代码审计手记

一.常用工具

前言

在渗透测试和代码审计中，我们经常会使用到一些工具来验证漏洞。一款好的工具可以加快渗透测试和代码审计进程。

1.编辑器

Notepad++，UltraEdit，Zend Studio，Sublime Text

2.火狐插件

firebug，HackBar，JS Switch，Cookies Manager+，User Agent Switcher

3.审计工具

Seay源代码审计系统，RIPS，Fortify

在线工具：http://tool.leavesongs.com/

4.验证工具

BurpSuite，MySQL执行监控工具

二.PHP核心配置

前言

PHP配置指令作用域：限制指令是否可修改，在那里修改。每个配置项都会有一个作用域。

PHP_INI_PERDIR：可以在php.ini、httpd.conf或.htaccess文件中修改

PHP_INI_SYSTEM：可以在php.ini 和 httpd.conf 文件中修改

PHP_INI_USER：可在用户脚本(如 ini_set())或Windows注册表(PHP 5.3 起)以及.user.ini中设定

PHP_INI_ALL：可以在任何地方修改

判断配置状态：ini_get("配置名")。

1.register_globals(全局变量注册开关)

版本:本特性已自 PHP 5.3.0 起*废弃*并将自 PHP 5.4.0 起*移除*。

该选项在设置为on的情况下，会直接把用户GET，POST等方式提交上来的参数注册成全局变量并且初始化值为参数对应的值，使得提交参数可以直接在脚本中使用。register_globals在PHP版本小于4.2.3时设置为PHP_INI_ALL，从PHP5.3.0起被弃用，不推荐使用，在PHP5.4.0中移除了该选项。

当register_global = ON时，变量来源可能是各个不同的地方，比如页面的表单，Cookie等。

官方实例

```php
<?php
// 当用户合法的时候，赋值 $authorized = true
if (authenticated_user()) {
  $authorized = true;
}
// 由于并没有事先把 $authorized 初始化为 false，
// 当 register_globals 打开时，可能通过GET auth.php?authorized=1 来定义该变量值
// 所以任何人都可以绕过身份验证
if ($authorized) {
  include "/highly/sensitive/data.php";
}
?>
```

2.allow_url_include(包含远程文件)

版本:PHP 4.3.0 起可用

该选项对PHP安全的影响不可小觑。在配置为on的情况下，它可以直接包含远程文件，当存在include($var)且$var可控的情况下，可以直接控制$var变量来执行PHP代码。

allow_url_include在PHP5.2.0后默认设置为off，配置范围是PHP_INI_ALL。

配置allow_url_include为On，可以直接包涵远程文件，否则只能包涵本地文件。

前提:allow_url_fopen开启。

实例

```php
<?php
include $_GET['a'];
?>
```

3.allow_url_fopen(远程文件读取)

版本:PHP 4.0.4 起可用

该配置为On的情况下，它可以读取远程文件，当存在fopen($var)且$var变量可控的情况下，可以直接控制变量来SSRF。

allow_url_fopen默认设置为On，配置范围是PHP_INI_SYSTEM。在PHP <= 4.3.4时是PHP_INI_ALL。

实例

```php
<?php
fopen($_GET['a'], 'r');
?>
```

4.magic_quotes_gpc(魔术引号自动过滤)

版本:本特性已自 PHP 5.3.0 起*废弃*并将自 PHP 5.4.0 起*移除*。

magic_quotes_gpc在安全方面做了很大的贡献，只要它被开启，在不存编码或者其他特殊绕过的情况下，可以使得很多漏洞无法被利用。当该选项设置为On

时，会自动在GET，POST，COOKIE变量中的单引号(')，双引号(")，反斜杠()以及空字符(NULL)的前面加上反斜杠()，但是在PHP5中magic_quotes_gpc并不会过滤$_SERVER变量，导致很多类似ip，referer一类的漏洞能够利用。在PHP5.3之后的不推荐使用magic_quotes_gpc，PHP5.4之后干脆被取消，所以你下载PHP 5.4之后的版本并打开配置文件会发现找不到这个配置选项。在PHP版本小于4.2.3时，配置范围是PHP_INI_ALL；在PHP版本大于4.2.3时，是PHP_INI_PERDIR。

实例

```
<?php
echo $_GET['a'];
?>
```

5.open_basedir(可访问目录)

