mysql能干嘛

- 普通用户权限下的`增删改查`，即常规sql注入，俗称`脱裤`
- 普通用户file权限下的`文件读写`，可尝试读取各类敏感配置，如各类账号或者尝试直接往站点目录中写`webshell`
- root用户权限下的`读写文件`，`系统命令执行`
- 针对`数据库连接`的`DDOS`，单用户大批量数据库连接可能会导致mysql无法再提供服务

演示环境

- CentOS6.8 x86_64 最小化,只带基础库安装 eth0: 192.168.3.42 eth1: 192.168.4.14 eth2: 192.168.5.14
- mysql-5.6.27-linux-glibc2.5-x86_64.tar.gz 此次mysql就不再手工编译了,时间比较长,直接用mysql官方提供好的二进制包来做演示

下载,解压 mysql-5.6.27-linux-glibc2.5-x86_64.tar.gz

```
tar xf mysql-5.6.27-linux-glibc2.5-x86_64.tar.gz
mv mysql-5.6.27-linux-glibc2.5-x86_64 /usr/local/
ln -s /usr/local/mysql-5.6.27-linux-glibc2.5-x86_64/ /usr/local/mysql
ls -l /usr/local/mysql/
```

初始化MySQL

- 务必以一个伪用户身份来运行mysql服务,防止别人利用mysql进行提权,后面还会再细说,另外,web服务和数据库服务严禁用同一个系统用户,这样做主要是为了防止入侵者直接通过sql语句往网站目录中写webshell

```
useradd -s /sbin/nologin -M mysql
chown -R mysql.mysql /usr/local/mysql/ && ll /usr/local/mysql/
暂时先让mysql用户对mysql的安装目录能正常读写
/usr/local/mysql/scripts/mysql_install_db --basedir=/usr/local/mysql/ --
datadir=/usr/local/mysql/data/ --user=mysql
chown -R root.root /usr/local/mysql/ && ll /usr/local/mysql/
chown -R mysql.mysql /usr/local/mysql/data/ && ll /usr/local/mysql/data/
cp /usr/local/mysql/support-files/my-default.cnf /etc/my.cnf      创建mysql
配置文件
/usr/local/mysql/bin/mysqld_safe &            启动mysql服务
```

- 把mysql工具包加到root用户的环境变量中,方便后续使用,切记不要放到全局环境变量下,那意思就是说当前系统所有普通用户也都可以使用mysql工具套件

```
echo "export PATH=$PATH:/usr/local/mysql/bin/" >> .bash_profile
source .bash_profile
```

- 让 mysql 随系统自启动的两种方式,更推荐前者,实际中大家可根据个人习惯而定:

```
echo "/usr/local/mysql/bin/mysqld_safe &" >> /etc/rc.local
或者
cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysqld
chmod +x /etc/init.d/mysqld
ll /etc/init.d/mysqld
/etc/init.d/mysqld start
lsof -i :3306
/etc/init.d/mysqld stop
```

- mysql初始化后的基础配置

设置复杂root密码,关于密码安全在之前已无数次强调,此处就不细说了吧,`同时包含大小写,特殊字符,12位以上的随机密码`,越随机越好,这里纯粹只是为了演示

```
/etc/init.d/mysqld start
mysqladmin -uroot password "admin"
```

设置好root密码后,立刻进到mysql下,删除多余数据库,如,test库...,如下

```
# mysql -uroot -p
mysql> show databases;
mysql> drop database test;
```

清除多余数据库用户,只保留`root`的`localhost`和`127.0.0.1`即可,如下

```
mysql> select user,host from mysql.user;
+------+-----------+
| user | host      |
+------+-----------+
| root | 127.0.0.1 |
| root | ::1       |
|      | localhost |
| root | localhost |
|      | oldlnmp   |
| root | oldlnmp   |
+------+-----------+
mysql> drop user ''@'localhost';
mysql> drop user ''@'oldlnmp';
mysql> drop user 'root'@'oldlnmp';
mysql> drop user 'root'@'::1';
mysql> select user,host from mysql.user;
+------+-----------+
| user | host      |
+------+-----------+
| root | 127.0.0.1 |
| root | localhost |
+------+-----------+
```

