Web 安全- PHP 开发者如何做代码审查? 常规漏洞篇

前言

工欲善其事，必先利其器。我们做代码审计之前选好工具也是十分必要的。下面我给大家介绍两款代码审计中比较好用的工具。

一、审计工具介绍

PHP 代码审计系统— RIPS

功能介绍

RIPS是一款基于PHP开发的针对PHP代码安全审计的软件。

另外，它也是一款开源软件，由国外安全研究员Johannes Dahse开发，程序只有450KB，目前能下载到的最新版是0.55。

在写这段文字之前笔者特意读过它的源码，它最大的亮点在于调用了PHP内置解析器接口 `token_get_all` ，

并且使用Parser做了语法分析，实现了跨文件的变量及函数追踪，扫描结果中非常直观地展示了漏洞形成及变量传递过程，误报率非常低。

RIPS能够发现SQL注入、XSS跨站、文件包含、代码执行、文件读取等多种漏洞，支持多种样式的代码高亮。比较有意思的是，它还支持自动生成漏洞利用。

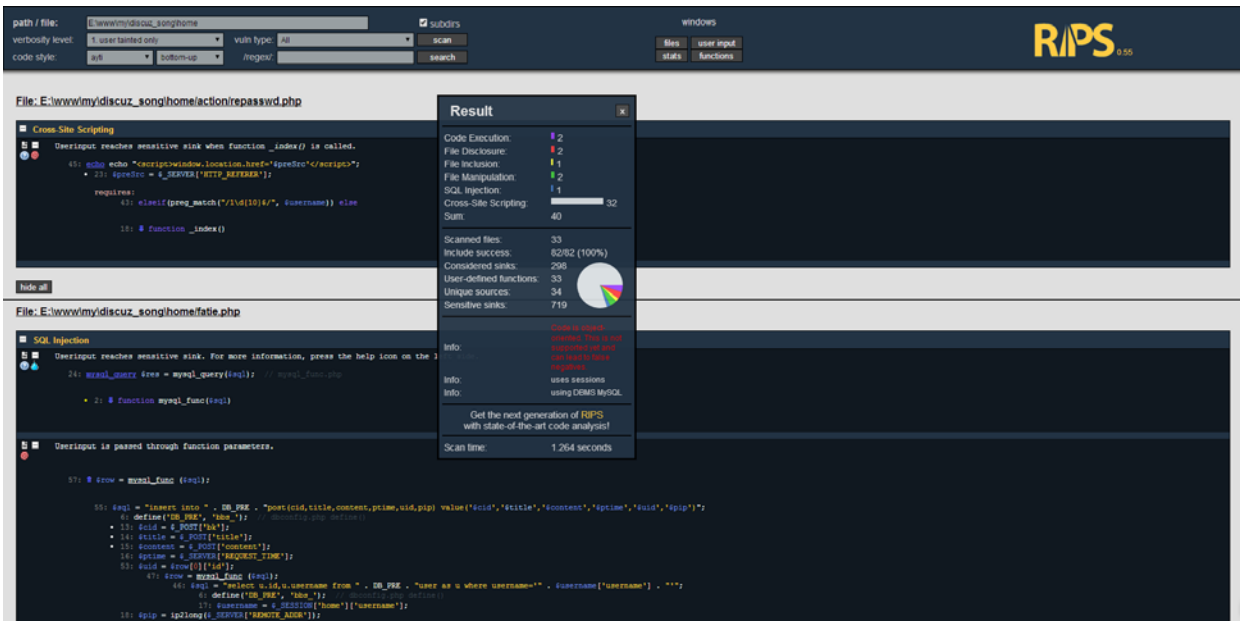
安装方法

下载地址：https://jaist.dl.sourceforge.net/project/rips-scanner/rips-0.55.zip.

解压到任意一个PHP的运行目录

在浏览器输入对应网址，可以通过下图看到有一个path 在里面填写你要分析的项目文件路径，点击 scan.