版本:目前支持所有PHP

将 PHP 所能打开的文件限制在指定的目录树，包括文件本身。本指令*不受*安全模式打开或者关闭的影响。一般只需要设置Web文件目录即可，如果需要加载外部脚本，也需要把脚本所在目录路径加入到open_basedir指令中，多个目录以分号(;)分割(windows)和冒号(:)分割(linux)。使用 open_basedir 需要注意的一点是，指定的限制实际上是前缀，而不是目录名。例如，如果配置 open_basedir =/www/a，那么目录/www/a 和 /www/ab 都是可以访问的。所以如果要将访问仅限制在指定的目录内，请用斜线结束路径名，设置成：open_basedir = /www/a/。

当 open_basedir 配置目录后，执行脚本访问其他文件都需要验证文件路径，因此在执行效率上面也会有一定的影响。该指令的配置范围在 PHP 版本小于 5.2.3 时是PHP_INI_SYSTEM，在 PHP 版本大于等于 5.2.3 是 PHP_INI_ALL。

注意:配置多个目录时，需要将目录父目录或者范围较大的目录放置前面。

实例

```
<?php
$dir = dir($_GET['dir']);
while (($file = $dir->read()) !== false) {
  echo $file . "<br />";
}
$dir->close();
?>
```

6.disable_functions(禁用函数)

版本:PHP 4.0.1 起可用

本指令可以使你出于安全的理由禁用某些类。用逗号(,)分隔类名。disable_classes 不受安全模式的影响。 本指令只能设置在 php.ini 中。例如不能将其设置httpd.conf。

当你想用本指令禁止一些危险函数时，切记要把 dl()函数也加到禁止列表，因为攻击者可以利用 dl() 函数来加载自定义的 PHP 扩展以突破disable_functions 指令的限制。

例如：disable_functions=phpinfo,eval,passthru,exec,system。

实例

```
<?php
echo "disable_functions:" . ini_get("disable_functions") . "<br />";
?>
```

7.display_errors 和 error_reporting(错误显示)

版本:目前支持所有PHP

该选项设置是否将错误信息作为输出的一部分显示到屏幕，或者对用户隐藏而不显示。

在调试 PHP 的时候，通常都把 PHP 错误显示打开，但是在生产环境中，建议关闭 PHP 错误回显，即设置display_errors=off，以避免带来一些安全隐患。

在设置 display_errors=on 时，还可以配置的一个指令是 error_reporting，这个选项用来配置错误显示的级别，可使用数字也可使用内置常量配置，见下图。这两个指令的配置范围都是 PHP_INI_ALL。

数字格式	常量格式	数字格式	常量格式
1	E_ERROR	128	E_COMPILE_WARNING
2	E_WARNING	256	E_USER_ERROR
4	E_PARSE	512	E_USER_WARNING
8	E_NOTICE	1024	E_USER_NOTICE
16	E_CORE_ERROR	2047	E_ALL
32	E_CORE_WARNING	2048	E_STRICT
64	E_COMPILE_ERROR		

实例

```
<?php
error_reporting(1);
?>
```

8.其他常用指令

指　令	可配置范围	说　明
safe_mode_gid	PHP_INI_SYSTEM	以安全模式打开文件时默认使用 UID 来比对；设置本指令为 on 时使用 GID 做宽松的比对
expose_php	php.ini only	是否在服务器返回信息 HTTP 头显示 PHP 版本
max_execution_time	PHP_INI_ALL	每个脚本最多执行秒数
memory_limit	PHP_INI_ALL	每个脚本能够使用的最大内存数量
log_errors	PHP_INI_ALL	将错误输入到日志文件
log_errors_max_len	PHP_INI_ALL	设定 log_errors 的最大长度
variables_order	PHP_INI_PERDIR	此指令描述了 PHP 注册 GET、POST、Cookie、环境和内置变量的顺序，注册使用从左往右的顺序，新的值会覆盖旧的值 .
post_max_size	PHP_INI_PERDIR	PHP 可以接受的最大的 POST 数据大小
auto_prepend_file	PHP_INI_PERDIR	在任何 PHP 文档之前自动包含的文件
auto_append_file	PHP_INI_PERDIR	在任何 PHP 文档之后自动包含的文件
extension_dir	PHP_INI_SYSTEM	可加载的扩展 (模块) 的目录位置
file_uploads	PHP_INI_SYSTEM	是否允许 HTTP 文件上传
upload_tmp_dir	PHP_INI_SYSTEM	对于 HTTP 上传文件的临时文件目录
upload_max_filesize	PHP_INI_SYSTEM	允许上传的最大文件大小

