

PHP代码审计入门指南

这本指南包含了我在学习PHP代码审计过程中整理出的一些技巧和对漏洞的一些理解

[无标题文档](#)

[PHP代码审计入门指南](#)

[序言](#)

[PHP审计基础](#)

[✕ 工具准备](#)

[VS CODE 常用快捷键](#)

[PHP代码审计思路](#)

[🔪 PHP用户可控输入速查表](#)

[🎯 PHP敏感函数速查表](#)

[🔧 PHP原生过滤方法](#)

[PHP动态调试-Xdebug安装配置](#)

[PHP常见漏洞](#)

[命令注入](#)

[代码注入](#)

[文件包含](#)

[SQL注入](#)

[文件操作](#)

[XSS](#)

[SSRF](#)

[CSRF](#)

[XXE](#)

[反序列化](#)

[LDAP注入](#)

[其他漏洞](#)

[PHP常见框架](#)

[Thinkphp](#)

[Laravel](#)

[Codeigniter](#)

[Yii](#)

[Cakephp](#)

PHP审计实例

-

PHP特性利用

[无标题](#)

PHP审计扩展

[PHP商业源码提取](#)

[浅谈PHP源代码保护方案&受保护PHP代码の解密还原](#)

附录

[changelog](#)

[PHP弱类型](#)

[扩展阅读](#)

 [总结](#)

 [参考](#)

PHP代码审计入门指南

作者

白帽酱 (橙子酱)(i@rce.moe)

简介

这本指南包含了我在学习PHP代码审计过程中整理出的一些技巧和对漏洞的一些理解

这本指南仍在在编写完善中

如果有遗漏或者是错误的地方 欢迎大家指出

序言

PHP代码审计

PHP代码审计是指对PHP程序源代码进行系统性的检查,它的目的是为了发现PHP程序存在的一些漏洞或者逻辑问题,避免程序缺陷被非法利用从而带来不必要的安全风险.

作者的话

你在这篇指南中将要学习到的不止是php审计!

本文除了审计代码点,另外还讲述了漏洞产生原理和防护方法. 在了解漏洞产生原理后你会对漏洞利用有更加深刻的理解和掌握. WEB代码审计不同语言之间其实有很多相通的东西呢,你会发现在学习php审计后学习其他代码审计速度会更快.相信你在学习完这本指南之后会让你对WEB安全有更加深刻的认识.

✂ 工具准备

工欲善其事,必先利其器. 如果想要高效快速地入门PHP代码审计. 那么你就需要一些强大的工具辅助审计.

✂ 工具准备

本指南主要面向手工审计方向 自动化审计工具不考虑在内

Visual Studio Code

简介

Visual Studio Code 是一个开源免费的代码编辑器 本指南中使用该编辑器作为示例

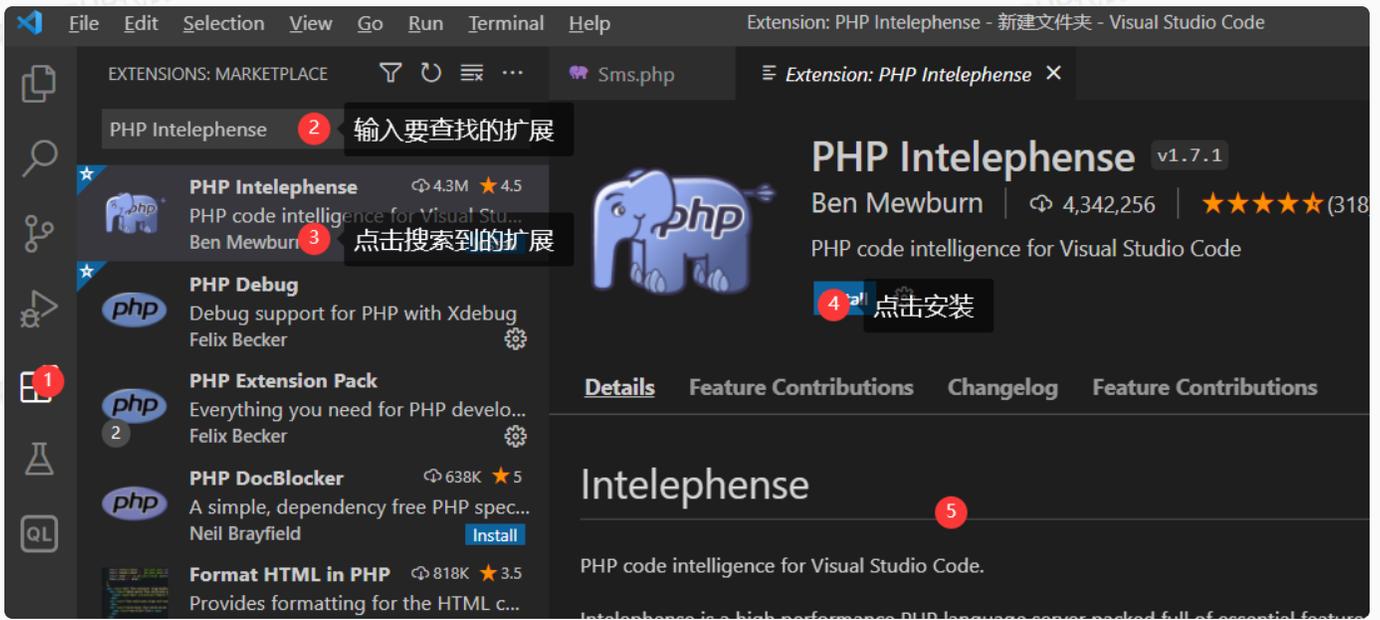
安装

前往微软官网下载安装

<https://code.visualstudio.com/>

插件

VS CODE 扩展商店拥有大量实用的扩展插件. 本文使用PHP Intelephense扩展用于辅助审计



点击左侧工具条扩展按钮安装图示进行安装

VS CODE 常用快捷键

常用快捷键

使用快捷键可以大幅度地提升审计效率

关闭当前窗口: Ctrl+W

文件之间切换: Ctrl+T_a_b

移动到行首: Home

移动到行尾: End

移动到文件开头: Ctrl+Home

移动到文件结尾: Ctrl+End

查找/转到定义: F12

查找/转到引用: Shift+F12

代码格式化: Shift+Alt+F

查找: Ctrl+F

F9 打断点

F5 开始调试

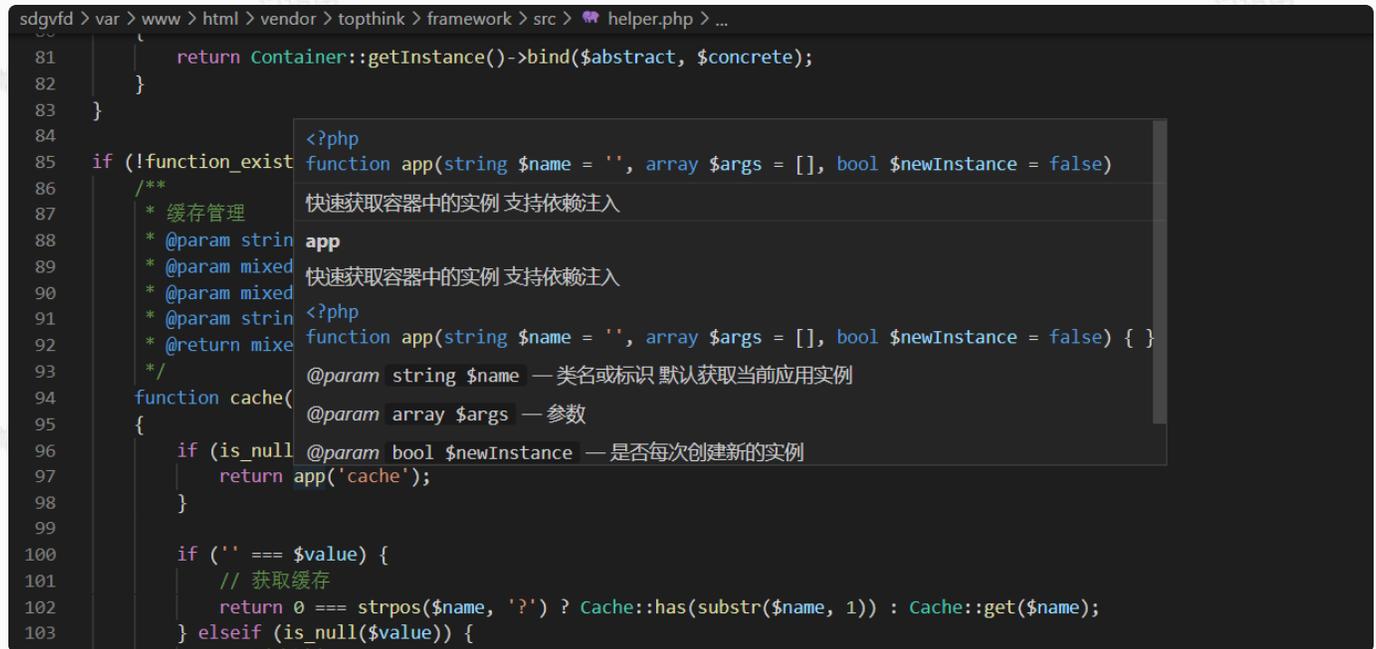
在项目文件中搜索: Ctrl+Shift+F

插件辅助功能

前文中安装的插件提供了一些强大的功能 本书主要使用该插件的函数跳转追踪功能

该插件可以帮助你快速查找代码引用的函数/类/变量/常量 以及被引用的代码位置

将鼠标指针指向要查找的内容可以预览定义



```
sdgvyd > var > www > html > vendor > tophink > framework > src > helper.php > ...
81     return Container::getInstance()->bind($abstract, $concrete);
82 }
83 }
84
85 if (!function_exists('app')) {
86     /**
87      * 缓存管理
88      * @param string $name
89      * @param mixed $args
90      * @param mixed $newInstance
91      * @param string $cacheKey
92      * @return mixed
93      */
94     function cache(
95         @param string $name — 类名或标识 默认获取当前应用实例
96         @param array $args — 参数
97         @param bool $newInstance — 是否每次创建新的实例
98     ) {
99
100         if ('' === $value) {
101             // 获取缓存
102             return 0 === strpos($name, '?') ? Cache::has(substr($name, 1)) : Cache::get($name);
103         } elseif (is_null($value)) {
```

按住ctrl 点击被引用对象可以快速跳转定义位置(或把光标放在查找位置 点击F12)

按住ctrl 点击定义可以快速查找项目中所有的引用位置(或把光标放在查找位置 点击Shift+F12)

这个功能在有复杂的文件包含时(如CMS框架)可以快速在众多文件中找到定义位置

PHP代码审计思路

这里列出几个审计思路 除了PHP外其他的大部分web开发语言也可按照这个思路审计

PHP代码审计思路

敏感函数方法回溯(反向审计)

查找项目中的敏感函数方法 回溯传入的参数 判断用户是否可控 是否得到有效的过滤

- 缺点:很难发现越权等逻辑漏洞

用户可控参数追踪(正向审计)

查找项目中的用户输入 追踪用户输入 判断是否得到有效的过滤/调用敏感函数/存在逻辑问题

关键业务功能分析(功能审计)

专门审计易出现漏洞的关键功能点

如 头像上传 系统登陆 文件下载 等功能

- 优点:可以快速完成审计 减少审计面 可发现越权等逻辑漏洞
- 缺点:审计不完全

完全审计源代码

- 优点:完全覆盖源代码 可发现一些特殊条件的漏洞
- 缺点:耗时长 容易遗漏

白帽酱 -PHP代码审计入门指南



PHP用户可控输入速查表

来自用户可控的输入 --永远不要相信用户输入

变量/常量/函数/等	描述
<code>\$_SERVER</code>	包含 服务器信息 环境变量 用户传入的http头和 uri路径等信息
<code>\$_GET \$HTTP_GET_VARS</code>	包含 用户传入的URL 参数
<code>\$_POST \$HTTP_POST_VARS</code>	包含 用户传入的POST BODY的参数 (当 HTTP头中Content-Type 值为 application/x-www-form-urlencoded 或 multipart/form-data时才会被传入)
<code>\$_FILES \$HTTP_POST_FILES</code>	包含 用户上传文件信息 文件内容 原文件名 临时文件名 大小 等信息
<code>\$_COOKIE \$HTTP_COOKIE_VARS</code>	包含 用户传入的HTTP头中的Cookies kv值
<code>\$_REQUEST</code>	同时包含 <code>\$_GET</code> <code>\$_POST</code> <code>\$_COOKIE</code>
<code>php://input \$HTTP_RAW_POST_DATA</code>	包含 用户POST请求中BODY 的完整数据 常见用法 <code>file_get_contents('php://input');</code>
<code>apache_request_headers()getallheaders()</code>	包含 用户传入的http头 (Apache ONLY)
下方为PHP框架常见输入	
<code>Request::instance()->get();input('get.');</code> input('变量类型.变量名/修饰符') 详见官方文档	获取用户传入的URL参数 可用过滤器和类型转换 THINKPHP框架5 例子:获取url参数中的id值 <code>Request::instance()->get('id');</code> 调用时如不传入参数默认获取全部 <code>Request::instance()->get();input('get.id');</code> 调用时如传入get.则获取全部 <code>input('get.');</code> <code>input('get.id/d');</code> // 强制变量转换为整型 <code>Request::instance()->get('name','',htmlspecialchars);</code> //过滤器 <code>input('get.name/s');</code> // 强制转换变量为字符串

	input('get.ids/a');// 强制变量转换为数组 默认为/s
Request::instance()->post();input('post.');	获取用户传入的POST参数 THINKPHP框架5 例子:获取post请求body中的name值 Request::instance()->post('name'); input('post.name'); 同get
Request::instance()->param();input('param.');	自动判断用户提交方法(POST GET PUT)获取参数 THINKPHP框架5 用法同get 除此之外 可直接调用input('');获取全部参数 或使用input('name');获取单个参数 注:input方法默认获取param
Request::instance()->request(); input('request.');	用法同get 获取\$_REQUEST 变量 THINKPHP框架5
Request::instance()->server(); input('server.');	用法同get 获取\$_SERVER 变量THINKPHP框架5
Request::instance()->cookie(); input('cookie.');	用法同get 获取\$_COOKIE 变量THINKPHP框架5
Request::instance()->header(); input('header.');	用法同get 获取用户传入的HTTP头THINKPHP框架5
Request::instance()->file();	用法同get 获取\$_FILES 变量THINKPHP框架5
I('变量类型.变量名/修饰符',['默认值'],['过滤方法或正则'])	获取变量 THINKPHP框架3.* 例子 I('get.id'); I('get.');
request();	实例化request对象THINKPHP框架5 例 \$req=request(); 相当于\$req=Request::instance() 这种使用方法比较常见 还可以获取用户传入的请求信息 可将前面的Request::instance()直接替换成request() 例 request()->post();
{Request.变量类型.变量名}	THINKPHP框架 在模板中获取参数
路由传入值 (Action参数绑定) namespace Home\Controller;	THINKPHP框架 (Action参数绑定) 通过路由传入

