PHP 脚本以 <?php 开始，以 ?> 结束。PHP 文件中代码和 html 可以参杂着写。
变量以 $ 符号开始，后面跟着变量的名称
echo - 可以输出一个或多个字符串

在 PHP 中，只有一个字符串运算符。并置运算符 (.) 用于把两个字符串值连接起来。

在 PHP 中，array() 函数用于创建数组：

PHP 表单处理有一点很重要的事情值得注意，当处理 HTML 表单时，PHP 能把来自 HTML 页面中的表单元素自动变成可供 PHP 脚本使用。 PHP 中的 $_GET 和 $_POST 变量用于检索表单中的信息，比如用户输入。

PHP $_REQUEST 变量
预定义的 $_REQUEST 变量包含了 $_GET、$_POST 和 $_COOKIE 的内容。
$_REQUEST 变量可用来收集通过 GET 和 POST 方法发送的表单数据。也就是说不管是使用 POST 还是使用 GET 方式提交的数据，都可以使用 $_REQUEST 来获取，甚至可以使用 $_REQUEST 来获取 COOKIE 的信息。
欢迎 <?php echo $_REQUEST["fname"]; ?>!
你的年龄是 <?php echo $_REQUEST["age"]; ?> 岁。

include （或 require）语句会获取指定文件中存在的所有文本/代码/标记，并复制到使用 include 语句的文件中。

include 和 require 语句是相同的，除了错误处理方面：

- require 会生成致命错误（E_COMPILE_ERROR）并停止脚本
- include 只生成警告（E_WARNING），并且脚本会继续

通过使用 PHP 的全局数组 $_FILES，你可以从客户计算机向远程服务器上传文件。

第一个参数是表单的 input name，第二个下标可以是 "name", "type", "size", "tmp_name" 或 "error"。就像这样：

- $_FILES["file"]["name"] - 被上传文件的名称
- $_FILES["file"]["type"] - 被上传文件的类型
- $_FILES["file"]["size"] - 被上传文件的大小，以字节计
- $_FILES["file"]["tmp_name"] - 存储在服务器的文件的临时副本的名称
- $_FILES["file"]["error"] - 由文件上传导致的错误代码

这是一种非常简单文件上传方式。基于安全方面的考虑，您应当增加有关什么用户有权上传文件的限制。

PHP 中的超全局数组包括以下 9 个：

$GLOBALS：包含一个引用指向每个当前脚本的全局变量范围内有效的变量。该数组的键名为全局变量的名称。

$_SERVER：显示或获取服务器的相关信息；
$_GET：把数据通过地址栏传递到服务器，这是方式必须是$_GET 方式传递；
$_POST：通过表单发送的数据必须是 POST 方式；
$_REQUEST：包含上面两种数据传递方式（POST 和 GET）；
$_COOKIE：处理客户端的会话数据；
$_SESSION：处理服务器端的会话数据；
$_FILES：上传文件需要用到的数组；
$_ENV： 执行环境提交至脚本的变量

eval() 函数把字符串按照 PHP 代码来计算。

该字符串必须是合法的 PHP 代码，且必须以分号结尾。
assert 把整个字符串参数当 php 代码执行，eval 把合法的 php 代码执行

at 符号（@）在 PHP 中用作错误控制操作符。当表达式附加@符号时，将忽略该表达式可能生成的错误消息。

isset() 函数用于检测变量是否已设置并且非 NULL。
die() 函数输出一条消息，并退出当前脚本

DVWA

1.sql 注入
中级 Medium
数字型，利用 mysql_real_escape_string 函数对特殊符号\x00,\n,\r,,',",\x1a 进行转义，并且出现了下拉菜单使得用户无法控制输入的变量。所以解下来要做代理服务器设置并使用 burp suite 抓包工具。

```
mysqli_real_escape_string(connection,escapestring);//第二个参数为需转义字符串，第一个参数为数据库连接
```

高级（High）

只是在后面加了 LIMIT 1 且 为字符型注入，那只需要在输入语句最后#注释掉 LIMIT 1 即可，其余步骤基本与 low 级别一样。还有不同呢就是在前端页面上，没有输入框了，点击会跳转页面，基本杜绝了自动化的 SQL 注入。High 级别的查询提交页面与查询结果显示页面不是同一个，也没有执行 302 跳转，这样做的目的是为了防止一般的 sqlmap 注入，因为 sqlmap 在注入过程中，无法在查询提交页面上获取查询的结果，没有了反馈，也就没办法进一步注入。但是源码中没有对参数进行防御，还是可以通过 burp suite 抓包，修改参数，获取数据库信息。