界面截图



seay 源代码审计系统

功能介绍

这些是seay 第一个版本的部分功能，现在最新版本是2.1。

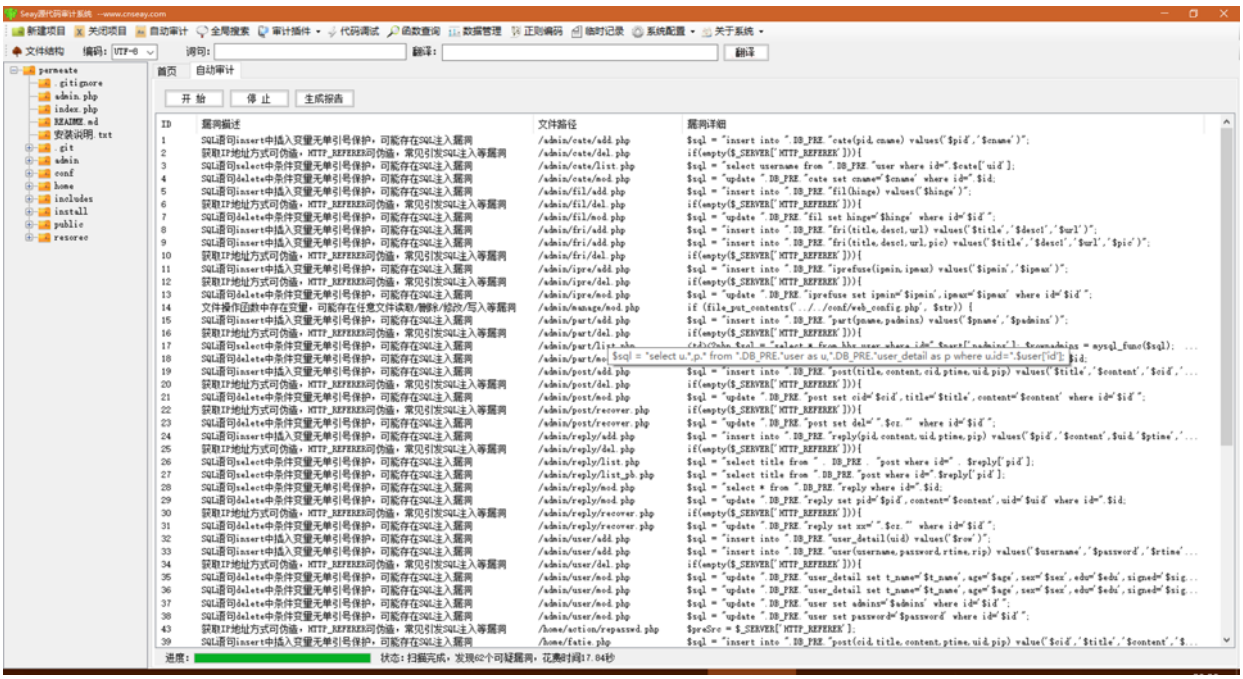
1. 傻瓜化的自动审计。
2. 支持php代码调试。
3. 函数/变量定位。
4. 生成审计报告。
5. 自定义审计规则。
6. mysql数据库管理。
7. 黑盒敏感信息泄露一键审计。
8. 支持正则匹配调试。
9. 编辑保存文件。
10. POST数据包提交。

安装方法

安装环境需要 .NET2.0以上版本环境才能运行，下载安装包之后点击下一步就安装好了，非常的简便。

安装包下载地址：http://enkj.jb51.net:81/201408/tools/Seayydmsjxt(jb51.net).rar

操作界面的截图



二、代码审计实战

前言

通过刚才安装的两个审计工具运行后我们可以发现，会分析出很多隐藏的漏洞，那下面我们看看其中的SQL注入、XSS、CSRF产生的原因,通过原因来分析如何去审计代码。

SQL注入

前言

SQL注入漏洞一直是web系统漏洞中占比非常大的一种漏洞，下面我们来看看SQL注入的几种方式。

SQL 注入漏洞分类

从利用方式角度可以分为两种类型:常规注入、宽字节注入。

常规注入方式，通常没有任何过滤，直接把参数存放到了SQL语句当中，如下图。

```php
<?php

$connDb = mysql_connect( server: 'localhost', username: 'root', password: 'root');

mysql_select_db( database_name: "users", $connDb);

//接收用户ID
$uid = $_GET['id'];
//构造查询SQL语句
$sql = " select * from user where user_id = $uid";

$userInfo = mysql_query($sql, $connDb);

var_dump(mysql_fetch_row($userInfo));