<pre>use Think\Controller; class BlogController extends Controller { public function read(\$id) { echo 'id='.\$id; } public function archive(\$year='2013',\$month='01') {echo 'year='.\$year.'&month='.\$month;}}</pre>	<p>/index.php/Home/Blog/read/id/5 /index.php/Home/Blog/archive/year/2013/month/11 ?c=Blog&a=read&id=5 ?c=Blog&a=archive&year=2013&month=11 来自 <https://www.kancloud.cn/manual/thinkphp/1715></p>
<p>Request::instance() 其他用户变量</p>	<p>https://www.kancloud.cn/manual/thinkphp5/158834 THINKPHP框架5 见官方文档 略</p>
<pre>\$this->input->post() \$this->input->get() \$this->input->cookie() \$this->input->server() \$this->input->user_agent(); \$this->input->request_headers(); \$this->input->get_request_header('some-header', TRUE); \$this->input->ip_address(); \$this->input->raw_input_stream(); \$this->input->input_stream('key'); (POST BODY)</pre>	<p>Codeigniter2/3框架 \$this->input->input_stream('key', TRUE); // XSS 过滤器开关 \$this->input->cookie('some_cookie'); 3.* 官方文档 2.*官方文档 文件上传 https://codeigniter.org.cn/userguide2/libraries/file_uploading.html https://codeigniter.org.cn/userguide3/libraries/file_uploading.html https://codeigniter.org.cn/userguide3/libraries/file_uploading.html?highlight=%E6%96%87%E4%BB%B6#id5</p>
<pre>\$request->getGet() \$request->getPost() \$request->getServer() \$request->getCookie() \$request->getPostGet()– 先 \$_POST, 后 \$_GET \$request->getPostGet()– 先 \$_GET, 后 \$_POST \$request->getJSON(); \$this->request->getFiles(); \$this->request->getFile('123'); 获取post body json数据 \$request->getRawInput() 同php://input</pre>	<p>Codeigniter4框架 https://codeigniter.org.cn/user_guide/incoming/incomingrequest.html?highlight=post#id4</p>
<pre>\$this->request->getQuery('page'); (GET参数) \$this->request->getQueryParams(); 全部get参数 \$this->request->getData('title'); (POST参数) \$this->request->getParsedBody(); 全部post参数 \$this->request-</pre>	<p>Cakephp 4.* 框架 文件上传 https://book.cakephp.org/4/en/controllers/request-response.html#file-uploads</p>

```

>getUploadedFile('attachment');
$jsonData=$this->request-
>input('json_decode'); $request-
>getUploadedFiles(); $data=$this->request-
>input('Cake\Utility\Xml::build',
['return'=>'domdocument']);
$userAgent=$this->request-
>getHeaderLine('User-Agent');// Get an array
of all values.$acceptHeader=$this->request-
>getHeader('Accept'); $this->request-
>getCookie('remember_me');

```

```

$request= Yii::$app->request;
$get= $request->get(); // 等价于: $get =
$GET;
$id= $request->get('id'); // 等价于: $id =
isset($GET['id']) ? $GET['id'] : null;
$id= $request->get('id', 1); // 等价于: $id =
isset($GET['id']) ? $GET['id'] : 1;
$post= $request->post(); // 等价于: $post =
$POST;
$name= $request->post('name'); // 等价于:
$name = isset($POST['name']) ?
$POST['name'] : null;
$name= $request->post('name', ''); // 等价于:
$name = isset($POST['name']) ?
$POST['name'] : '';
$request= Yii::$app->request; // 返回所有参
数
$params= $request->bodyParams; // 返回参
数 "id"
$params= $request->getBodyParam('id'); //
$headers 是一个 yii\web\HeaderCollection 对
象
$headers= Yii::$app->request->headers; //
返回 Accept header 值
$accept= $headers->get('Accept');
if($headers->has('User-Agent')) { / 这是一个

```

Yii 2.0框架

`User-Agent 头 /}`

`$userHost= Yii::$app->request->userHost;`

`$userIP= Yii::$app->request->userIP;`

`<https://www.yiichina.com/doc/guide/2.0/runtime-requests>`

`$name=$request->input('name');` 所有请求
`$name=$request->query('name');` get url参数
`$query=$request->query();` `$value=$request->cookie('name');` Use

`Illuminate\Support\Facades\Cookie;`

`$value=Cookie::get('name');` `$file=$request->file('photo');` `$file=$request->photo;`

`<https://learnku.com/docs/laravel/8.x/requests/9369>`

Laravel 框架

PHP敏感函数速查表

PHP一些容易出现安全问题的函数方法

命令执行

一些常见的可以执行系统命令的函数/语法

函数/语法	描述	例子
system	执行命令并输出结果	system('id');
exec	执行命令 只可获取最后一行结果	exec('id',\$a); print_r(\$a);
passthru	同 system	passthru('id');
shell_exec `(反引号)	执行命令并返回结果	<pre>\$a=shell_exec('id');print_r(\$a); \$a=`id`;print_r(\$a);</pre>
popen	执行命令并建立管道 返回一个指针 使用fread等函数操作指针进行读写	<pre>\$a=popen("id", "r"); echo fread(\$a, 2096);</pre>
proc_open	同 popen (进程控制功能更强)	见PHP手册
pcntl_exec	执行命令 只返回是否发生错误	pcntl_exec('id');

代码注入/文件包含

函数/语法结构	描述	例子
eval	将传入的参数内容作为PHP代码执行 eval 不是函数 是一种语法结构 不能当做函数动态调用	<code>eval('phpinfo();');</code>
assert	将传入的参数内容作为PHP代码执行 版本在PHP7以下是函数 PHP7及以上为语法结构	<code>assert('phpinfo();');</code>
preg_replace	当preg_replace使用/e修饰符且原字符串可控时时 有可能执行php代码	<code>echo preg_replace("/e", "\${PHPINFO()}", "123");</code>
call_user_func	把第一个参数作为回调函数调用 需要两个参数都完全可控才可利用 只能传入一个参数调用	<code>call_user_func('assert', 'phpinfo();');</code>
call_user_func_array	同call_user_func 可传入一个数组带入多个参数调用函数	<code>call_user_func_array('file_put_contents', ['1.txt', '6666']);</code>
create_function	根据传递的参数创建匿名函数，并为其返回唯一名称 利用需要第二个参数可控 且创建的函数被执行	<code>\$f = create_function("", 'system(\$_GET[123]);'); \$f();</code>
include	包含并运行指定文件 执行出错会抛出错误	<code>include 'vars.php';</code> (括号可有可无)
require	同include 执行出错会抛出警告	<code>require('somefile.php');</code> (括号可有可无)
require_once	同require 但会检查之前是否已经包含该文件 确保不重复包含	
include_once	同include 但会检查之前是否已经包含该文件 确保不重复包含	

SQL/LDAP注入

函数/方法	备注
mysql_query	
odbc_exec	
mysqli_query	
mysql_db_query	
mysql_unbuffered_query	
mysqli::query 用法 \$mysqli = new mysqli("localhost", "my_user", "my_password", "world"); \$mysqli->query();	
pg_query	
pg_query_params	
pg_send_query	
pg_send_query_params	
sqlsrv_query	
pdo::query \$pdo=new PDO("mysql:host=localhost;dbname=phpdem o","root","1234"); \$pdo->query(\$sql);;	PDO
SQLite3::query SQLite3::exec \$db = new SQLite3('mysqlitedb.db'); \$db-> query('SELECT bar FROM foo'); \$db-> exec('CREATE TABLE bar (bar STRING)');	
\$mongo = new MongoClient(); \$data = \$coll-> find(\$data);	https://wooyun.js.org/drops/Mongodb%E6%B3%A8%E5%85%A5%E6%94%BB%E5%87%BB.html
\$ld = ldap_connect("localhost");... \$lb = @ldap_bind(\$ld, "cn=test,dc=test,dc=com",	https://www.cnblogs.com/0nc3/p/12063436.html

"test");	
Db::query	Thinkphp
Db::execute	Thinkphp

文件读取/SSRF

函数	描述	例子
file_get_contents	读入文件返回字符串	<pre>echo file_get_contents("flag.txt"); echo file_get_contents("https://www.bilibili.com/");</pre>
curl_setopt curl_exec	Curl访问url获取信息	<pre>function curl(\$url){ \$ch = curl_init(); curl_setopt(\$ch, CURLOPT_URL, \$url); curl_exec(\$ch); curl_close(\$ch); } \$url = \$_GET['url']; curl(\$url); https://www.php.net/manual/zh/function.curl-exec.php</pre>
fsockopen	打开一个套接字连接(远程tcp/udp raw)	https://www.php.net/manual/zh/function.fsockopen.php
readfile	读取一个文件，并写入到输出缓冲	同file_get_contents
fopen/fread/fgets/fgetss/fgetc/fgetcsv/fpassthru/fscanf	打开文件或者 URL 读取文件流	<pre>\$file = fopen("test.txt","r"); echo fread(\$file,"1234"); fclose(\$file);</pre>
file	把整个文件读入一个数组中	<pre>echo implode(", file('https://www.bilibili.com/'));</pre>
highlight_file/show_source	语法高亮一个文件	highlight_file("1.php");
parse_ini_file	读取并解析一个ini配置文件	print_r(parse_ini_file('1.ini'));
simplexml_load_file	读取文件作为XML文档解析	

文件上传/写入/其他

函数	描述	例子
file_put_contents	将一个字符串写入文件	file_put_contents("1.txt","6666");
move_uploaded_file	将上传的临时文件移动到新的位置	move_uploaded_file(\$_FILES["pictures"]["tmp_name"],"1.php")
rename	重命名文件/目录	rename(\$oldname,\$newname);
rmdir	删除目录	
mkdir	创建目录	
unlink	删除文件	
copy	复制文件	copy(\$file, \$newfile);
fopen/fputs/fwrite	打开文件或者 URL	https://www.php.net/manual/zh/function.fwrite.php
link	创建文件硬链接	link(\$target, \$link);
symlink	创建符号链接(软链接)	symlink(\$target, \$link);
tmpfile	创建一个临时文件 (在临时目录存放 随机文件名 返回句柄)	\$temp = tmpfile(); fwrite(\$temp, "123456"); fclose(\$temp);
request()->file()->move() request()->file()->file()	Thinkphp 文件上传	\$file = request()->file(\$name); \$file->move(\$filepath);
extractTo	解压ZIP到目录	
DOMDocument loadXML simplexml_import_dom	加载解析XML 有可能存在 XXEE 漏洞 file_get_contents 获取客户端输入内容 new DOMDocument()初始化XML 解析器 loadXML(\$xmlfile)加载客户端输入的XML内容 simplexml_import_dom(\$dom) 获取XML 文档节点 加载成功	<?php \$xmlfile=file_get_contents('php://input'); \$dom=new DOMDocument(); \$dom->loadXML(\$xmlfile); \$xml=simplexml_import_dom(\$dom); \$xxe=\$xml->xxe; \$str="\$xxe \n"; echo \$str; ?>

	加载XML文件失败，如不成功则返回SimpleXMLElement对象，如果失败则返回FALSE。	<pre> <?xml-stylesheet type="text/css" src="http://x.z.aliyun.com/t/6887" /> </pre> <p>来自 https://xz.aliyun.com/t/6887</p>
simplexml_load_string	加载解析XML字符串 有可能存在XXE 漏洞	<pre> \$xml=simplexml_load_string(\$_REQUEST['xml']); print_r(\$xml); </pre>
simplexml_load_file	读取文件作为XML文档解析 有可能存在XXE 漏洞	
unserialize	反序列化	

PHP原生过滤方法

过滤函数

`escapeshellarg` 传入参数添加单引号并转义原有单引号 用于防止命令注入

例 传入 'id #' 处理后 '\ id #' 处理后的字符串可安全的添加到命令执行参数

`escapeshellcmd` 转义字符串中的特殊符号 用于防止命令注入

反斜线 (\) 会在以下字符之前插入: '|*?~<>^()[]{}\$\\, \x0A 和 \xFF。' 和 " 仅在不配对儿的时候被转义

来自 <<https://www.php.net/manual/zh/function.escapeshellcmd.php>>

`addslashes` 在单引号 (')、双引号 (")、反斜线 (\) 与 NUL 前加上反斜线 可用于防止SQL注入

`mysqli::real_escape_string` `mysqli::escape_string` `mysqli_real_escape_string`
`mysql_real_escape_string` `SQLite3::escapeString`

以上函数会在 \x00(NULL), \n, \r, , ' , " 和 \x1a (CTRL-Z) 前加上反斜线 并考虑了当前数据库连接字符集进行处理

注意: 经过以上函数处理后的字符串不可直接用于sql查询拼接 需要使用引号包裹后拼接到sql语句中 否则仍可导致sql注入

`PDO::quote` 转义特殊字符 并添加引号

`PDO::prepare` 预处理SQL语句 有效防止SQL注入 (推荐)

`htmlspecialchars` 和 `htmlspecialchars` 将特殊字符转义成html实体 可用于防止XSS

`intval($input) floatval() floatval() floor() (int)$input num+0` 将输入强制转换为整数/浮点 常见于防止 SQL注入

