$key);

}

} 从前还有个全局变量$\_REQUEST

eval('$str="'.addslashes($str).'";'); ${${这里执行代码}}可以执行自己制定的代码

<?php 不会有任何输出，但是如果判断是true会输出1.

echo 0==2;

?>PS：1=='1'

mysql中支持的精度比php小不少，所以index.php?id=1.0000000000001可以绕过以下代码

$id = @(float)$\_GET['id'];

$secretId = 1;

if($id == $secretId){

echo 'Invalid id ('.$id.').';

}

else{

$query = 'SELECT \* FROM users WHERE id = \''.$id.'\';';

$result = mysql\_query($query);

$row = mysql\_fetch\_assoc($result);

echo "id: ".$row['id']."</br>";

echo "name:".$row['name']."</br>";

}

序列化:

**serialize** — 产生一个可存储的值的表示（这个值可以是一个函数的返回值）

**unserialize** — 从已存储的表示中创建 PHP 的值

$arr = array(“测试1″,”测试2″,”测试3″);//数组  
$sarr = serialize($arr);//产生一个可存储的值(用于存储)：a:3:{i:0;s:5:”测试1″;i:1;s:5:”测试2″;i:2;s:5:”测试3″;}

写题目的时候通过：

$a=new just4fun();

$a->enter="abc";

$a->secret=&$a->enter;

print unserialize(serialize($a));

'a:1:{i:1;C:11:"ArrayObject":37:{x:i:0;a:2:{i:1;R:4;i:2;r:1;};m:a:0:{}}}'

**array**(1) { // outer\_array

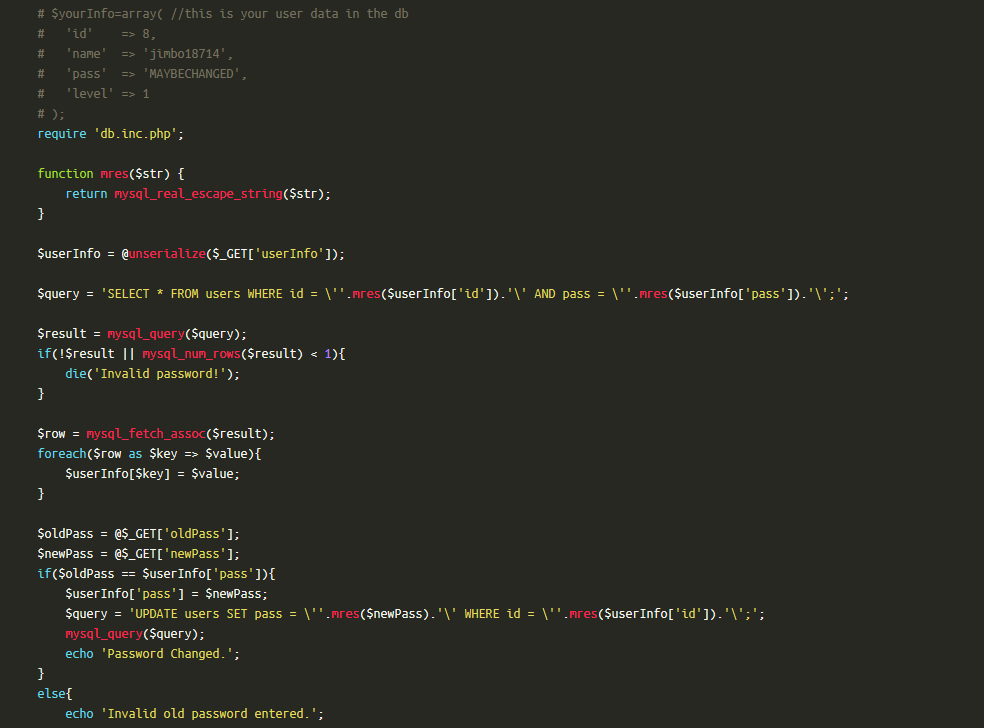
[1]=> object(ArrayObject)#1 (1) {

["storage":"ArrayObject":**private**]=>

**array**(2) { // inner\_array [1]=> // Reference to inner\_array

[2]=> // Reference to outer\_array } } }

序列号与数据库操作：



关键：糟糕的序列话、以及一行在实际生产中不会出现的代码$userInfo['pass'] = $newPass;。

解法：提交, index.php?userInfo=a:2:{s:2:"id";i:8;s:4:"pass";s:12:"MAYBECHANGED";}&oldPass=MAYBECHANGED&newPass=8，目的是将id为8的用户的密码修改为8

第二次提交，index.php?userInfo=s:1:"8";&oldPass=8&newPass=1,这样序列化$userInfo得到的就是字符串‘8',即$userInfo = ‘8' ，**这样数据库查询验证就可以通过**。之后的if验证也可以通过，通过这行代码$userInfo['pass'] = $newPass;,由于$newpass的值为1，那么上述代码变为了$userInfo['pass'] = 1; ,$userInfo由于一个字符串类型，最后得到的是$userInfo='1' ,最后就可以更新id为1的用户的密码了。

Xdebug是一个开源的PHP程序调试工具,可以使用它来调试、跟踪及分析程序运行状态。不使用IDE的话。

PHP垃圾回收机制：

学习网址：http://www.freebuf.com/vuls/122938.html

变量是zval保存的，类似于C++中的智能指针，记录了值以及引用个数，还需要一个变量判断当前变量是不是引用类型is\_ref，虽然我觉得不应该有。

<?