三.代码审计思路

前言

代码审计工具加快了漏洞点的寻找，只有工具是不可能手到擒来的。如何高效地挖掘到最有质量的漏洞，仍是需要一定的方法论与思路的体验。

1.通读全文代码

只有了解整个应用的业务逻辑，才能挖掘到更多更有价值的漏洞。当然通读全文代码也有一定的技巧，并不是真的从index.php开始逐个读完。

首先我们要看程序的大体代码结构，如主目录有哪些文件，模块目录有哪些文件，插件目录有哪些文件，除了关注有哪些文件，还要注意文件的大小、创建时间。我们根据这些文件的命名就可以大致知道这个程序实现了哪些功能，核心文件是哪些。

1）函数集文件，通常命令中包含functions或者common等关键字，这些文件里面是一些公共的函数，提供给其他文件统一调用，所以大多数文件都会在文件头部包含到其他文件。寻找这些文件的一个非常好用的技巧是去打开入口文件。

2）配置文件，通常命名中包括config关键字，配置文件包括web程序运行必须的功能性配置选项以及数据库等配置信息。

3）安全过滤文件，安全过滤文件对我们做代码审计至关重要，关系到我们挖掘到的可疑点能不能利用，通常命名中有filter，safe，check等关键字。

4）index文件，index是一个程序的入口文件，所以通常我们只要读一遍index文件就可以大致了解整个程序的架构、运行的流程、包含到的文件。而不同目录的 index 文件也有不同的实现方式，建议最好先将几个核心目录的index 文件都简单读一遍。

总结

前面介绍了几类我们需要注意的文件，这些文件可以帮助我们更有针对性地去了解代码的结构和功能。对于接触代码审计不久的小伙伴们，可能会经常遇到各种框架，这时候会被搞得晕头转向。建议大家不要以框架入手，可以先尝试一些chinaz，seacms等web程序入手。并且一定要多找几套程序通读全文代码，这样才能总结经验。提高代码审计的能力。对于PHP比较熟悉之后，再去读一些TP，Yii，Zend这样的框架。

通读全文代码的好处显而易见，可以更好地了解程序的架构以及业务逻辑，能够挖掘到更多、更高质量的漏洞。而缺点就是花费的时间比较多，如果程序比较大，读起来也会比较累。

2.敏感关键字回溯参数传递过程

根据敏感函数来逆向追踪参数的传递过程，是目前使用得最多的方式之一，因为大多数漏洞是由于函数的使用不当造成的。另外非函数使用不当的漏洞，如 SQL 注入，也有一些特征，比如 Select、Insert 等，再结合 From 和 Where 等关键字，我们就可以判断这是否是一条 SQL 语句，通过对字符串的识别分析，就能判断这个 SQL 语句里面的参数有没有使用单引号过滤，或者根据我们的经验来判断。像 HTTP 头里面的 HTTP_CLIENT_IP 和 HTTP_X*FORWORDFOR 等获取到的 IP 地址经常没有安全过滤就直接拼接到 SQL 语句中，并且由于它们是在 $*SERVER 变量中不受 GPC 的影响，那我们就可以去查找 HTTP_CLIENT_IP 和 HTTP_X_FORWORDFOR 关键字来快速寻找漏洞。

总结

这种方式的优点是只需要搜索相应敏感关键字，即可以快速地挖掘想要的漏洞，具有可定向挖掘和高效，高质量的优点。

其缺点为由于没有通读代码，对程序的整体框架了解不够深入，在挖掘漏洞时定位利用点会话费一点时间，另外对逻辑漏洞挖掘覆盖不到。

3.功能点定向审计

根据功能点定向审计是最实用的方法。首先安装好并且运行程序，到处点点，浏览一下，看下程序有哪些功能，这些功能的程序文件分别是怎么样的，是独立的模块还是以插件形式存在，或者是写在一个通用类里面，然后多处调用。在了解这些功能的存在形式后，可以先寻找经常会出问题的功能点，简单黑盒测试一下，如果没有发现很普通、很常见的漏洞，再去读这个功能的代码，这样我们读起来就可以略过一些刚才黑盒测试过的点，提高审计速度。

根据经验，经常会出现漏洞点：

1）文件上传功能

2）文件管理功能

3）登陆认证功能

4）找回密码功能

总结

对于功能点上的漏洞查找需要我们多读代码和多看一些网上的功能点爆出的漏洞，通过黑盒测试寻找对应的漏洞的突破口。

4.查找可控变量，正向追踪变量传递过程

在挖掘漏洞的过程中，我们常会对代码的数据流进行追踪。对于克控的变量，跟踪其在代码中的轨迹，当所跟踪的变量被危险函数所使用的时候。此时就可能存在漏洞点。

bypupil

2017-7-14