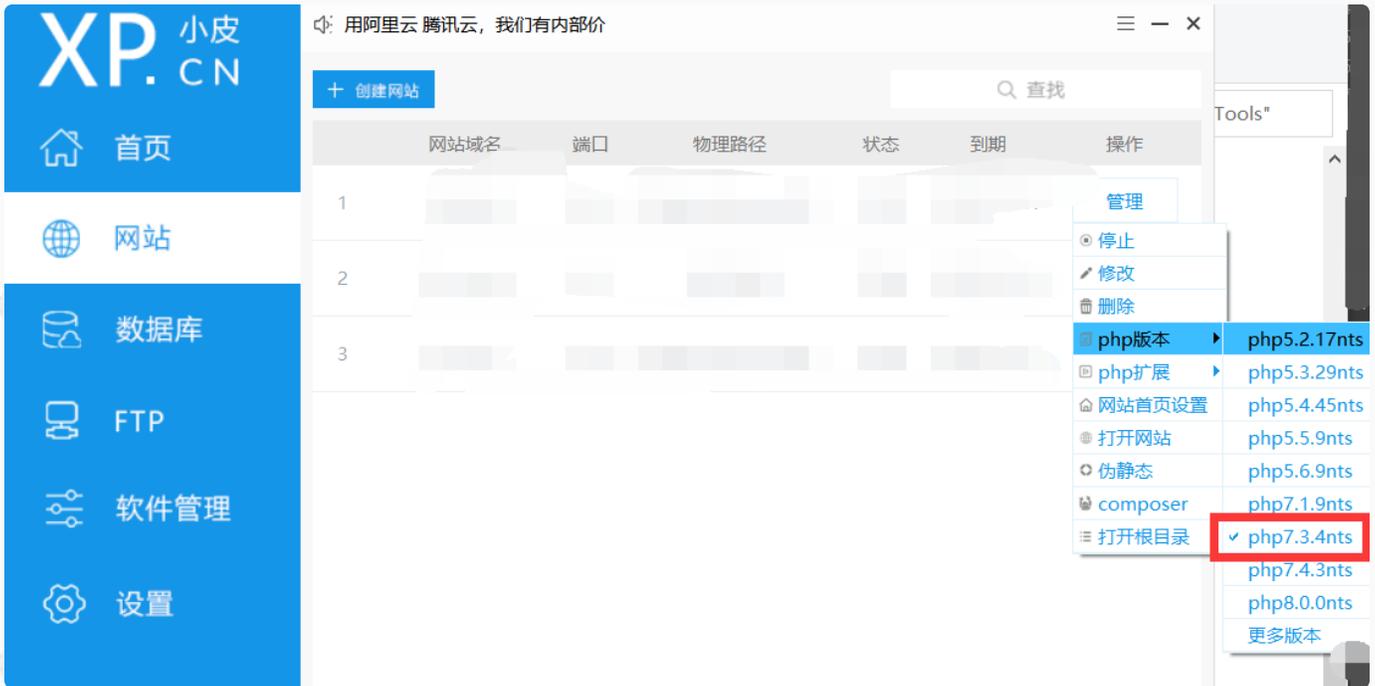
防护配置项

todo

白帽酱 -PHP代码审计入门指南

PHP动态调试-Xdebug安装配置

phpStudy+Xdebug+VSCode(Windows)



查看目标调试站点的PHP版本 确定Xdebug插件版本 也可以查看PHPINFO判断版本 (需要安装对应PHP版本的Xdebug插件 更换PHP版本需要重新安装)

<https://xdebug.org/download> Xdebug插件官方下载地址

Latest Release

Xdebug 3.1.1

Release date: 2021-10-15

- Linux, macOS:

[source](#)

- Windows binaries:

PHP 7.2 VC15 (64 bit)	PHP 7.2 VC15 (32 bit)	PHP 7.2 VC15 TS (64 bit)	PHP 7.2 VC15 TS (32 bit)
PHP 7.3 VC15 (64 bit)	PHP 7.3 VC15 (32 bit)	PHP 7.3 VC15 TS (64 bit)	PHP 7.3 VC15 TS (32 bit)
PHP 7.4 VC15 (64 bit)	PHP 7.4 VC15 (32 bit)	PHP 7.4 VC15 TS (64 bit)	PHP 7.4 VC15 TS (32 bit)
PHP 8.0 VS16 (64 bit)	PHP 8.0 VS16 (32 bit)	PHP 8.0 VS16 TS (64 bit)	PHP 8.0 VS16 TS (32 bit)
PHP 8.1 VS16 (64 bit)	PHP 8.1 VS16 (32 bit)	PHP 8.1 VS16 TS (64 bit)	PHP 8.1 VS16 TS (32 bit)

Command Line Debug Client

[Linux \(x86_64\)](#)

[macOS \(x86_64\)](#)

[Windows \(x86_64\)](#)

选择对应版本下载 (不标TS的版本为NTS)

将下载好的插件放在 PHPSTUDY安装目录\Extensions\php\PHP版本\ext\

例 X:\phpstudy_pro\Extensions\php\php7.3.4nts\ext\

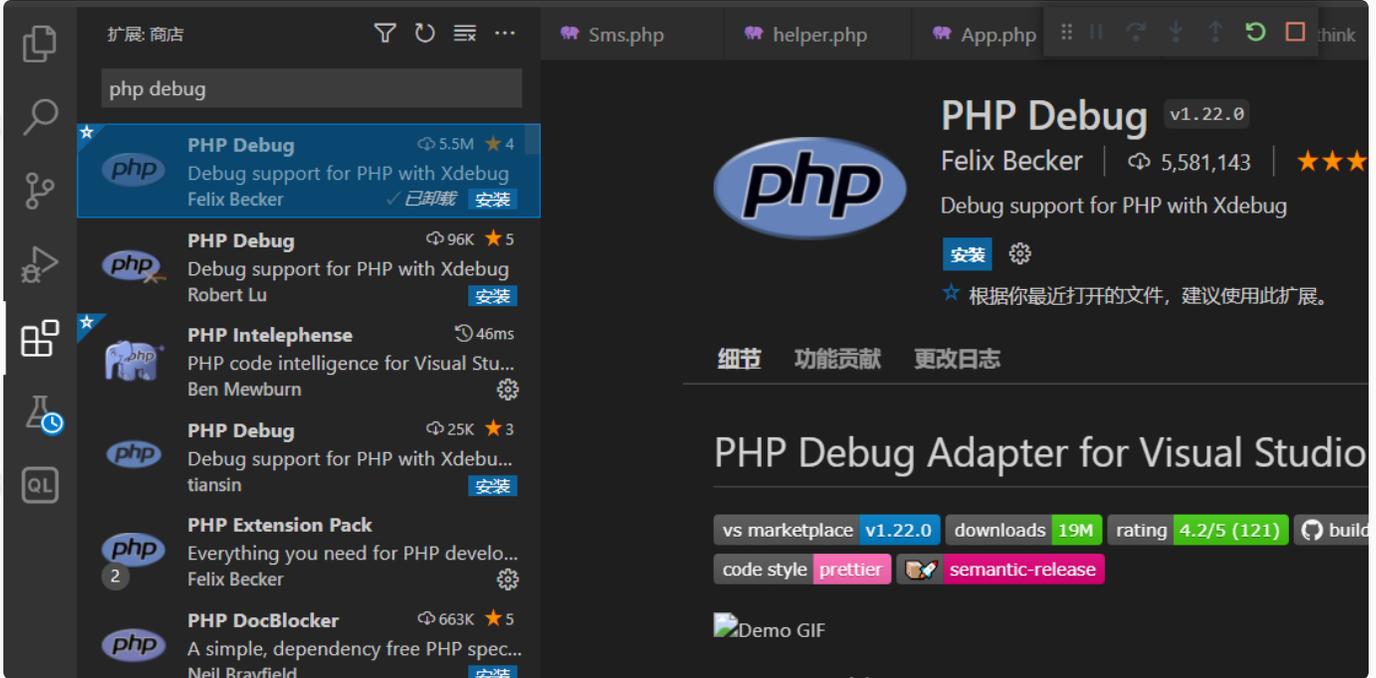
插件重命名为php_xdebug.dll (为了美观好记 雾)

在php.ini 末尾添加配置 (php.ini位于 PHPSTUDY安装目录\Extensions\php\PHP版本\php.ini)

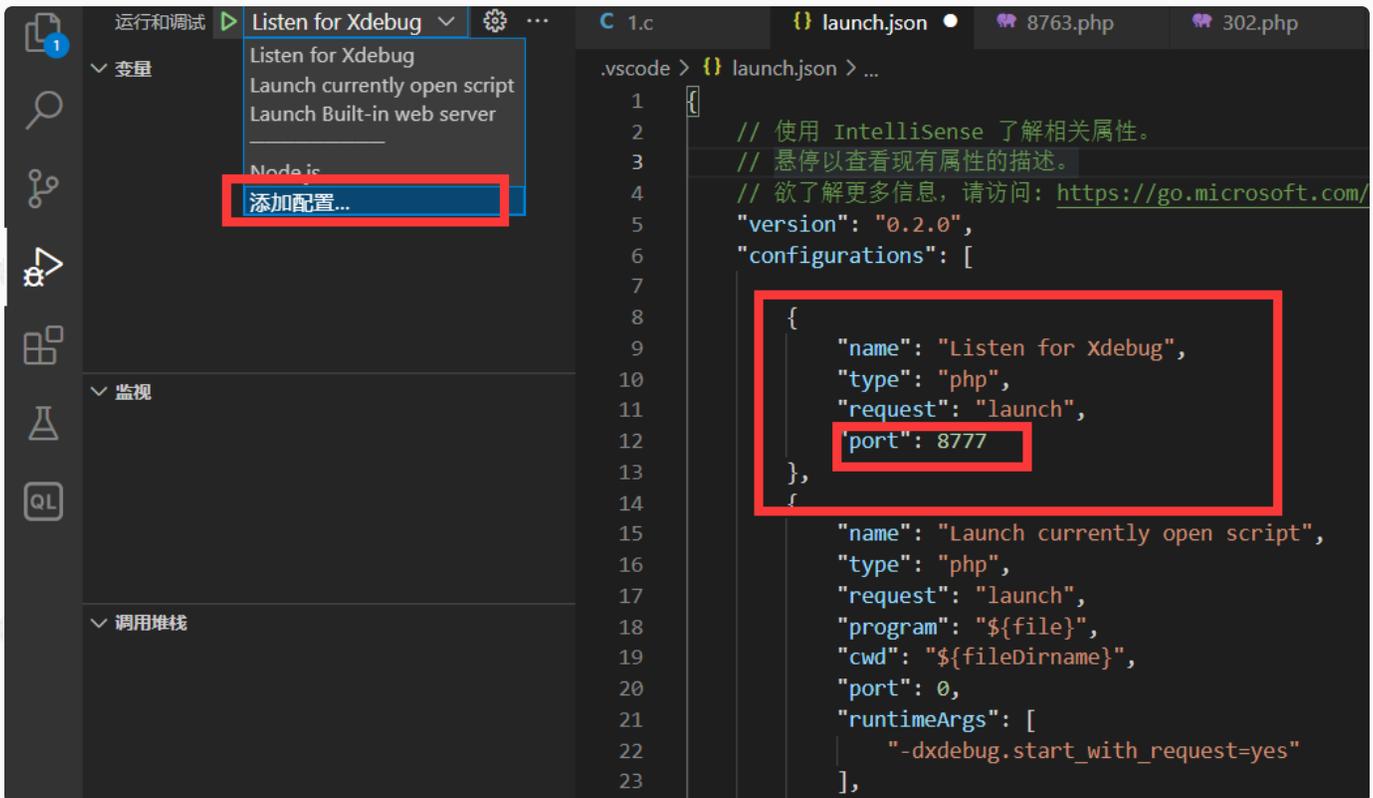
```
1  [Xdebug]
2  zend_extension=php_xdebug.dll
3  xdebug.client_port=8777
4  xdebug.client_host=127.0.0.1
5  xdebug.mode=debug
6  xdebug.remote_host=127.0.0.1
7  xdebug.remote_handler=dbgp
8  xdebug.start_with_request = yes
```