$a=[];

$b=[];

$a[0] =$b; 这时候b的refcount增加 数组的refcount就是元素个数

$b[0] =$a;

?> unset的时候先将refcount减一，如果变成0，才会回收

unset($a) unset($b) 互相依赖，两个之前都已经将refcount变成2，减了1以后还是存在，却不能再被访问使用，这就是PHP5.3 php新垃圾回收机制的开始。之前的回收机制仅仅使用了refcount计数。

<?php

$a = **array**('one');

$a[] = &$a; **unset**($a);

?>也会造成内存泄露Memory Leak

变量按照链表的实行连接到root节点，达到一定量（缓冲区塞满）自动触发回收机制or自己调用gc\_collect\_cycles()函数。通过深度优先算法， 对每一个节点（除了初节点）所包含的zval的refcount减一，有一个标志位记录是否某一个zval已经经历过减一操作。$A=123是最简单的节点，只有一个root于是不会经历深度遍历减一操作。如果减一以后还是大于0的，那么认为这个变量不是垃圾，最后还要加一加回去，并置为正常的状态。

垃圾回收机制相关函数：

gc\_mark\_roots() 扫描root开始的节点并执行zval\_mark\_grey 进入减一的灰色状态2。

gc\_scan\_roots() 扫描root执行zval\_scan。zval\_scan() 检查zval的refcoutn是否已经到0了

gc\_collect\_roots() 垃圾回收

gc\_collec t\_cycle() 调用垃圾会收取

由于php垃圾回收机制实现上面的问题，导致一个特殊变量不该销毁的却被销毁，但是变量还是指向那个位置，于是内存重用….. CVE-2016-5771。这个漏洞传闻只能使用序列化来构造特殊的漏洞利用数组，直接正向构造说因为存在内部处理而失败。内存回收算法的实现存在问题是内部节点指向了外部节点，当时却会被当成一个内部子节点继续深度下去减一。

配合unserialize的内部实现，传入一个地址，能够将一个变量指向任意的内存。实际利用漏洞的时候，还会受到PHP内存管理块结构的限制。

在实际环境中利用此漏洞要解决几个问题。首先是漏洞环境一般不会手工调用gc\_collect\_cycles()，所以就需要在单一unserialize()调用的情况下完成垃圾回收。在PHP中默认的gc\_root\_buffer缓冲区大小是100000，所以只要构造一个超过这个数量元素的数组就可以自动触发gc\_collect\_cycles()。

通过内存释放和UAF，配合其余PHP内部的实现jmp\_buf以及unserialize实现对内存读写、程序流程的改变。最后绕过DEP等等借助eval函数的实现执行自己的PHP shell。

测试代码：在释放以后的变量内存上建立了新变量  
<?php

$serialized\_string = 'a:1:{i:1;C:11:"ArrayObject":37:{x:i:0;a:2:{i:1;R:4;i:2;r:1;};m:a:0:{}}}';

$outer\_array = unserialize($serialized\_string);

gc\_collect\_cycles();

$filler1 = "aaaa"; $filler2 = "bbbb";

var\_dump($outer\_array); //输出bbbb UAF use after free

?>

服务器端的漏洞代码：

<?php

$a =unserialize($POST[‘in’]);

var\_dump($a);

?>



类似漏洞CVE-2016-5771

上面介绍的垃圾回收机制只考虑到了变量到了refcount=1的时候，并没有讲明如果变量之后要用到怎么办。因为按照垃圾回收机制，会对refcount=1的复杂变量减一变成0以后回收。

PHPShellCode：

首先介绍一下jmp\_bufjmp\_buf，是PHP在C语言层面异常处理的底层实现的要用到的数据结构，具体可以参见zend\_try的定义。它是通过调用setjmp/longjmp来实现的，而jmp\_buf用于保存恢复调用环境所需的寄存器信息jmp\_buf。这个结构保存在栈上面，我们只要覆盖了jmp\_buf结构中保存的返回地址RIP的值，那么就等于控制了程序执行流程。

在PHP下执行我们自己的代码首先需要对抗DEP，也就是需要ROP来绕过执行限制。但是在PHP下又有其特殊性比其他的应用漏洞利用起来要容易得多。因为PHP有自己的代码执行函数eval不需要我们去编写二进制shellcode。我们只需要获得eval的内部实现函数zend\_eval\_string的地址，并排列好栈布局把PHP代码放在后面，这样就可以直接利用zend\_eval\_string执行我们的PHP代码了。