目录

概述

代码审核，是对应用程序源代码进行系统性检查的工作。它的目的是为了找到并且修复应用程序在开发阶段存在的一些漏洞或者程序逻辑错误，避免程序漏洞被非法利用给企业带来不必要的风险。

代码审核不是简单的检查代码，审核代码的原因是确保代码能安全的做到对信息和资源进行足够的保护，所以熟悉整个应用程序的业务流程对于控制潜在的风险是非常重要的。审核人员可以使用类似下面的问题对开发者进行访谈，来收集应用程序信息。

应用程序中包含什么类型的敏感信息，应用程序怎么保护这些信息的？

应用程序是对内提供服务，还是对外？哪些人会使用，他们都是可信用户么？

应用程序部署在哪里？

应用程序对于企业的重要性？

最好的方式是做一个 checklist，让开发人员填写。Checklist 能比较直观的反映应用程序的信息和开发人员所做的编码安全，它应该涵盖可能存在严重漏洞的模块，例如：数据验证、身份认证、会话管理、授权、加密、错误处理、日志、安全配置、网络架构。

输入验证和输出显示

大多数漏洞的形成原因主要都是未对输入数据进行安全验证或对输出数据未经过安全处理，比较严格的数据验证方式为：

对数据进行精确匹配

接受白名单的数据

拒绝黑名单的数据

对匹配黑名单的数据进行编码

在 PHP 中可由用户输入的变量列表如下：

$_SERVER

$_GET

$_POST

$_COOKIE

$_REQUEST

$_FILES

$_ENV

$_HTTP_COOKIE_VARS

$_HTTP_ENV_VARS

$_HTTP_GET_VARS

$_HTTP_POST_FILES

$_HTTP_POST_VARS

$_HTTP_SERVER_VARS

我们应该对这些输入变量进行检查

命令注入

PHP 执行系统命令可以使用以下几个函数：system、exec、passthru、``、shell_exec、

popen、proc_open、pcntl_exec

我们通过在全部程序文件中搜索这些函数，确定函数的参数是否会因为外部提交而改变，检查这些参数是否有经过安全处理。

防范方法：

使用自定义函数或函数库来替代外部命令的功能

使用 escapeshellarg 函数来处理命令参数

使用 safe_mode_exec_dir 指定可执行文件的路径

跨站脚本

反射型跨站常常出现在用户提交的变量接受以后经过处理，直接输出显示给客户端；存储型跨站常常出现在用户提交的变量接受过经过处理后，存储在数据库里，然后又从数据库中读取到此信息输出到客户端。输出函数经常使用：echo、print、printf、vprintf、<%=$test%>

对于反射型跨站，因为是立即输出显示给客户端，所以应该在当前的 php 页面检查变量被客户提交之后有无立即显示，在这个过程中变量是否有经过安全检查。

对于存储型跨站，检查变量在输入后入库，又输出显示的这个过程中，变量是否有经过安全检查。

防范方法：

如果输入数据只包含字母和数字，那么任何特殊字符都应当阻止

对输入的数据经行严格匹配，比如邮件格式，用户名只包含英文或者中文、下划线、连字符

对输出进行 HTML 编码，编码规范

< <

> >

((

))

# #

& &

" "

' '

` %60

文件包含

PHP 可能出现文件包含的函数：include、include_once、require、require_once、show_source、highlight_file、readfile、file_get_contents、fopen、file

防范方法：

对输入数据进行精确匹配，比如根据变量的值确定语言 en.php、cn.php，那么这两个文件放在同一个目录下’language/’.$_POST[‘lang’].’.php’，那么检查提交的数据是否是 en 或者 cn 是最严格的，检查是否只包含字母也不错

通过过滤参数中的/、..等字符

代码注入

PHP 可能出现代码注入的函数：eval、preg_replace+/e、assert、call_user_func、call_user_func_array、create_function

查找程序中程序中使用这些函数的地方，检查提交变量是否用户可控，有无做输入验证

防范方法：

输入数据精确匹配

白名单方式过滤可执行的函数

SQL 注入

SQL 注入因为要操作数据库，所以一般会查找 SQL 语句关键字：insert、delete、update、

select，查看传递的变量参数是否用户可控制，有无做过安全处理

防范方法：

使用参数化查询

XPath 注入

Xpath 用于操作 xml，我们通过搜索 xpath 来分析，提交给 xpath 函数的参数是否有经过安全处理

防范方法：

对于数据进行精确匹配

HTTP 响应拆分

PHP 中可导致 HTTP 响应拆分的情况为：使用 header 函数和使用$_SERVER 变量。注意 PHP 的高版本会禁止 HTTP 表头中出现换行字符，这类可以直接跳过本测试。