client_port要与vscode配置一致

重启WEB(PHP进程)服务

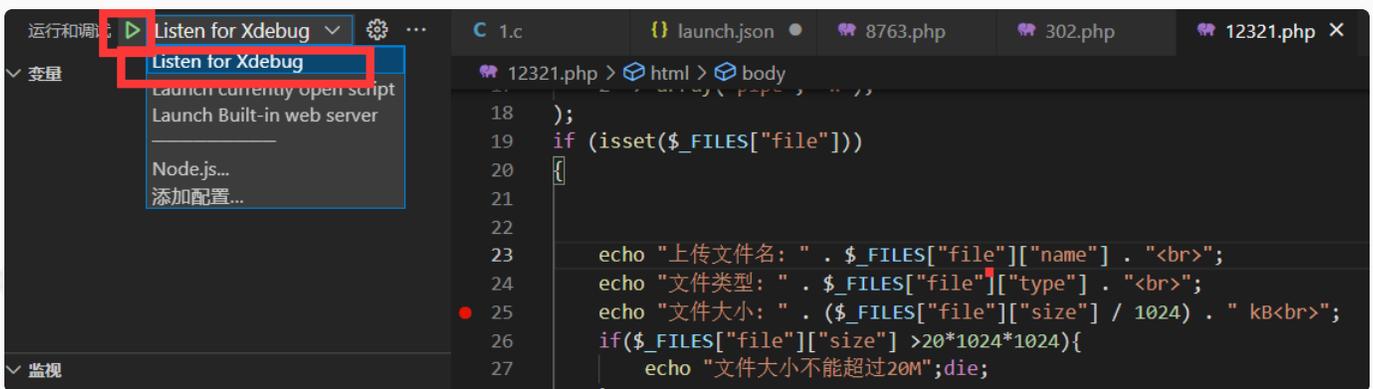


安装完毕 添加调试器配置



修改为之前的端口

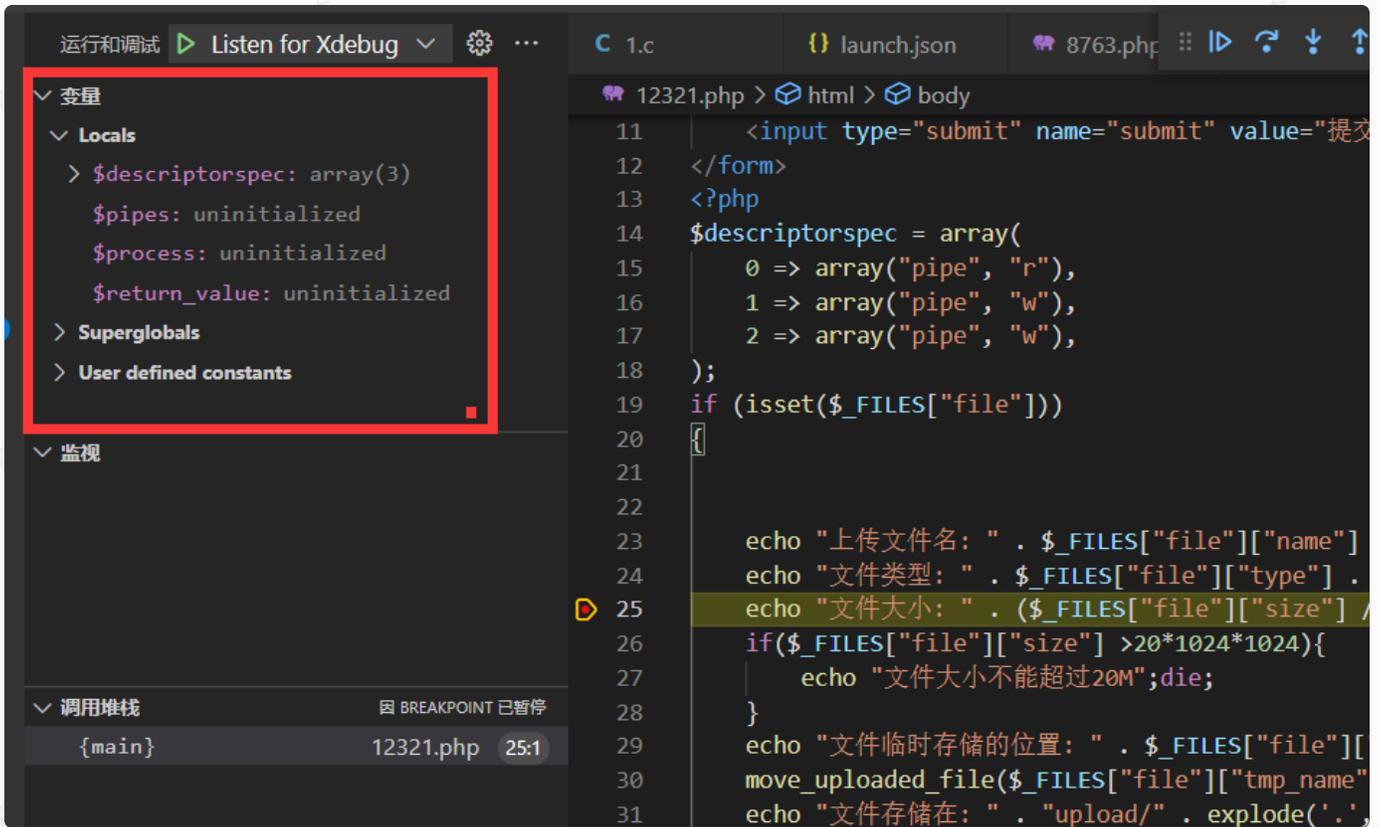
添加断点(快捷键 F9)后开始调试



访问目标调试页面

左侧出现变量即为成功调试

(注意 PHPSTUDY Apache环境默认有超时限制 超时返回 HTTP 500 – Internal Server Error 服务器内部错误)



Apache 超时配置 (TODO)

X:\phpstudy_pro\Extensions\Apache*\conf\original\extra\httpd-default.conf

白帽酱 -PHP代码审计入门指南

白帽酱 -PHP代码审计入门指南

白帽酱 -PHP代码审计入门指南

白帽酱 -PHP代码审计入门指南

白帽酱 -PHP代码审计入门指南

白帽酱 -PHP代码审计入门指南

命令注入

将用户输入拼接到命令行中执行 导致的任意命令执行问题

例子

```
1 <?php
2 $command = 'ping -c 1 ' . $_GET['ip'];
3 system($command); //system函数特性 执行结果会自动打印
4 ?>
```

这是一段简单的php代码 专门执行ping 命令并输出内容

正常输入: /xxx.php?ip=114.114.114.114

执行命令 ping -c 1 114.114.114.114

由于ip参数没有任何过滤限制

所以攻击者可以这样输入: /xxx.php?ip=114.114.114.114;whoami

执行命令 ping -c 1 114.114.114.114;whoami

这样就可以执行攻击者定义的命令 whoami

实际审计时输入常常不会非常简单 都有复杂的处理 慢慢追踪参数来源

遇到不会的函数可以在PHP手册查找对应的功能

命令执行基本语法不会的可以学习学习linux windows命令行语法基础

在审计时遇到输入可控时 要检查是否存在 `escapeshellarg` `escapeshellcmd` 函数转义 或者是其他的处理方法(如 强制类型转换 替换字符 等)

常见bash shell 语法

符号	描述	示例
<和>	输入输出重定向	<code>echo abc >1.txt</code>
分号	按照从左到右顺序执行命令	<code>id;whoami;ls</code>
管道符	将左侧命令的输出作为右侧命令的输入	<code>ps -aux grep root</code>
&&	按照从左到右顺序执行命令 只有执行成功才执行后面的语句	
	按照从左到右顺序执行命令 只有执行失败才执行后面的语句	

一些常见的可以执行系统命令的函数/语法

函数/语法	描述	例子
<code>system</code>	执行命令并输出结果	<code>system('id');</code>
<code>exec</code>	执行命令 只可获取最后一行结果	<code>exec('id',\$a); print_r(\$a);</code>
<code>passthru</code>	同 <code>system</code>	<code>passthru('id');</code>
<code>shell_exec</code> (反引号)	执行命令并返回结果	<code>\$a=shell_exec('id');print_r(\$a);</code> <code>\$a=`id`;print_r(\$a);</code>
<code>popen</code>	执行命令并建立管道 返回一个指针 使用 <code>fread</code> 等函数操作指针 进行读写	<code>\$a=popen("id", "r"); echo fread(\$a, 2096);</code>
<code>proc_open</code>	同 <code>popen</code> (进程控制功能更强大)	见PHP手册
<code>pcntl_exec</code>	执行命令 只返回是否发生错误	<code>pcntl_exec('id');</code>

代码注入

将用户输入拼接到PHP代码中执行 导致的任意代码执行问题

例子

有些特殊业务使用了eval等代码执行函数

```
PHP | 复制代码  
1 <?php eval( 'echo ('. $_GET['a']. ');'); //计算器?>
```

正常输入: ?a=9*9

服务器执行 echo (9*9);

输出:81

攻击者输入?a=system('whoami')

服务器执行echo (system('whoami'));

成功调用system函数执行命令

实际业务中要尽量避免使用eval这种动态执行代码方法 必要使用时做好过滤

函数/语法结构	描述	例子
eval	将传入的参数内容作为PHP代码执行 eval 不是函数 是一种语法结构 不能当做函数动态调用	<code>eval('phpinfo();');</code>
assert	将传入的参数内容作为PHP代码执行 版本在PHP7以下是函数 PHP7及以上为语法结构	<code>assert('phpinfo();');</code>
preg_replace	当preg_replace使用/e修饰符且原字符串可控时 有可能执行php代码	<code>echo preg_replace("/e","\${PHPINFO()}", "123");</code>
call_user_func	把第一个参数作为回调函数调用 需要两个参数都完全可控才可利用 只能传入一个参数调用	<code>call_user_func('assert', 'phpinfo();');</code>
call_user_func_array	同call_user_func 可传入一个数组带入多个参数调用函数	<code>call_user_func_array('file_put_contents', ['1.txt', '6666']);</code>
create_function	根据传递的参数创建匿名函数，并为其返回唯一名称 利用需要第二个参数可控 且创建的函数被执行	<code>\$f = create_function("", 'system(\$_GET[123]);'); \$f();</code>

文件包含

将远程/本地文件 包含入当前页面的PHP代码并执行 详细加载执行原理见[PHP7内核剖析](#)

例子

开发人员希望自己写的页面实现更加灵活的加载

```
PHP | 复制代码
1  <?php
2  $file = $_GET['page'];
3  include("pages/$file");
4  ?>
```

正常输入 ?page=login.php

```
PHP | 复制代码
1  <?php
2  $file = $_GET['page'];
3  /*
4  pages/login.php 文件代码被包含执行
5  */
6  ?>
```

服务器包含并执行pages目录下的login.php

攻击者输入 ?page=../image/123.jpg

服务器包含并执行pages的上层目录image目录下的123.jpg

该漏洞通常需要参数后半部分可控或者参数完全可控才存在

注意:当代码运行环境 php版本小于5.3.4且 php的magic_quotes_gpc为OFF状态时 参数在中间拼接也可利用 (CVE-2006-7243)(这是个PHP本身的问题 不是代码的问题 解决方法: 升级PHP)

参数在中间拼接时 如果用户仍可向拼接出的文件进行写入则可以利用

如 `include("pages/$file.tpl");`

假设用户不能上传php文件 但可上传tpl文件

可以上传一个tpl文件 构造路径包含tpl文件 执行php代码

文件包含利用方法

包含上传文件 (上传头像图片等)

包含 `data://php://filter` 或 `php://input` 伪协议 (php.ini allow_url_include 设置为 on)

包含日志 Apache nginx 等web服务器访问日志 SSH FTP 等登陆错误日志 PHP框架日志

包含 `/proc/self/environ` (必须是有proc伪文件系统的操作系统 比如LINUX) 当前进程的环境变量(PHP会将HTTP头 请求URI等信息写入当前进程环境变量)

包含 session文件 (通常在临时目录下 (linux /tmp/) sess_会话ID文件)

PHP间接或直接创建的其他文件 比如数据库文件 缓存文件 应用日志等

函数/语法结构	描述	例子
include	包含并运行指定文件 执行出错会抛出错误	include 'vars.php'; (括号可有可无)
require	同include 执行出错会抛出警告	require('somefile.php'); (括号可有可无)
require_once	同require 但会检查之前是否已经包含该文件 确保不重复包含	
include_once	同include 但会检查之前是否已经包含该文件 确保不重复包含	

SQL注入

将用户输入拼接到数据库将要执行的SQL语句中 导致攻击者可以修改原有执行的SQL语句

例子

```
PHP | 复制代码
1
2 <?php
3 include('conn.php');//数据库连接省略
4 $sql = "SELECT id, name FROM users WHERE id=$_GET['id']";
5 $result = $mysqli->query($sql);
6 if ($result->num_rows > 0) {
7     while($row = $result->fetch_assoc()) {
8         echo "id: " . $row["id"]. " - Name: " . $row["name"];
9     }
10 } else {
11     echo "没有查询到结果";
12 }
13 ?>
```

正常请求 ?id=123

执行SQL

```
SQL | 复制代码
1 SELECT id, name FROM users WHERE id=123;
```

攻击者构造请求: ?id=123 UNION SELECT name,password FROM users;

执行SQL

```
1 SELECT id, name FROM users WHERE id=123 UNION SELECT name,password FROM users;
```

攻击者改变了原有的SQL语句逻辑

常见过滤/防护

`addslashes` 在单引号 (')、双引号 (")、反斜线 (\) 与 `NUL` 前加上反斜线 可用于防止SQL注入

`mysqli::real_escape_string` `mysqli::escape_string` `mysqli_real_escape_string`

`mysql_real_escape_string` `SQLite3::escapeString`

以上函数会在 `\x00`(NULL)、`\n`、`\r`、`,`、`'`、`"` 和 `\x1a` (CTRL-Z) 前加上反斜线 并考虑了当前数据库连接字符集进行处理

注意: 经过以上函数处理后的字符串不可直接用于sql查询拼接 需要使用引号包裹后拼接到sql语句中 否则仍可导致sql注入

例如 上文中的例子 攻击者输入并没有使用到引号反斜线 逗号可使用其他方法绕过 仍可构成SQL注入

防护方法

```
1
2 <?php
3 /*强制类型转换*/
4 $id=intval($_GET['id']); //因查询ID为整数 所以可以强制转换为整数
5 /*转义特殊字符 加上引号 (字符串类型)*/
6 $id=$pdo->quote($_GET['name']);
7 /*预处理语句*/
8 $stmt =$pdo->prepare("SELECT id, name FROM users WHERE id=?;");
9 $stmt->execute( [$_GET['id']] );//简单的预处理 完整使用方法见PHP手册
10 ?>
```

PDO::quote 转义特殊字符 并添加引号

PDO::prepare 预处理SQL语句 有效防止SQL注入 (推荐)

intval(\$input) floatval() floatval() floor() (int)\$input num+0

将输入强制转换为整数/浮点 用于整数/浮点类型的输入参数处理 可防止SQL注入

一些执行SQL语句的函数

函数/方法	备注
mysql_query	
odbc_exec	
mysqli_query	
mysql_db_query	
mysql_unbuffered_query	
mysqli::query 用法 \$mysqli = new mysqli("localhost", "my_user", "my_password", "world"); \$mysqli->query();	
pg_query	
pg_query_params	
pg_send_query	
pg_send_query_params	
sqlsrv_query	
pdo::query \$pdo=new PDO("mysql:host=localhost;dbname=phpdem o","root","1234"); \$pdo->query(\$sql);	PDO
SQLite3::query SQLite3::exec \$db = new SQLite3('mysqitedb.db'); \$db-> query('SELECT bar FROM foo'); \$db-> exec('CREATE TABLE bar (bar STRING)');	

文件操作

文件操作相关关键参数用户可控 导致文件/目录 删除/移动/写入(上传)/读取等

文件/目录删除

```
PHP | 复制代码  
1 unlink(文件路径)//删除文件  
2 rmdir(文件夹路径)//删除目录
```

攻击者常见用法

删除lock文件(解除重复程序安装保护等安全限制)