```

非常容易发现，现在开发者一般都会做一些过滤，比如使用addslashes()，但是过滤有时候也不一定好使。

编码注入方式

宽字节注入，这个是怎么回事呢？

在实际环境中程序员一般不会写上面类似的代码，一般都会用addslashes()等过滤函数对从web传递过来的参数进行过滤。不过有句话叫做，道高一尺魔高一丈，我们看看白帽子是怎么突破的。用PHP连接MySQL的时候，当设置 character_set_client=gbk时候会导致一个编码漏洞。我们知道addslashes() 会把参数 1’ 转换成 1\’,而我们提交参数 1%df’ 时候会转成 1縗’，那我们输入 1%df’ or 1=1%23时候，会被转换成 1縗’ or 1=1#’。

简单来说%df’会被过滤函数转义为%df\’ ，%df\’ = %df%5c%27 在使用gbk编码的时候会认为%df%5c是一个宽字节%df%5c%27=縗’，这样就会产生注入。

那如何防御这个宽字节呢？我希望大家开发网站尽量使用UTF8编码格式，如果转换麻烦，最安全的方法就是使用PDO预处理。挖掘这种漏洞主要是检查是否使用了gbk，搜索 `guanjianc character_set_client=gbk 和mysql_set_chatset('gbk')` 。

二次urldecode注入，这中方式也是因为使用了urldecode不当所引起的漏洞。

我们刚才知道了 addslashes()函数可以防止注入，他会在(‘)、(“)、()前面加上反斜杠来转义。

那我们假设我们开启了GPC，我们提交了一个参数，/test.php?uid=1%2527,因为参数中没有单引号，所以第一次解码会变成uid=1%27,%25解码出来就是%，

这时候程序里如果再去使用urldecode来解码，就会把%27解码成单引号(‘)，最终的结果就是uid=1’.

我们现在知道了原有是因为urldecode引起的，我们可以通过编辑器的搜索urldecode和rawurldecode找到二次url漏洞。

从漏洞类型区分可以分为三种类型：

1. 可显

   攻击者可以直接在当前界面内容中获取想要获得的内容。

2. 报错

   数据库查询返回结果并没有在页面中显示，但是应用程序将数据库报错信息打印到了页面中。

   所以攻击者可以构造数据库报错语句，从报错信息中获取想要获得的内容，所以我建议在数据库类中设置不抛出错误信息。

3. 盲注

数据库查询结果无法从直观页面中获取攻击者通过使用数据库逻辑或使数据库库执行延时等方法获取想要获得的内容。

SQL 注入漏洞挖掘方法

针对上面提到的利用漏洞方法，总结了以下的挖掘方法：

1. 参数接收位置，检查是否有没过滤直接使用$GET、_POST、$_COOKIE参数的。
2. SQL语句检查，搜索关键词 `select update insert` 等SQL语句关键处，检查SQL语句的参数是否可以被控制。
3. 宽字节注入，如果网站使用的GBK编码情况下，搜索 `guanjianc character_set_client=gbk` 和 `mysql_set_chatset('gbk')` 就行。
4. 二次urldecode注入，少部分情况，gpc可以通过编辑器的搜索urldecode和rawurldecode找到二次url漏洞。

SQL 注入漏洞防范方法

虽然SQL注入漏洞非常多，但是防范起来却挺简单的，下面介绍几个过滤函数和类:

- gpc/rutime 魔术引号
- 过滤函数和类
    - addslashes
    - mysql_real_escape_string
    - intval
- PDO 预处理