Impossible 级别

采用了 PDO(PDO 可以被认作是一种通过编译 SQL 语句模板来运行 SQL 语句的机制)技术，划清了代码与数据的界限，有效防御 SQL 注入，同时只有返回的查询结果数量为一时，才会成功输出

```
//这里必须有第三个参数，否则默认为字符类型，查询结果为空
$stmt->bindParam(':start', $start, PDO::PARAM_INT);
```

2.文件上传

Low：basename() 函数：返回路径中的文件名部分。服务器对上传文件的类型、内容没有做任何的检查、过滤，存在明显的文件上传漏洞。生成上传路径后，服务器会检查是否上传成功并返回相应提示信息

medium 级别源码分析

从源码中很明显可以看出，函数对上传文件的类型和大小作了限制，上传文件类型必须是 image/jpeg 或者 image/png，且上传文件大小不能超过 100000B（大约等于 97.6KB）

方法一：抓包修改 content-type

方法二：在文件名后放一个空字节（p153 00 绕过）

High

strrpos() 函数查找字符串在另一字符串中最后一次出现的位置；

仅仅文件后缀名为图片格式的不行，文件内容必须也是图片格式的；制作图片木马。利用 DVWA 的文件包含漏洞，让我们的图片格式的一句话木马以 php 格式运行；

也可使用 00 截断

Impossible 级别

对上传的文件进行了重命名（为 md5 值，导致 00 截断无法绕过过滤规则），并加入 Anti-CSRF token 防护 CSRF 攻击，同时对文件的内容作了严格的检查，导致攻击者无法上传含有恶意脚本的文件。

补充 00 截断：

0x00 是十六进制表示方法，是 ascii 码为 0 的字符，在有些函数处理时，会把这个字符当做结束符。这个可以用在对文件类型名的绕过上。而%00 是 URL 解码之前的字符，它被解码成 16 进制 ASCII 码之后实际上也是 0x00，所以它们最终都对应的是空字符

3.文件包含

High

fnmatch() 函数根据指定的模式来匹配文件名或字符串。

```
fnmatch(pattern,string,flags)
```

要求 page 参数的开头必须是 file，服务器才会去包含相应的文件。

可以利用 file 协议绕过防护策略，用浏览器打开一个本地文件时，用的就是 file 协议。

Impossible

从源码中可以很清楚的看到，page 参数必须为“include.php”、“file1.php”、“file2.php”、“file3.php”之一，它实质上是使用了白名单机制进行防护，以非常简洁明了的方式说明具体哪些文件被允许，彻底杜绝了文件包含漏洞。

4.存储型 XSS

medium

比起 LOW 等级的代码，MEDIUM 等级主要是有这三个高亮的地方的修改：

（1）对 message 的内容中的预定义字符之前添加反斜杠(addslashes 函数)。预定义字符包括 '、"、\、NULL（其实感觉这步挺多余的，因为默认地，PHP 对所有的 GET、POST 和 COOKIE 数据自动运行 addslashes()，所以不应对已转义过的字符串使用 addslashes()，会导致双层转义，详细可见 https://www.w3school.com.cn/php/func_string_addslashes.asp）然后用 strip_tags() 函数剥去字符串中的 HTML、XML 以及 PHP 的标签。
（2）将 message 中的预定义字符转换为 html 实体（htmlspecialchars 函数）。预定义的字符包含&、<、>、'、"
（3）将 name 中的<script>删除（使用 str_replace 函数进行字符串替换，由于该函数是区分参数大小写的，所以也可以采用大写绕过）
<scr<script>ipt>alert(/xss/)</script>
因此 name 这里至少可以用三种方法绕过：（1）**双写**<script>绕过，如本文上述示例（2）**大写**绕过，比如<sCript>（3）换成不带<script>标签的 payload

High

可以看到，high 级别只是在 medium 级别上，name 参数用了正则表达式进行过滤

```
// Sanitize message input
$message = strip_tags( addslashes( $message ) );
$message = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_es
$message = htmlspecialchars( $message );

                                    使用了正则对<script进行了无限次过滤，所以不能双写绕过
// Sanitize name input
$name = preg_replace( pattern: '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', replacement: '', $name );
$name = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape

// Update database
$query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
```


HTML 的 < img > 标签定义 HTML 页面中的图像，该标签支持 onerror 事件，在装载文档或图像的过程中如果发生了错误就会触发。使用这些内容构造出 payload 如下，因为我们没有图片可供载入，因此会出错从而触发 onerror 事件。
<img src = 1 onerror = alert('xss')>

仍然是抓包之后改参数，放包后成功弹出页面。

Impossible

htmlspecialchars() 函数把一些预定义的字符转换为 HTML 实体
源码在 high 级别的基础上对 name 参数也进行了更严格的过滤，导致 name 参数也无法进行 JS 脚本注入。同时加入 Anti-CSRF token 防护 CSRF 攻击，进一步提高安全性。