删除网站关键文件(导致网站拒绝服务 数据丢失)

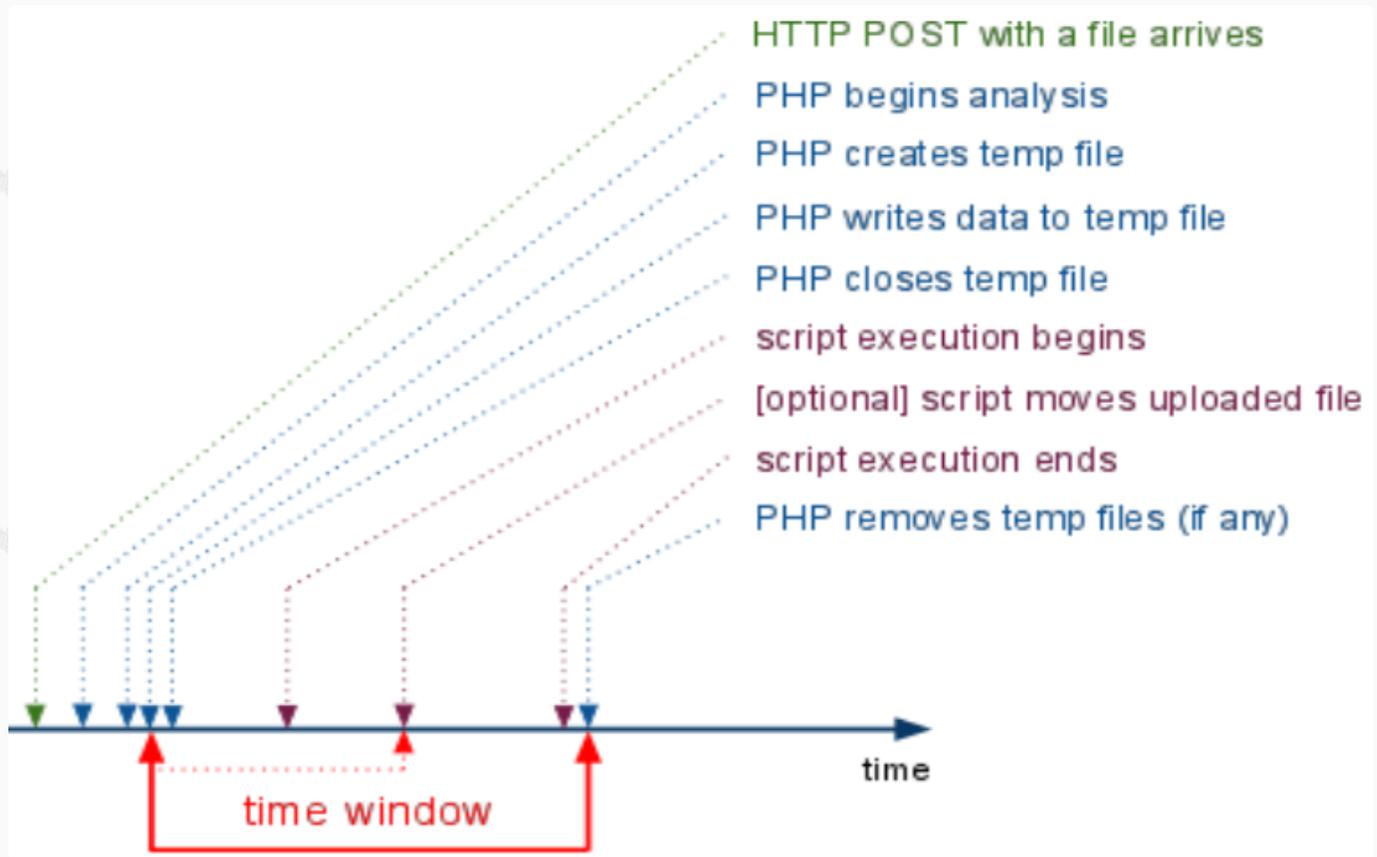
文件写入/上传

```
PHP | 复制代码  
1 文件写入  
2 file_put_contents(路径,写入字符串);//直接将字符串写入文件(不存在会自动创建)  
3  
4 $fp = fopen(文件路径, "w");//以写入模式打开一个文件 返回文件指针(不存在会自动创建)  
5 fwrite($fp,写入字符串);//写入数据  
6 fclose($fp);//关闭文件  
7 文件上传  
8 move_uploaded_file(临时上传文件路径,目标文件路径);//移动临时上传文件
```

php的原生文件上传 收到POST表单->随机文件名写入临时目录->(执行PHP文件处理逻辑->移动临时文件到保存位置)->删除临时文件(如果临时文件没有被移动)

临时文件路径必须是php上传表单自动处理产生的 例如 `$_FILES["pictures"]["tmp_name"]`

"pictures"为表单中的name "tmp_name"为固定变量名(临时文件名)



Gynvael Coldwind 《PHP_LFI_rfc1867_temporary_files》 2011.3

注:只要PHP收到POST上传文件表单 哪怕php页面一行代码没有 都会将上传文件保存到临时目录 在请求结束后如果临时文件没有被移走就会被自动删除 从写入文件到删除文件有个短暂的窗口时间 **可用于文件包含**

文件解压

```
1 $zip = new \ZipArchive;
2 $zip->open('test_new.zip', \ZipArchive::CREATE) //打开一个zip文件
3 $zip->addFile('test.txt'); //添加压缩文件
4 $zip->addEmptyDir('newdir');//添加空目录
5 $zip->addFromString('new.txt', '文本');//从字符串添加文件到压缩包
6 $zip->extractTo('upload');//将压缩包文件解压到upload目录下
7 $zip->close();//关闭zip
```

注:ZipArchive扩展在windows平台 php版本>5.6时默认安装. linux及windows其他版本需要手动编译安装.

审计时重点查找 extractTo方法

判断解压目录是否在web目录下 是否检查压缩包内文件类型 如果不在web目录下也可以使用.. 进行目录穿越控制上传目录 到web目录下 或者在权限足够的情况下写入文件到系统关键目录 (自启动 定时任务 ssh公钥 覆盖shadow 等)

文件写入/上传

函数	描述	例子
file_put_contents	将一个字符串写入文件	file_put_contents("1.txt","6666");
move_uploaded_file	将上传的临时文件移动到新的位置	move_uploaded_file(\$_FILES["pictures"] ["tmp_name"],"1.php")
rename	重命名文件/目录	rename(\$oldname, \$newname);
rmdir	删除目录	
mkdir	创建目录	
unlink	删除文件	
copy	复制文件	copy(\$file, \$newfile);
fopen/fputs/fwrite	打开文件或者 URL	https://www.php.net/manual/zh/function.fwrite.php
link	创建文件硬链接	link(\$target, \$link);
symlink	创建符号链接(软链接)	symlink(\$target, \$link);
tmpfile	创建一个临时文件 (在临时目录存放 随机文件名 返回句柄)	\$temp = tmpfile(); fwrite(\$temp, "123456"); fclose(\$temp);
request()->file()->move() request()->file()->file()	Thinkphp 文件上传	\$file = request()- >file(\$name); \$file->move(\$filepath);
extractTo	解压zip 到目录	

文件读取

函数	描述	例子
file_get_contents	读入文件返回字符串	<pre>echo file_get_contents("flag.txt"); echo file_get_contents("https://www.bilibili.com/");</pre>
readfile	读取一个文件，并写入到输出缓冲	同file_get_contents
fopen/fread/fgets/fgetss/fgetc/fgetcsv/fpassthru/fscanf	打开文件或者 URL 读取文件流	<pre>\$file = fopen("test.txt","r"); echo fread(\$file,"1234"); fclose(\$file);</pre>
file	把整个文件读入一个数组中	<pre>echo implode(file('https://www.bilibili.com/'));</pre>
highlight_file/show_source	语法高亮一个文件	highlight_file("1.php");
parse_ini_file	读取并解析一个ini配置文件	print_r(parse_ini_file('1.ini'));
simplexml_load_file	读取文件作为XML文档解析	

全部文件操作函数可参考PHP官方手册 (几乎大部分文件操作函数都可用于ssrf phar反序列化)
<https://www.php.net/manual/zh/ref.filesystem.php>

XSS

白帽酱 - PHP代码审计入门指南

白帽酱 - PHP代码审计入门指南

跨站脚本(攻击) 让用户浏览器执行到攻击者指定的JS脚本代码

XSS

白帽酱 - PHP代码审计入门指南

白帽酱 - PHP代码审计入门指南

反射型XSS

反射型XSS是服务器后端处理时把处理不当的用户输入输出到网页 导致用户浏览器执行恶意代码

```
PHP | 复制代码
1 <?php
2 echo 'Hello ' . $_GET['name'] . '!';
```

正常输入: ?name=cz

服务器返回 Hello cz!

浏览器渲染纯文本 Hello cz!

```
HTML | 复制代码
1 攻击者输入: ?name=<script type="text/javascript">alert('XSS!');</script>
2 服务器返回 Hello <script type="text/javascript">alert('XSS!');</script>!
3 浏览器渲染 Hello ! script被作为html标签解析 执行其中的代码 弹出警告框XSS!
```

此种漏洞比较明显 很容易分析问题的存在

审计时注意 PHP常使用 htmlspecialchars 和 htmlentities函数 转义用户的输入作为防护

储存型XSS

将服务器储存的处理不当的用户输入输出到网页 导致用户浏览器执行恶意代码

攻击者输入->服务器储存 攻击者得到一个返回储存值的页面

受害者请求页面->服务器调用储存并输出->XSS

常见于评论 留言 文章 等

该漏洞需要同时联合查看储存与输出 有时业务复杂 多写入点 多输出点 不易发现

比如 发布评论 攻击者发布带有恶意代码的评论 受害者访问评论展示页面 触发XSS 后台管理员审核评论时 触发XSS

再比如 用户系统设置昵称时 攻击者将昵称写入数据库 在评论显示时读取数据库值输出用户昵称

DOM型 XSS

纯dom

储存(反射)dom

DOM型 XSS 只与浏览器前端DOM渲染有关 不做赘述

前端外部文件引用

攻击者修改前端引用的文件链接 引用外部网站文件

常见于用户头像 文章/评论图片等

受害者访问到攻击者个人页面 文章 评论 聊天内容时会访问远程图片文件

这可能会使攻击者获取到访问者的ip 浏览器 系统等信息

也可以绕过内容审查 在审查通过后动态替换文件内容(hsbc广告等信息)

解决方法:正则匹配限制url域名

防护

后端过滤

服务端返回HTTP头 添加内容安全策略 `content-security-policy` 头

正确设置安全策略可以有效减少未知XSS/html外部文件引用漏洞产生的危害

COOKIES添加Httponly属性 防止使用js读取用户cookies (js发起表单仍可携带cookies)

SSRF

服务端请求伪造

SSRF

让服务器发起攻击者指定的请求(HTTP/HTTPS/TCP/UDP 等)

攻击者通常用来访问/攻击内网服务 获取内网信息 绕过ip限制

SSRF的函数几乎和文件读取操作一样 php中绝大多数文件读取/写入操作函数都支持多种协议(包括 HTTP/S FTP 伪协议 等)

漏洞常见处: 远程图片下载

函数	描述	例子
file_get_contents	读入文件返回字符串	echo file_get_contents("https://www.bilibili.com/");
fsockopen	打开一个套接字连接(远程 tcp/udp raw)	https://www.php.net/manual/zh/function.fsockopen.php
readfile	读取一个文件, 并写入到输出缓冲	同file_get_contents
fopen/fread/fgets/fgetss/fgetc/fgetcsv/fpassthru/fscanf	打开文件或者 URL 读取文件流	\$file = fopen("test.txt","r"); echo fread(\$file,"1234"); fclose(\$file);
file	把整个文件读入一个数组中	echo implode(", file('https://www.bilibili.com/');
highlight_file/show_source	语法高亮一个文件	highlight_file("1.php");
parse_ini_file	读取并解析一个ini配置文件	print_r(parse_ini_file('1.ini'));
simplexml_load_file	读取文件作为XML文档解析	

CSRF

跨站请求伪造

CSRF

表单请求

攻击者使被害者的浏览器在用户不知情的情况下发起目标网站表单请求 这些表单常常带有目标网站的用户cookies 可以以用户在目标网站的身份进行操作 (攻击者不能获取cookies)

检查鉴权后的操作是否添加token/Referrer校验 拒绝空Referrer

JSONP请求

除了表单常见的还有jsonp请求 服务器返回一段带有函数调用的json 浏览器把jsonp页面当做js加载执行 调用回调函数将数据传入函数 用于浏览器从服务器动态获取信息 由于js等静态资源调用浏览器默认放行 造成了风险

(get表单也可以使用这种方式发起请求 但是无法获取服务器返回的内容)

可获取用户登陆后才能获取的信息 比如登陆用户个人资料 账户余额 等

检查Referrer 拒绝空Referrer

AJAX请求

一种使用js动态加载数据的技术

浏览器会先发送一个HEAD请求 获取HTTP头 检查Access-Control-Allow-Origin等Access-Control安全策略

这会决定是否发起带有目标域浏览器cookies的请求 如果没有头或不符合策略则拒绝请求

审计时检查 HTTP是否错误地添加"Access-Control-Allow-Origin:*" 头

防护方法

添加随机token 在表单/jsonp请求时附加token(非常有效)

服务端检测 Referrer (**一定要拒绝空Referrer** html表单可以发起空referrer)(表单/静态资源引用/jsonp请求)

如果使用正则匹配一定要检查正则是否可以被绕过

服务器返回Access-Control-Allow-Origin 头 (表单/静态资源引用/jsonp请求无效 仅AJAX请求)(**一定不要设置成 ***)

XXE

XML外部实体(注入) 攻击者利用xml的性质可以获取本地/远程文件内容 (不同于其他语言 PHP 中xml实体可以使用PHP伪协议)

XXE

XML外部实体是XML的一个特性 XML可以使用外部实体引用来包含和解析其他文档

当然XML还有其他实体 详细内容可以参考[这个DTD教程](#)