XSS跨站

前言

XSS又叫CSS (Cross Site Script) ，跨站脚本攻击。它指的是恶意攻击者往Web页面里插入恶意html代码，当用户浏览该页之时，嵌入其中Web里面的html代码会被执行，从而达到恶意的特殊目的。

XSS属于被动式的攻击，因为其被动且不好利用，所以许多人常呼略其危害性。在WEB2.0时代，强调的是互动，使得用户输入信息的机会大增，在这个情况下，我们作为开发者，在开发的时候，要提高警惕。

xss 漏洞分类

1. 反射型，危害小，一般

   反射型XSS原理：就是通过给别人发送带有恶意脚本代码参数的URL，当URL地址被打开时，特定的代码参数会被HTML解析，执行，如此就可以获取用户的COOIKE，进而盗号登陆。比如hack甲构造好修改密码的URL并把密码修改成123，但是修改密码只有在登陆方乙才能修改，乙在登陆的情况下点击甲构造好的URL将直接在不知情的情况下修改密码。

   特点是：非持久化，必须用户点击带有特定参数的链接才能引起。

2. 存储型，危害大，影响时间长

   存储型XSS原理，假设你打开了一篇正常的文章页面，下面有评论功能。这个时候你去评论了一下，在文本框中输入了一些JavaScript代码，提交之后,你刷新这个页面后发现刚刚提交的代码又被原封不动的返回来并且执行了。

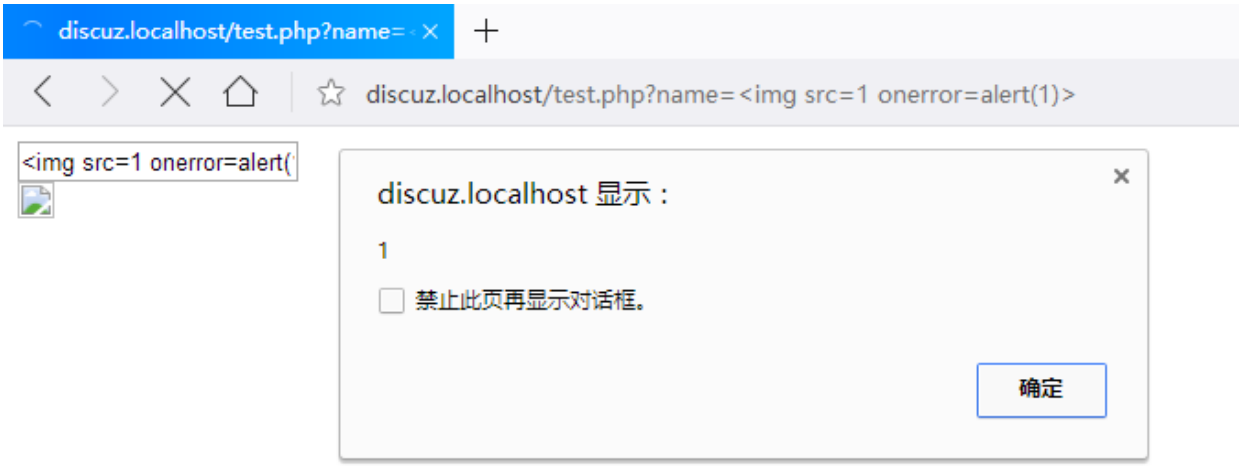
   这个时候你会想,我要写一段JavaScript代码获取cookie信息，然后通过ajax发送到自己的服务器去。构造好代码后你把链接发给其他的朋友，或者网站的管理员，他们打开JavaScript代码就执行了，你服务器就接收到了sessionid，你就可以拿到他的用户权限了。