防范方法：

精确匹配输入数据

检测输入输入中如果有\r 或\n，直接拒绝

文件管理

PHP 的用于文件管理的函数，如果输入变量可由用户提交，程序中也没有做数据验证，可能成为高危漏洞。我们应该在程序中搜索如下函数：copy、rmdir、unlink、delete、fwrite、chmod、fgetc、fgetcsv、fgets、fgetss、file、file_get_contents、fread、readfile、ftruncate、file_put_contents、fputcsv、fputs，但通常 PHP 中每一个文件操作函数都可能是危险的。

http://ir.php.net/manual/en/ref.filesystem.php

防范方法：

对提交数据进行严格匹配

限定文件可操作的目录

文件上传

PHP 文件上传通常会使用 move_uploaded_file，也可以找到文件上传的程序进行具体分析

防范方式：

使用白名单方式检测文件后缀

上传之后按时间能算法生成文件名称

上传目录脚本文件不可执行

注意%00截断

变量覆盖

PHP 变量覆盖会出现在下面几种情况：

遍历初始化变量

例：

```
foreach($_GET as $key => $value)

$$key = $value;
```

函数覆盖变量：parse_str、mb_parse_str、import_request_variables

Register_globals=ON 时，GET 方式提交变量会直接覆盖

防范方法：

设置 Register_globals=OFF

不要使用这些函数来获取变量

动态函数

当使用动态函数时，如果用户对变量可控，则可导致攻击者执行任意函数。

例：

```
<?php
$myfunc=$_GET['myfunc'];
$myfunc();
?>
```

防御方法：

不要这样使用函数

会话安全

HTTPOnly 设置

session.cookie_httponly = ON 时，客户端脚本(JavaScript 等)无法访问该 cookie，打开该指令可以有效预防通过 XSS 攻击劫持会话 ID

domain 设置

检查 session.cookie_domain 是否只包含本域，如果是父域，则其他子域能够获取本域的 cookies

path 设置

检查 session.cookie_path，如果网站本身应用在/app，则 path 必须设置为/app/，才能保证安全

cookies 持续时间

检查 session.cookie_lifetime，如果时间设置过程过长，即使用户关闭浏览器，攻击者也会危害到帐户安全

secure 设置

如果使用 HTTPS，那么应该设置 session.cookie_secure=ON，确保使用 HTTPS 来传输 cookies

session 固定

如果当权限级别改变时（例如核实用户名和密码后，普通用户提升到管理员），我们就应该修改即将重新生成的会话 ID，否则程序会面临会话固定攻击的风险。

CSRF

跨站请求伪造攻击，是攻击者伪造一个恶意请求链接，通过各种方式让正常用户访问后，会以用户的身份执行这些恶意的请求。我们应该对比较重要的程序模块，比如修改用户密码，添加用户的功能进行审查，检查有无使用一次性令牌防御 csrf 攻击。

加密

明文存储密码

采用明文的形式存储密码会严重威胁到用户、应用程序、系统安全。

密码弱加密

使用容易破解的加密算法，MD5加密已经部分可以利用 md5破解网站来破解

密码存储在攻击者能访问到的文件

例如：保存密码在 txt、ini、conf、inc、xml 等文件中，或者直接写在 HTML 注释中

认证和授权

用户认证

检查代码进行用户认证的位置，是否能够绕过认证，例如：登录代码可能存在表单注入。

检查登录代码有无使用验证码等，防止暴力破解的手段

函数或文件的未认证调用

一些管理页面是禁止普通用户访问的，有时开发者会忘记对这些文件进行权限验证，导致漏洞发生

某些页面使用参数调用功能，没有经过权限验证，比如 index.php?action=upload

密码硬编码

有的程序会把数据库链接账号和密码，直接写到数据库链接函数中。

随机函数

rand()

rand()最大随机数是32767，当使用 rand 处理 session 时，攻击者很容易破解出 session，建议使用 mt_rand()

mt_srand()和 mt_rand()

PHP4 和 PHP5<5.2.6，这两个函数处理数据是不安全的。在 web 应用中很多使用 mt_rand 来处理随机的 session，比如密码找回功能等，这样的后果就是被攻击者恶意利用直接修改密码。