这里就不详细将利用技巧了

审计时如果发现使用了文末列表的函数 就要检查是否禁用了外部实体

```
libxml_disable_entity_loader(true); //禁用外部实体使用到的函数 参数为true时禁用
```

注意: php环境中libxml 版本 $\geq 2.9.0$ 时外部实体默认禁用 (**PHP版本 ≥ 8.0 时 就开始使用 $\geq 2.9.0$ 版本的 libxml** 且 **libxml_disable_entity_loader**函数被完全废弃 使用该函数会抛出错误)

漏洞常见处: 支付等回调api

函数	描述	
DOMDocument::loadXML	加载解析XML	<pre><?php \$xml=file_get_contents('php:/input'); \$dom=new DOMDocument(); \$dom->loadXML(\$xml); \$xml=simplexml_import_dom(\$dom); \$xml->xxe; echo \$xxe; ?></pre>
simplexml_load_string	加载解析XML字符串	<pre>\$xml=simplexml_load_string(\$_REQUEST['xml']); print_r(\$xml);</pre>
simplexml_load_file	读取文件作为XML文档解析	<pre>simplexml_load_file("1.xml")</pre>

反序列化

序列化

函数: `serialize`

将PHP中的值转化为一个字符串 有利于存储或传递 PHP 的值 同时不丢失其类型和结构

如果被序列化的值是一个对象 则会调用对象的`__sleep` 魔法函数

反序列化

函数:`unserialize`

反序列化在序列化操作后产生的字符串 还原序列化前的值

如果被反序列化的结果包含对象 则会调用对应对象的`__wakeup` 魔法函数

如果反序列化的字符串被用户可控 攻击者则有可能利用PHP中现有的对象调用对应魔法函数进行攻击(主要看对象定义的魔法函数所拥有的功能)

如果存在可控反序列化点则存在风险. 但在PHP语言中,如果反序列化没有可以利用的反序列化链,则无法造成危害 [可以看看这篇文章来详细了解反序列化原理及利用方法](#)

除了直接调用`unserialize`函数外 还有覆盖session文件 其他调用反序列化的扩展函数 触发反序列化 因并不常见未做整理

LDAP注入

Ldap的注入是Ldap搜索过滤器的注入

审计时可以搜索ldap_search 函数判断第三个参数是否可控

检查是否正确转义了特殊符号

```
Plain Text | 复制代码  
1 "' \ / \ \ 空格 # <> , ; + * () \x00(null)
```

将以上字符转换成ascii码值 在其前面加上反斜线

只转义以下这6个字符就足以防止常见的Ldap注入

```
Plain Text | 复制代码  
1 * \ ( ) \x00
```

```
PHP | 复制代码  
1 function ldapspecialchars($string) {  
2     $sanitized=array('\ ' => '\5c',  
3         '*' => '\2a',  
4         '(' => '\28',  
5         ')' => '\29',  
6         "\x00" => '\00');  
7     return  
8     str_replace(array_keys($sanitized),array_values($sanitized),$string);  
}
```

Ldap过滤函数代码来自 [Pino_HD 【LDAP】LDAP注入漏洞与防御 \[2017.09.27\]](#)

如果想要详细了解利用及原理可以参考文章 [ldap注入入门学习](#)

```
PHP | 复制代码
1
2 $ds=ldap_connect($ldapSrv,$port);//建立ldap连接
3 if($ds) {
4     $r=ldap_bind($ds, "cn=".$username.", ".$dn, $passwd);//绑定ldap区域(相当
    于登陆ldap服务器) 使用域管用户登陆 检索用户列表
5     if($r) {
6         $sr=ldap_search($ds, $dn, "(|(cn=".$_GET["user"].")
        (mail=".$_GET["user"]."))");//在ldap中使用过滤器搜索
7         $info = ldap_get_entries($ds, $sr);
8         if($info["count"]==0){
9             die('用户不存在');
10        }
11        //if($info[0]["userpassword"][0]==$_GET["pass"]){
12        // die('登陆成功');
13        // }else{
14        // die('密码错误');
15        // }
16        ldap_close($ds);
17        $ds=ldap_connect($ldapSrv,$port);//建立ldap连接
18        $bd = ldap_bind($conn, $_GET["user"], $passwd); // 绑定ldap区域
        (相当于登陆ldap服务器) 以普通用户登陆 判断是否登陆成功
19        if ($bd) {
20            echo '登陆成功';
21        } else {
22            echo '密码错误';
23        }
24        } else {
25            echo "Unable to connect to LDAP server.";
26        }
27    }
```

Ldap过滤器结构表

```
1   Fileter = (filtercomp)
2   Filtercomp = and / or / not / item
3   And = & filterlist
4   Or = | filterlist
5   Not = ! filter
6   Filterlist = 1*filter
7   Item = simple / present / substring
8   Simple = "=" / "~=" / ">=" / "<="
9   Present = attr =*
10  Substring = attr "=" [initial]*[final]
11  Initial = assertion value
12  Final = assertion value
```

其他漏洞

这里的漏洞列表与业务功能强相关 需要根据当前功能分析是否存在问题

其他漏洞

要比产品更懂业务

业务相关漏洞是非常灵活的一类漏洞

在实际漏洞挖掘过程中慢慢学习 不要受所学内容的限制 大胆的尝试 可能会发现一些意想不到的漏洞

- 多挖挖互联网大厂SRC平台的业务强相关漏洞 在挖掘达到一定数量后 相信你会业务漏洞会有更深的理解

越权漏洞

越权是指用户可以进行超出业务设计上的权限限制的访问/操作

平行越权: 访问/操作与当前用户同等权限的其他用户的数据

例:普通用户A可以删除普通用户B投稿的文章 从业务逻辑设计上用户A只能删除自己创建的稿件 但是在后端未做相应的限制

垂直越权: 访问/操作与当前用户未拥有权限的数据

例:日志审计管理员 设计上可以审计服务器日志 但不能进行其他操作 系统管理员可以进行 系统配置 管理账号等操作

如果使用日志审计管理员账号可以进行系统管理员的系统配置操作 这就是一个垂直越权

越权漏洞非常灵活 具体细节要结合业务功能进行判断是否存在问题

未授权访问/无鉴权/鉴权绕过

访问/操作业务前未进行权限检查 或权限检查不严易被绕过

比如上文的越权漏洞

如果用户未进行登陆操作仍然可以访问后台页面获取数据/进行后台操作 这就是未授权访问(未授权访问 >>垂直越权)

有些后台操作为了方便添加后门 采用了弱校验(硬编码token 从head获取的UA和IP地址 弱JWT秘钥等) 这就可能造成鉴权绕过

频率限制

在一些关键业务点应做频率限制

例如 用户登陆 使用ip/用户名限制单位时间内登陆次数 必要时增加多次失败锁定限制 (可以用例如 memcache redis等缓存机制 记录次数 限制频率)配合验证码进行限制 防止账号密码爆破

用户回复 发表文章 订阅关注功能 发起订单 等 也应增加频率和次数限制 防止大量填充垃圾信息 频率过高时增加验证码校验

邮箱/短信验证码 加上单日次数及频率限制 必要时增加验证码校验

拒绝服务

通过特殊的用户输入 消耗的服务器资源 比如生成图片不限宽高(二维码 验证码 图片处理)

URL跳转

跳转到网站外第三方链接 可用于钓鱼 如果存在head注入则可以构成xss

验证码问题

用于区分机器的验证码

敏感业务操作 无验证码/前端验证码

例如后台登陆校验用户密码处无验证码 可爆破密码

生产环境业务 无验证码 可撞库/爆破

验证码无有效期限限制 (验证码重复使用)

验证码过于简单 易识别

比如直接使用简单的ocr就可以识别的验证码 文字背景无干扰数据

用于临时凭据的验证码/验证链接

数字验证码

无尝试次数/频率限制时易被爆破 (例如 邮件/短信验证码 **TOTP 没错 TOTP也可以爆破 曾经在真实环境用了某些技巧后爆破成功过几次**)

验证码在特定条件被返回给用户(获取验证码的同时返回验证码 系统执行出错时 错误信息返回验证码)

验证码无有效期限限制 (验证码重复使用)

验证链接

验证链接易被猜测/遍历

验证链接hash使用弱盐 爆破盐值

验证链接无有效期限限制 (验证链接重复使用)

验证链接在特定条件被返回给用户(获取验证码的同时返回验证码 系统执行出错时 错误信息返回验证码)

不安全的凭据储存/处理

储存/传递用户密码使用弱hash/明文

比如不加盐的单个(MD5/SHA1)弱hash处理 易使用彩虹表碰撞 (加随机盐 只能使预先验算的hash+明文组合彩虹表无法使用 但是仍易使用明文再次碰撞)

如果对密码储存安全性要求高的话 可以使用PHP内置的Bcrypt强hash算法进行hash处理 (BCrypt的安全性是通过牺牲性能来获取的 所以说**可能会有性能问题**)

传递明文密码

比如说web登录页 未对密码进行hash处理

攻击者通过某种手段(截获用户端/isp/服务器/网络设备(交换机/流控)等流量)可以获取到用户流量明文密码

密码重放

hash之后的密码就安全了吗?

攻击者仍可以将截获的hash进行碰撞猜测明文

或使用hash直接完成登陆过程

使用非对称加密可以完美解决问题 比如RSA

服务器生成一组固定或随机的RSA密钥 和一串随机生成的challenge

客户端获取RSA公钥和challenge 将密码和challenge拼接 RSA公钥加密后发送到服务器

服务器解密数据 对比临时储存的challenge和数据库中的密码是否符合

密码储存在意外位置

比如将密码等凭据打印到日志中

使用GET请求参数传递密码 导致密码记录在web日志中

密码规则限制

设置密码时应检测是否为常见弱口令

密码强度限制 比如强制大小写字母数字特殊符号组合

密码长度限制 密码长度不易过短至少8个字符 太短易被猜测 **密码长度不易过长 超长密码在hash计算时可能会导致拒绝服务**

密码不可以包含(部分)用户名 生日 姓名 身份证号码等易被猜测的个人信息
修改密码时应检测是否使用了历史密码 比如保存10次以内的历史密码hash 修改密码时比较

白帽酱 -PHP代码审计入门指南

Thinkphp

模型 视图 控制器

THINKPHP 是一个MVC PHP框架

所谓MVC 是Model View Controller(模型 视图 控制器)的缩写

简单理解

模型主要负责数据处理 (数据库交互 为控制器提供数据)

视图 负责渲染输出展示给客户端的HTML页面

控制器 接收用户请求 与客户端发生交互 处理基本的业务逻辑 决定要输出的视图

使用THINKPHP框架开发的项目 主要审计模型和控制器文件

目录结构

```
1 project 应用部署目录
2 |—application 应用目录 (可设置)
3 |   |—common 公共模块目录 (可更改)
4 |   |—index 模块目录(可更改)
5 |   |   |—config.php 模块配置文件
6 |   |   |—common.php 模块函数文件
7 |   |   |—controller 控制器目录
8 |   |   |—model 模型目录
9 |   |   |—view 视图目录
10 |   |   └─... 更多类库目录
11 |   |—command.php 命令行工具配置文件
12 |   |—common.php 应用公共 (函数) 文件
13 |   |—config.php 应用 (公共) 配置文件
14 |   |—database.php 数据库配置文件
15 |   |—tags.php 应用行为扩展定义文件
16 |   └─route.php 路由配置文件
17 |—extend 扩展类库目录 (可定义)
18 |—public WEB 部署目录 (对外访问目录)
19 |   |—static 静态资源存放目录(css, js, image)
20 |   |—index.php 应用入口文件
21 |   |—router.php 快速测试文件
22 |   └─.htaccess 用于 apache 的重写
23 |—runtime 应用的运行时目录 (可写, 可设置)
24 |—vendor 第三方类库目录 (Composer)
25 |—thinkphp 框架系统目录
26 |   |—lang 语言包目录
27 |   |—library 框架核心类库目录
28 |   |   |—think Think 类库包目录
29 |   |   └─traits 系统 Traits 目录
30 |   |—tpl 系统模板目录
31 |   |—.htaccess 用于 apache 的重写
32 |   |—travis.yml CI 定义文件
33 |   |—base.php 基础定义文件
34 |   |—composer.json composer 定义文件
35 |   |—console.php 控制台入口文件
36 |   |—convention.php 惯例配置文件
37 |   |—helper.php 助手函数文件 (可选)
38 |   |—LICENSE.txt 授权说明文件
39 |   |—phpunit.xml 单元测试配置文件
40 |   |—README.md README 文件
41 |   └─start.php 框架引导文件
42 |—build.php 自动生成定义文件 (参考)
43 |—composer.json composer 定义文件
44 |—LICENSE.txt 授权说明文件
45 |—README.md README 文件
```

./application/xxx/controller/xxxx.php 控制器文件默认位置

./application/xxx/model/xxxx.php 模型文件默认位置

application 目录名可以修改 (某些版本为app 可以在配置文件中自定义)

thinkphp问题大多都出在控制器 初学者建议重点审计控制器文件

对框架有一定了解后可以深入审计 不要太局限于控制器

未完待续

Laravel

白帽酱 -PHP代码审计入门指南

Codeigniter

白帽酱 - PHP代码审计入门指南

Yii

白帽酱 -PHP代码审计入门指南

Cakephp

白帽酱 -PHP代码审计入门指南

-
白帽酱 -PHP代码审计入门指南

白帽酱 -PHP代码审计入门指南

TODO

白帽酱 -PHP代码审计入门指南

无标题

路径长度截断

超出文件系统长度限制

?file=../../../../../../../../etc/passwd/../../../../.[...]/../../../../.

(php版本<5.2.8 linux需要长于4096 windows需要长于256)

PHP商业源码提取

从安装镜像提取

从磁盘镜像提取

重置linux密码

白帽酱 -PHP代码审计入门指南

浅谈PHP源代码保护方案&受保护PHP代码の解密还原

前言

php是一种解释型脚本语言.

与编译型语言不同,php源代码不是直接翻译成机器语言.而是翻译成中间代码(OPCODE),再由解释器(ZEND引擎)对中间代码进行解释运行.