3.dom型，特殊的一种

dom型xss是因为JavaScript执行了dom操作，所造成的xss漏洞，具体如下图。可以看到虽然经过html转义了，但是这块代码在返回到html中，又被JavaScript作为dom元素操作。那当我输入 `?name=<img src=1 onerror=alert(1)>` 的时候依然会存在XSS漏洞。

```php
<?php
error_reporting(0);
$name = htmlspecialchars($_GET["name"]);
?>
<input id="username" type="text" value="<?php echo $name;?>" />
<div id="content"></div>

<script type="text/javascript">
// 获取输入的名称，并且输出在content内。导致了一个dom-xss。
var username = document.getElementById("username");
var content = document.getElementById("content");
content.innerHTML = username.value;
</script>
```



xss 漏洞挖掘方法

根据上面的一些特点，可以总结出几个分析出几个挖掘方法：

1. 数据接收位置，检查$_GET、_POST、$_COOKIE是否经过转义。
2. 常见的反射型xss搜索这种类似位置发现次数较多。
3. 而存储型在文章，评论出现比较多。

XSS 漏洞防范方法

1. 转义html实体，有两种方式：在入口和出口,我建议是在入口处转义，防止出口位置取出来的时候忘记转义，如果已经在入口转义了，出口位置就不用再次转义。
2. 在富文本编辑器中，经常会用到一些元素的属性，比如上图的onerror，那我们还需对元素的属性建立黑白名单。
3. httpOnly 即使存在xss漏洞，可以把危害大大降低。

CSRF漏洞

CSRF 漏洞介绍

CSRF（Cross-site request forgery）跨站请求伪造，通常缩写为CSRF或者XSRF，是一种对网站的恶意利用。听起来像跨站脚本（XSS），但它与XSS非常不同，XSS利用站点内的信任用户。

而CSRF则通过伪装来自受信任用户的请求来利用受信任的网站。与XSS攻击相比，CSRF攻击往往不大流行（因此对其进行防范的资源也相当稀少）和难以防范，所以被认为比XSS更具危险性。

csrf主要用来做越权操作，而且csrf一直没有被关注起来，所以很多程序现在也没有相关的防范措施。

CSRF 案例

我们来看下面的一段代码,这个表单当被访问到的时候，用户就退出了登录。假设有一个转账的表单，只需要填写对方的用户名，和金额就可以，那如果我提前把URL构造好，发给受害者，当点击后，钱就被转走了。或者我把这个URL放到我的网页中，通过 `<img src="我构造的URL" ` ，当其他人打开我的网址后，就中招了。

```php
<?php
    session_start();
    //删除session信息,退出登录
    unset($_SESSION['home']['username']);
    setcookie('adminusername','',time()-1,'/');
    session_destroy();
    header("location:../index.php");
?>
```

CSRF漏洞挖掘方法

通过上面的描述，我们知道了漏洞的原有，那我们审计的时候可以检查处理表单有没有以下判断。

1. 是否有验证token。
2. 是否有图片验证码。
3. 是否有refe信息。

如果三个判断都没有，那么就存在了CSRF漏洞，CSRF不仅限于GET请求，POST请求同样存在。

CSRF 漏洞防范方法

1. 图片验证码，这个想必大家都知道，但是用户体验并不好，我们可以看下面的一些处理方法。
2. token验证。

   token验证方法如下，每次访问表单页的时候，生成一个不可预测的token存放在服务器session中，另外一份放页面中，提交表单的时候需要把这个token带过去，接收表单的时候先验证一下token是否合法。
3. Referer信息验证

   大多数情况下，浏览器访问一个地址，其中header头里面会包含Referer信息,里面存储了请求是从哪里发起的。

如果HTTP头里包含有Referer的时候，我们可以区分请求是同域下还是跨站发起的，所以我们也可以通过判断有问题的请求是否是同域下发起的来防御CSRF攻击。

Referer验证的时候有几点需要注意，如果判断Referer是否包含*.XXX.com,如果有子域名有漏洞，会存在绕过的可能。

如果判断的条件的是Referer中是否包含字符'xxx.com' 那攻击者在他目录中建立一个xxx.com文件夹同样存在绕过的可能。如果可以最合适的判断是，直接判断是否等于当前域名。