特殊字符和多字节编码

多字节编码

PHP 危险函数

缓冲区溢出

confirm_phpdoc_compiled

影响版本：

phpDocumentor phpDocumentor 1.3.1

phpDocumentor phpDocumentor 1.3 RC4

phpDocumentor phpDocumentor 1.3 RC3

phpDocumentor phpDocumentor 1.2.3

phpDocumentor phpDocumentor 1.2.2

phpDocumentor phpDocumentor 1.2.1

phpDocumentor phpDocumentor 1.2

mssql_pconnect/mssql_connect

影响版本：PHP <= 4.4.6

crack_opendict

影响版本：PHP = 4.4.6

snmpget

影响版本：PHP <= 5.2.3

ibase_connect

影响版本：PHP = 4.4.6

unserialize

影响版本：PHP 5.0.2、PHP 5.0.1、PHP 5.0.0、PHP 4.3.9、PHP 4.3.8、PHP 4.3.7、PHP 4.3.6、PHP 4.3.3、PHP 4.3.2、PHP 4.3.1、PHP 4.3.0、PHP 4.2.3、PHP 4.2.2、PHP 4.2.1、PHP 4.2.0、PHP 4.2-dev、PHP 4.1.2、PHP 4.1.1、PHP 4.1.0、PHP 4.1、PHP 4.0.7、PHP 4.0.6、PHP 4.0.5、PHP 4.0.4、PHP 4.0.3pl1、PHP 4.0.3、PHP 4.0.2、PHP 4.0.1pl2、PHP 4.0.1pl1、PHP 4.0.1

session_destroy()删除文件漏洞

影响版本：不祥，需要具体测试

测试代码如下：

```
<?php
session_save_path('./');
session_start();
if($_GET['del']) {
session_unset();
session_destroy();
}else{
$_SESSION['do']=1;
echo(session_id());
print_r($_SESSION);
}
?>
```

当我们提交 cookie:PHPSESSIONID=/../1.php，相当于删除了此文件

unset()-zend_hash_del_key_or_index 漏洞

zend_hash_del_key_or_index PHP4 小于 4.4.3 和 PHP5 小于 5.1.3，可能会导致 zend_hash_del 删除了错误的元素。当 PHP 的 unset()函数被调用时，它会阻止变量被 unset。

信息泄露

phpinfo

如果攻击者可以浏览到程序中调用 phpinfo 显示的环境信息，会为进一步攻击提供便利

PHP 环境

open_basedir 设置

open_basedir 能限制应用程序能访问的目录，检查有没有对 open_basedir 进行设置，当然有的通过 web 服务器来设置，例如：apache 的 php_admin_value，nginx+fcgi 通过 conf 来控制 php 设置

allow_url_fopen 设置

如果 allow_url_fopen=ON，那么 php 可以读取远程文件进行操作，这个容易被攻击者利用

allow_url_include 设置

如果 allow_url_include=ON，那么 php 可以包含远程文件，会导致严重漏洞

safe_mode_exec_dir 设置

这个选项能控制 php 可调用的外部命令的目录，如果 PHP 程序中有调用外部命令，那么指定外部命令的目录，能控制程序的风险

magic_quote_gpc 设置

这个选项能转义提交给参数中的特殊字符，建议设置 magic_quote_gpc=ON

register_globals 设置

开启这个选项，将导致 php 对所有外部提交的变量注册为全局变量，后果相当严重

safe_mode 设置

safe_mode 是 PHP 的重要安全特性，建议开启

session_use_trans_sid 设置

如果启用 session.use_trans_sid，会导致 PHP 通过 URL 传递会话 ID，这样一来，攻击者就更容易劫持当前会话，或者欺骗用户使用已被攻击者控制的现有会话。

display_errors 设置

如果启用此选项，PHP 将输出所有的错误或警告信息，攻击者能利用这些信息获取 web 根路径等敏感信息

expose_php 设置

如果启用 expose_php 选项，那么由 PHP 解释器生成的每个响应都会包含主机系统上所安装的 PHP 版本。了解到远程服务器上运行的 PHP 版本后，攻击者就能针对系统枚举已知的盗取手段，从而大大增加成功发动攻击的机会。

参考文档：

https://www.fortify.com/vulncat/zh_CN/vulncat/index.html

http://secinn.appspot.com/pstzine/read?issue=3&articleid=6

http://riusksk.blogbus.com/logs/51538334.html

http://www.owasp.org/index.php/Category:OWASP_Code_Review_Project