在php源代码的保护在原理可以分为3大类.

- 源代码混淆(编码)
- OPCODE混淆(编码)
- 修改解释引擎(虚拟机)

在部署上可以分为2大类.

- 无扩展
- 有扩展

下面分析下各种加密方案的实现方法

PHP 加密方案分析

无扩展方案

源代码混淆

无扩展的加密在一些小开发者比较常见。

这种源代码保护方式侵入性小，无需对服务器做额外的配置，兼容性较强。

这种情况混淆后的源代码还原非常简单，可完全还原出源代码。有时连注释都会保留（x 我觉得这种混淆都不能称之为加密

基本流程 压缩代码->混淆变量函数类名->使用简单函数和方法进行编码加密 例:base64 异或


```

9  if (!defined( constant_name: "BDFBFACFBEDBFBF") {
10 define("BDFBFACFBEDBFBF", __FILE__);
11 global $DEL, $配, $潘, $津, $愁, $咪, $崑, $至, $妖, $撩, $污, $屋;
12 function DEL($DEL, $配 = "") {
13     1. 字符串解码函数
14     2. 内容
15     3. 换表
16     if (empty($配)) {
17         return base64_decode($DEL);
18     } else {
19         return DEL(strstr($DEL, $配), $屋);
20     }
21 }
22 $屋 = DEL("c3RycmV2");
23 $屋 = DEL("c3RydHI=");
24 $配 = DEL("nHJlZ19yZ资BsYWNl", "cmRfn");
25 $愁 = DEL("NmBgZTN0X2RLN29kZQ==", "YFtgzLBN");
26 $污 = DEL("B2Yy0GJm0e酒iM2RhNzVh槽eR10GNLj脊1MDA5M2Nk俱DAzB2U=", "LWmeB");
27 $火 = DEL("SXShbA=", "ZtyKzxS");
28 $酒 = DEL("w3p1bmXv澄NBiwNX", "ZXFyiRNw");
29 function 潘($潘)
30 {
31     global $DEL, $配, $潘, $津, $愁, $咪, $崑, $至, $妖, $撩, $污, $屋;
32     $潘 = DEL("RGLL", "ZcYR");
33     @($潘($潘)) . "(@$潘($潘))";
34     return "Z";
35 }
36 }
37 } else {

```

这里有一个php的特性 php中的base64遇到非base64表中字符会直接忽略 不会影响解码
 注: PHP7 遇到空字符可能会抛出error 可以使用php5.6执行 (这里有一个兼容性问题)
 遇到这种加密最简单的方法就是找文件中最后一步执行的函数 直接把内容打印出来
 这种编码方法最后一步肯定要使用eval执行还原后的php代码 所以打印最后一个函数基本上php代码就会全部出来 (x 前面操作一大顿毫无卵用)
 注: 有保护方案也使用了call_user_func或call_user_func_array间接调用eval

这里演示的是 hook zend_compile_string 函数

白帽酱 -PHP代码审计入门指南

```
1  /* $Id$ */
2
3  #include "php.h"
4  #include "ext/standard/info.h"
5
6
7
8
9  static zend_op_array* (*old_compile_string)(zval *source_string, char
    *filename TSRMLS_DC);
10
11
12  static zend_op_array* evalhook_compile_string(zval *source_string, char
    *filename TSRMLS_DC)
13  {
14
15      if(strstr(filename, "eval()'d code")) {
16
17          printf("\n-----eval-----\n%s\n-----eval-----
    \n", Z_STRVAL_P(source_string));
18      }
19      return old_compile_string(source_string, filename TSRMLS_CC);
20  }
21
22
23  PHP_MINIT_FUNCTION(evalhook)
24  {
25      return SUCCESS;
26  }
27
28  PHP_MSHUTDOWN_FUNCTION(evalhook)
29  {
30      return SUCCESS;
31  }
32
33  PHP_RINIT_FUNCTION(evalhook)
34  {
35      old_compile_string = zend_compile_string;
36      zend_compile_string = evalhook_compile_string;
37      return SUCCESS;
38  }
39
40  PHP_RSHUTDOWN_FUNCTION(evalhook)
41  {
42      zend_compile_string = old_compile_string;
```


成功还原源代码

PHP扩展方案

源代码混淆

使用php扩展的代码混淆和无扩展代码混淆比较相似，只不过是把代码还原过程从php代码转到了php扩展。

同样也是使用aes des 异或等加密方法直接加密php代码，HOOK翻译php的函数在翻译PHP文件前对文件进行解密操作。这种方案也可以完全还原出源代码。在无其他混淆和压缩时甚至还会保留注释。

典型开源项目：[php-beast](#) [tonyenc](#) [screw-plus](#)

手工解密

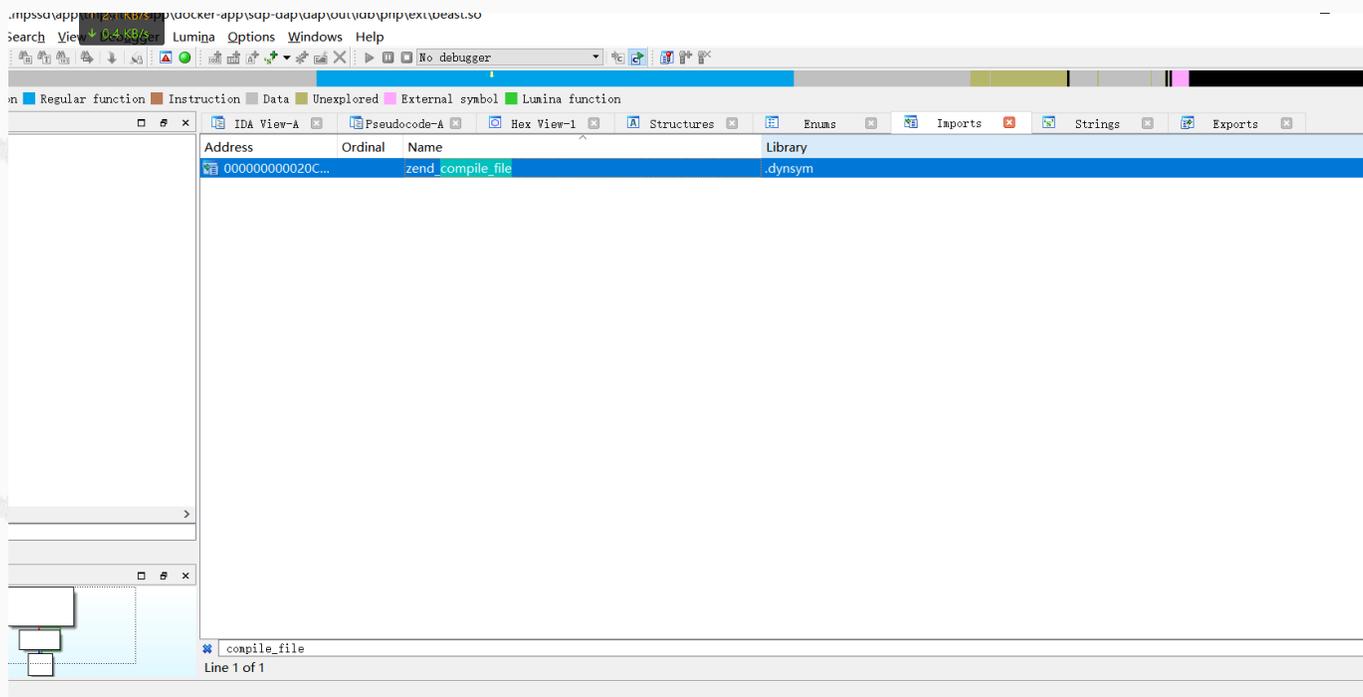
这里以beast为例。

首先在php的扩展目录下找到beast.so

beast的加密方案会把加密key编译进扩展中。我们只需要寻找key就可以完成解密

beast由于是开源项目。有现成的符号表和源码这使得反编译寻找key变得非常简单。

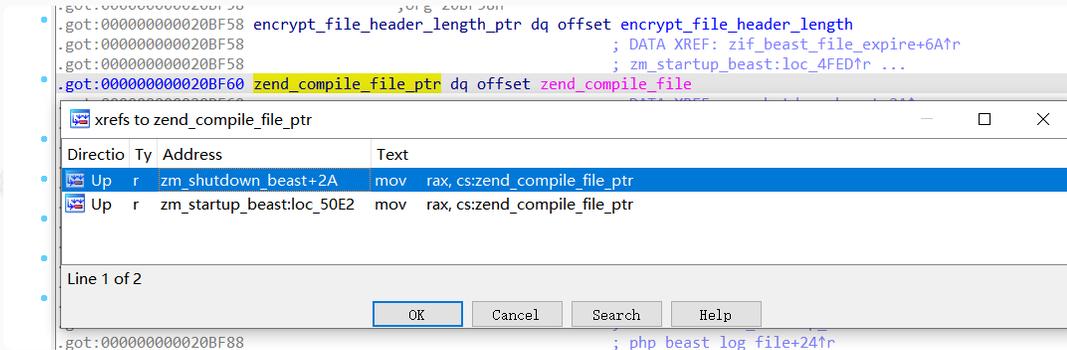
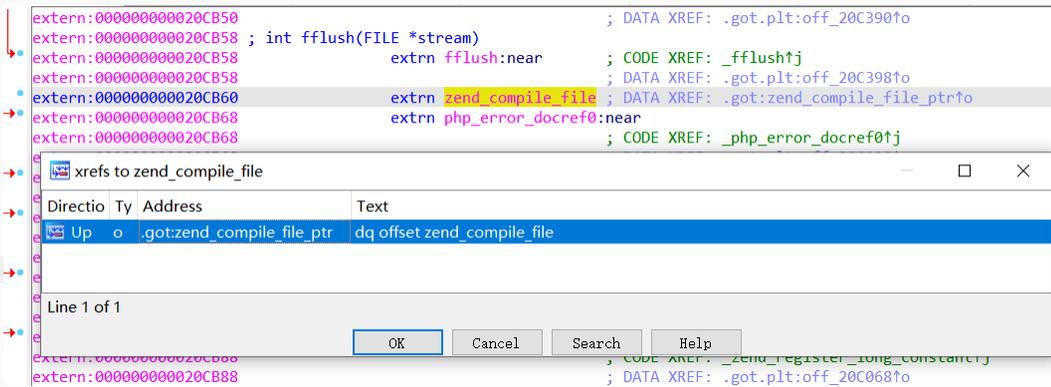
但这样有点太简单了。所以这里演示的是在没有源码的情况下使用IDA分析解密流程。



首先在导入表找到zend_compile_file

这个函数会将php文件翻译成opcode

因此大部分php加密扩展都需要hook这个函数达到拦截php文件载入和替换php文件的功能



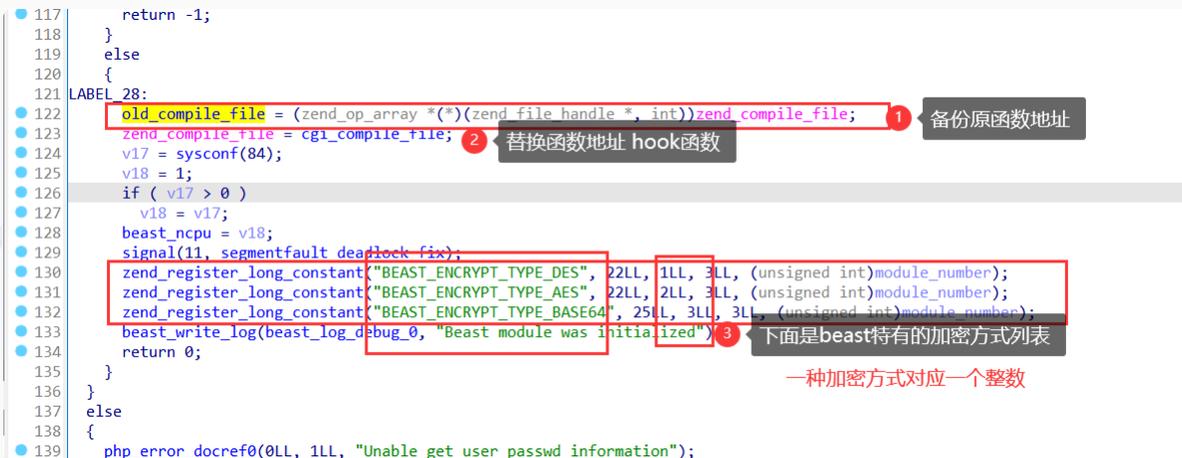
继续跟入

发现有两个函数

一般在这种php加密扩展设计时会对这个函数有两次操作：

一个是在启动时hook 这个函数，一个是在停止时恢复这个函数。

继续跟入启动hook



显然文件处理逻辑在cgi_compile_file内

```

1 zend_op_array *_fastcall cgi_compile_file(zend_file_handle *h, int type)
2 {
3     const char *filename; // rdi
4     FILE *v4; // rax
5     FILE *v5; // rbp
6     int v6; // eax
7     int v7; // eax
8     zend_stream_type v8; // eax
9     file_handler *v9; // rax
10    int v10; // edx
11    int v11; // er13
12    beast_free_buf_t *free; // rax
13    beast_ops *ops; // [rsp+8h] [rbp-40h] BYREF
14    char *buf; // [rsp+10h] [rbp-38h] BYREF
15    int free_buffer; // [rsp+18h] [rbp-30h] BYREF
16    int size[3]; // [rsp+1Ch] [rbp-2Ch] BYREF
17
18    filename = h->filename;
19    free_buffer = 0;
20    ops = 0LL;
21    v4 = fopen(filename, "rb");
22    v5 = v4;
23    if ( !v4 )
24        goto final;
25    v6 = fileno(v4);
26    v7 = decrypt_file(h->filename, v6, &buf, size, &free_buffer, &ops);
27    if ( v7 == -2 )
28    {
29        php_error_docref(0LL, 1LL, "This program was expired, please contact administrator");
30        return 0LL;
31    }
32    if ( v7 == -1 )