三、常规漏洞的防范方法

taint PHP 安全扩展

功能介绍

Taint 可以用来检测隐藏的XSS code, SQL注入， Shell注入等漏洞，并且这些漏洞如果要用静态分析工具去排查， 将会非常困难， 我们来看下面这张图:


```
<?php
 $a = $_GET['a'];

 $file_name = '/tmp' .  $a;
 $output    = "Welcome, {$a} !!!";
 $var       = "output";
 $sql       = "Select *  from " . $a;
 $sql      .= "ooxx";

 echo $output;
//Warning: main(): Attempt to echo a string which might be tainted in xxx.php on line x

 print $$var;
//Warning: main(): Attempt to print a string which might be tainted  in xxx.php on line x

 include($file_name);
//Warning: include() [function.include]: File path contains data that might be tainted in xxx.php on x

 mysql_query($sql);
//Warning: mysql_query() [function.mysql-query]: First argument contains data that might be tainted in xxx.php on line x
?>
```

安装方法

- 下载taint： http://pecl.php.net/package/taint
- 配置

```
/usr/local/php/bin/phpize
./configure --with-php-config=/usr/local/php/bin/php-config
make && make install
```

- 更加详细的可以参考 http://www.cnblogs.com/linzhenjie/p/5485474.html

应用场景

开发团队要求每个人都做到非常的安全比较难，但是把taint安装在开发环境，特别适合，一看到warning信息一般都回去改。

ngx_lua_waf

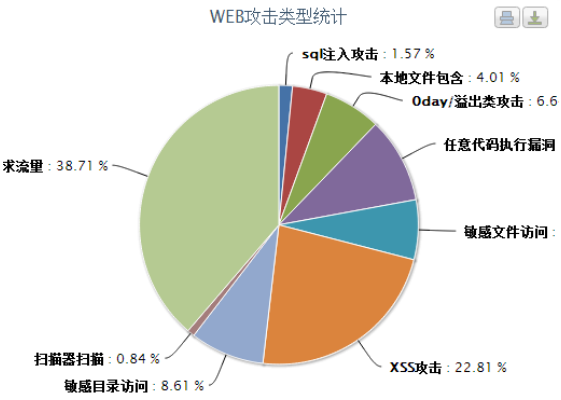
功能介绍

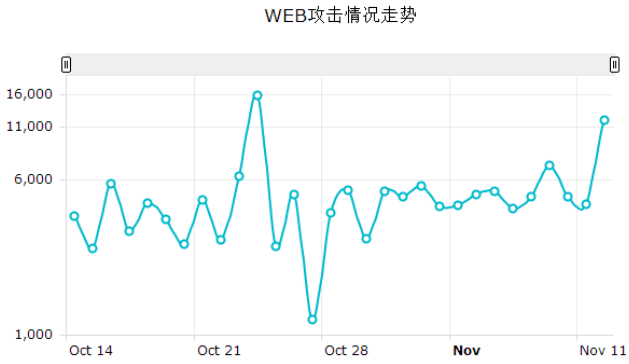
1. 防止sql注入，本地包含，部分溢出，fuzzing测试，xss,SSRF等web攻击。
2. 防止svn/备份之类文件泄漏。
3. 防止ApacheBench之类压力测试工具的攻击。
4. 屏蔽常见的扫描黑客工具，扫描器。
5. 屏蔽异常的网络请求。
6. 屏蔽图片附件类目录php执行权限。
7. 防止webshell上传。

安装方法

- 安装依赖: `luajit` 、 `ngx_devel_kit` 、 `nginx_lua_module`
- 安装 `nginx` 、 `ngx_lua_waf`
- 在 `nginx.conf` 里的http添加配置
- 详细安装文档

效果图





概况统计

已拦截180672次web攻击	攻击者IP地址个

攻击行为统计

SQL注入2837条	文件包含7238条	任意代码执行17869条
命令执行11988条	XSS攻击41216条	目录探测15554条
敏感文件12509条	扫描器扫描1516条	异常请求69945条

总结

这次分享的内容代码审计处理常规漏洞部分挑选了三种类型，还有其他的一些一次也讲不完。代码身材除了像刚才的的参数检查之外，还有逻辑性漏洞审查，下次如果有时间会再做一次逻辑漏洞审计分享。