```

跟踪文件句柄

decrypt_file函数的参数存在文件句柄 所以这个函数应该就是文件解密函数

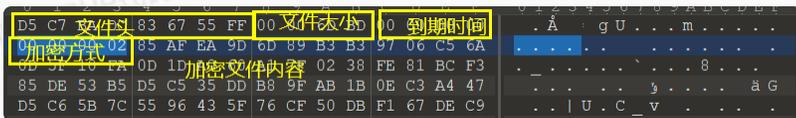
```

70     (unsigned int)v10,
71     chk);
72     return -1;
73 }
74 if ( memcmp(header, encrypt_file_header_sign, encrypt_file_header_length) )
75 {
76     if ( log_normal_file )
77     {
78         beast_write_log(beast_log_error_0, "File '%s' isn't a encrypted file", filename);
79         return -1;
80     }
81     return -1;
82 }
83 v16 = *(__DWORD *)&header[encrypt_file_header_length];
84 v17 = *(__DWORD *)&header[encrypt_file_header_length + 4];
85 v18 = *(__DWORD *)&header[encrypt_file_header_length + 8];
86 v42 = 4386;
87 v19 = (v16 << 24) | ((int)(v16 & 0xFF0000) >> 8) | ((v16 & 0xFF00) << 8) | HIBYTE(v16);
88 v20 = (v17 << 24) | ((int)(v17 & 0xFF0000) >> 8) | ((v17 & 0xFF00) << 8) | HIBYTE(v17);
89 v21 = (v18 << 24) | ((int)(v18 & 0xFF0000) >> 8) | ((v18 & 0xFF00) << 8) | HIBYTE(v18);
90 if ( beast_max_filesize > 0 && v19 > beast_max_filesize )
91 {
92     beast_write_log(
93         beast_log_error_0,
94         "File size '%d' out of max size '%d'",
95         (unsigned int)v19,
96         (unsigned int)beast_max_filesize);
97     return -1;
98 }
99 if ( v20 > 0 && v20 < beast_now_time )
100 {
101     beast_write_log(beast_log_error_0, "File '%s' was expired", filename);

```

根据代码可以看出beast 加密文件的结构

| encrypt_file_header_sign 文件头标记 (不固定 可修改) | reallen文件长度 int 4字节 | expire 到期时间 int 4字节 | entype 加密方式 int 4字节 | 加密后文件



分析文件头发现该文件加密方式为 02

```

v20 = (v17 << 24) | ((int)(v17 & 0xFF0000) >> 8) | ((v17 & 0xFF00) << 8) | HIBYTE(v17);
v21 = (v18 << 24) | ((int)(v18 & 0xFF0000) >> 8) | ((v18 & 0xFF00) << 8) | HIBYTE(v18);
if ( beast_max_filesize > 0 && v19 > beast_max_filesize )
{
    beast_write_log(
        beast_log_error_0,
        "File size `%d' out of max size `%d'",
        (unsigned int)v19,
        (unsigned int)beast_max_filesize);
    return -1;
}
if ( v20 > 0 && v20 < beast_now_time )
{
    beast_write_log(beast_log_error_0, "File `%s' was expired", filename);
    return -2;
}
b1 = chk;
encrypt_algo = beast_get_encrypt_algo(v21);
*ret_encrypt = encrypt_algo;
v35 = st_size - v31;
v22 = st_size - v31;
v23 = malloc(v22);
v24 = v23;
if ( !v23 )

```

跟入beast_get_encrypt_algo

```

1 beast_ops *__fastcall beast_get_encrypt_algo(int type)
2 {
3     unsigned int v1; // edi
4
5     v1 = type - 1;
6     if ( v1 > 3 )
7         return ops_handler_list[0];
8     else
9         return ops_handler_list[v1];
10 }

```

```

.data:00000000020C6200 ops_handler_list dq offset des_handler_ops, offset aes_handler_ops, offset base64_handler_ops
.data:00000000020C6200 ; DATA XREF: LOAD:000000000001AE80
.data:00000000020C6200 ; .got:ops_handler_list_ptr0
.data:00000000020C6200 dq offset dword_0
.data:00000000020C6400 public encrypt_file_header_sign

```

2对应的是 aes_handler_ops

```

0C560 aes_handler_ops beast_ops <offset aAesAlgo, offset aes_encrypt_handler, \
0C560 ; DATA XREF: LOAD:000000000001BC00
0C560 ; .data:ops_handler_list0
0C560 offset aes_decrypt_handler, offset aes_free_handler, 0> ; "aes-algo"

```

```

int __fastcall aes_encrypt_handler(char *inbuf, int len, char **outbuf, int *outlen)
{
    int v4; // er13
    char *v5; // r14
    __int64 v6; // rbp
    int v7; // ebx
    __int64 v8; // rdx
    uint8_t *v9; // rdx
    unsigned int v11; // [rsp+1Ch] [rbp-5Ch]
    char in[16]; // [rsp+30h] [rbp-48h] BYREF

    v4 = len / 16;
    if ( (len & 0xF) != 0 )
        v4 = len / 16 + 1;
    v11 = 16 * v4;
    v5 = (char *)malloc(16 * v4);
    if ( v5 )
    {
        if ( v4 > 0 )
        {
            v6 = 0LL;
            v7 = 0;
            do
            {
                memset(in, 0, sizeof(in));
                if ( v4 - 1 != v7 || (v8 = len % 16, (len & 0xF) == 0) )
                    v8 = 16LL;
                ++v7;
                __memcpy_chk(in, &inbuf[v6], v8, 16LL);
                v9 = (uint8_t *)&v5[v6];
                v6 += 16LL;
                AES128_ECB_encrypt((uint8_t *)in, key, v9);
            } while (v7 < v4);
        }
    }
}

```

使用了AES 128 ECB加密模式
 直接提取key参数内容
 长度刚好16位

```

:000000000020C548          align 10h
:000000000020C550          ; uint8_t key[16]
:000000000020C550  key          db 2Ch, 6Eh, 13h, 56h, 88h, 1Eh, 0E2h, 0F6h, 1Ch, 0E7h
:000000000020C550          ; DATA XREF: aes_decrypt_handler+A1fo
:000000000020C550          ; aes_encrypt_handler+ADfo
:000000000020C550          db 33h, 55h, 3, 0EFh, 46h, 3Ch

```

到这一步就成功拿到了加密密钥

使用拿到的KEY就可以解密PHP文件

附录

PHP扩展编译

Bash | [复制代码](#)

```
1 docker run -it --rm -v /mnt/hgfs/tmpssd/php-eval-hook/:/ext/ php:5.6  
/bin/bash  
2 apt-get update  
3 apt install libtool
```

Bash | [复制代码](#)

```
1 phpize
```

phpize 生成Makefile

```
# cd /mnt/hgfs/tmpssd/php-eval-hook-master  
[root@kali]~/mnt/hgfs/tmpssd/php-eval-hook-master  
# phpize  
Configuring for:  
PHP Api Version:      20190902  
Zend Module Api No:  20190902  
Zend Extension Api No: 320190902  
configure.ac:22: warning: $as_echo is obsolete; use AS_ECHO(["message"]) in  
build/php.m4:2110: PHP_CONFIG_NICE is expanded from...  
configure.ac:22: the top level  
configure.ac:165: warning: The macro 'AC_PROG_LIBTOOL' is obsolete.  
configure.ac:165: You should run autoupdate.  
build/libtool.m4:99: AC_PROG_LIBTOOL is expanded from...  
configure.ac:165: the top level
```

Bash | [复制代码](#)

```
1 ./configure --enable-evalhook
```

配置编译选项 启用扩展

最后执行make 编译扩展

编译好的扩展会放在./modules/ 目录下

使用扩展

```
1 php -d extension=扩展位置 -f 文件
```

可以重复使用-d extension 加载多个扩展

总结

在选用PHP源码保护方案时 尽量选择opcode或虚拟机方案

源代码混淆类只能对源代码获取和阅读增加一点困难 在加密扩展可被攻击者获取到时并不能起到保护作用

[PHP代码审计入门指南](#)

参考

[php内核剖析](#)

[从Zend虚拟机分析PHP加密扩展](#)

[通用加密php文件还原方法](#)

changelog

2021-12-12

1. 弃用gitbook 改用语雀
2. 补充 其他漏洞 页面 小幅度修整页面格式

2021-12-20

1. 文件操作 函数补充
2. 增加扩展阅读页面
3. 增加PHP源码解密

2021-12-28

PHP源码解密页面完成

2022-03-10

补充其他漏洞

PHP弱类型

PHP是一种弱类型语言 在定义变量时无需定义变量值类型 PHP会根据变量内容自动转换类型

例如 整数与字符串松散比较时 字符串会转成整数 "0abcdefg"==0

PHP 的松散比较 (==) <https://www.php.net/manual/zh/types.comparisons.php>

值 1\值 2	tr ue	fa lse	1	0	-1	"1"	"0"	"- 1"	nul l	arr ay()	"p h p"	""
	tru e	tru e	tru e									
	fals e	fals e	fals e									
1	tru e	tru e	tru e									
0	fals e	fals e	fals e									
-1	tru e	tru e	tru e									
"1"	tru e	tru e	tru e									

											se	
"0"	fa lse	tr ue	fa lse	tr ue	fa lse	fa lse	tr ue	fa lse	fal se	fal se	f al se	false
"- 1"	tr ue	fa lse	fa lse	fa lse	tr ue	fa lse	fa lse	tr ue	fal se	fal se	f al se	false
nul l	fa lse	tr ue	fa lse	tr ue	fa lse	fa lse	fa lse	fa lse	tru e	tru e	f al se	true
arr ay()	fa lse	tr ue	fa lse	fa lse	fa lse	fa lse	fa lse	fa lse	tru e	tru e	f al se	false
"ph p"	tr ue	fa lse	fa lse	tr ue	fa lse	fa lse	fa lse	fa lse	fal se	fal se	t ru e	false
""	fa lse	tr ue	fa lse	tr ue	fa lse	fa lse	fa lse	fa lse	tru e	fal se	f al se	true

可以使用=== 同时比较类型 避免这个问题

扩展阅读

[巧用命令注入的N种方式 \[zeddy\]](#)

[PHP7内核剖析](#)

[来读读PHP的源码x\)](#)

XSS

[内容安全策略](#)

[我的CSP绕过思路及总结 \[evoA\]](#)

反序列化

[PHP反序列化入门之寻找POP链\(一\)](#)

[PHP反序列化入门之寻找POP链\(二\)](#)

[利用 phar 拓展 php 反序列化漏洞攻击面](#)

SQL注入

[Sql预编译与模拟预编译研究 \[熊本熊本熊\]](#)

[ThinkPHP5 SQL注入漏洞 && PDO真/伪预处理分析 \[phithon\]](#)

[Docker PHP裸文件本地包含综述 \[phithon\]](#)

[PHP动态特性的捕捉与逃逸](#)

[PHP利用PCRE回溯次数限制绕过某些安全限制](#)

[谈escapeshellarg绕过与参数注入漏洞](#)

[谈一谈php://filter的妙用](#)

[代码审计实战思路之浅析PHPCMS](#)

[user.ini文件构成的PHP后门](#)

[phpwind 利用哈希长度扩展攻击进行getshell](#)

[WordPress 5.0 RCE详细分析](#)

[A New Era of SSRF – Exploiting URL Parser in Trending Programming Langua \[Orange Tsai\]](#)

[\[pdf\] A New Era of SSRF – Exploiting URL Parser in Trending Programming Langua \[Orange Tsai\]](#)

[DEF CON 26 – Orange Tsai – Breaking Parser Logic Take Your Path Normalization Off and Pop 0Days Out](#)

[\[pdf\] DEF CON 26 – Orange Tsai – Breaking Parser Logic Take Your Path Normalization Off and Pop 0Days Out](#)



总结

要比产品更懂业务

你要对项目全部业务都了如指掌 这样才能发掘业务中潜在的逻辑漏洞 出现新的业务功能要及时跟进测试

要比测试更懂流程

你要详细掌握项目中 访问路由 输入处理等过程 形成测试流程 做到快速测试 提高审计调试效率

要比开发更懂代码

熟知PHP中危险函数和过滤方法

要比架构更懂框架

熟知常见PHP框架文件结构 执行流程 访问路由

没有人比我更懂PHP审计(雾)

参考

[1] PHP手册 <https://www.php.net/manual/>

[2] ThinkPHP5.0完全开发手册 <https://www.kancloud.cn/manual/thinkphp5/>

[3] ThinkPHP3.2.3完全开发手册 <https://www.kancloud.cn/manual/thinkphp/>

[4] CodeIgniter2用户指南 <https://codeigniter.org.cn/userguide2/index.html>

[5] CodeIgniter3用户指南 <https://codeigniter.org.cn/userguide3/general/welcome.html>

[6] CodeIgniter4用户指南 https://codeigniter.org.cn/user_guide/

[7] CakePHP 4.x Strawberry Cookbook <https://book.cakephp.org/4/en/contents.html>

[8] Yii 2.0 权威指南 <https://www.yiichina.com/doc/guide/2.0>

[9] Laravel 8 中文文档 <https://learnku.com/docs/laravel/8.x>

[10] Mongodb注入攻击 (wooyun drops 3939)

<https://wooyun.js.org/drops/Mongodb%E6%B3%A8%E5%85%A5%E6%94%BB%E5%87%BB.html>

[11] LDAP注入学习小记 <https://www.cnblogs.com/0nc3/p/12063436.html>

[12] 从XML相关一步一步到XXE漏洞 <https://xz.aliyun.com/t/6887>

[13] PHP7内核剖析 <https://www.kancloud.cn/nickbai/php7>

[14] 【LDAP】LDAP注入漏洞与防御 <https://www.jianshu.com/p/d94673be9ed0>

[15] LDAP注入入门学习

<https://damit5.com/2019/08/05/LDAP%E6%B3%A8%E5%85%A5%E5%85%A5%E9%97%A8%E5%AD%A6%E4%B9%A0/>

[16] php反序列化与POP链 <https://v0w.top/2020/03/05/unsearise-POP/>

[17] Gynvail Coldwind 《PHP_LFI_rfc1867_temporary_files》 2011.3

http://gynvail.coldwind.pl/download.php?f=PHP_LFI_rfc1867_temporary_files.pdf