- 从根源上限制住 mysql 在系统中的各种权限`[ 暂以防止服务器被入侵为最终目的,此处是防不住别人正常的增删改查的,如,'脱裤' ]`

首先,尽可能让`mysql服务`运行在一个较低的系统权限下,防止别人利用该服务提权,如,常见的`udf提权`,这里有些朋友可能会误解,以为只能要能执行系统命令,就是提权,其实不然,在linux中,普通用户也一样可以执行大部分系统命令,但,它依然只是个普通用户,提权的意思就是让你从一个普通用户甚至是一个伪用户身份的权限下直接提升到了root权限,言归正传,因为我们当前运行mysql服务的用户只是一个系统伪用户,也就是说,你当前运行任何sql语句所映射的权限,都是你mysql服务用户的权限,如果这个服务用户权限本身就很低,也一样达不到提权的效果,相对来讲,udf提权更适合用在一些比较古老的系统`<= win2003`和较低的一些mysql版本上`<= mysql 5.1`,新版的mysql除了性能优化之外,安全性也有大幅提升,话说回来,即使安全性提升了,也还是保不住傻逼的配置,之前在win平台下,也许还可以想办法通过dll劫持的方式来进行提权,但5.6.x已经很好的修复了这些问题,对进程的安全上下文也做了更为严格的控制

`useradd -s /sbin/nologin -M mysql`

严格控制住mysql安装目录在`本地文件系统`中的权限,我们再来简单回顾一下上面初始化mysql的详细过程,如下,在初始化之前,首先,我们创建了一个系统伪用户mysql,接着我们把mysql安装目录的属主,属组都改成了mysql,意思就是先让mysql用户对mysql的安装目录暂时能写,因为等会儿要初始化mysql,会生成一些mysql内部系统配置,比如,mysql系统库,所以,必须要让

mysql用户对mysql的安装目录能写才行,紧接着,我们指定了mysql的安装目录和数据存放目录,以及运行mysql服务时的系统用户`[ 即mysql用户 ]`,尝试进行初始化操作,这里务必注意,在第一次初始化完成后,我们后续的权限就不需要那么大了,所以,又把mysql安装目录的属主,属组都改成了root,因为最终还要保证别人能正常的往数据库中写数据,所以,data目录的属主要再改回mysql,说到这份上,想必大家此时都已经非常清晰了吧

```
# chown -R mysql.mysql /usr/local/mysql/ && ll /usr/local/mysql/
# /usr/local/mysql/scripts/mysql_install_db --basedir=/usr/local/mysql/ --datadir=/usr/local/mysql/data/ --user=mysql
# chown -R root.root /usr/local/mysql/ && ll /usr/local/mysql/
# chown -R mysql /usr/local/mysql/data/ && ll /usr/local/mysql/data/
```

- 关于 mysql自身的一些安全配置

在通过上面的一些初步加固后,别人此时再想单单通过mysql拿到服务器权限就比较困难了,毕竟,是从根源上进行控制的,下面我们就再来对针对mysql自身配置做些简要优化

为每个站点,创建独立的数据库以及数据库用户,只允许该用户对该库有最基本的增删改查权限且只能让特定的内网ip才能访问到,有条件,最好站库进行分离,分离的好处在于可以让入侵者无法再正常读写文件,毕竟不在同一台机器上,因为数据库服务器上,根本没有web服务,即使侥幸找到了物理路径,也没啥大用,此外,要严格遵守密码复杂性要求,其实,实际生产环境中,这些权限已经基本能够适应所有日常业务需求,别的权限一律不要加,另外,在授权时,也可通过shell脚本自动对指定库中除管理或系统表之外的其它表进行一一单独授权,而管理表则单独授权给其它数据库用户,这样做的好处就是,此时即使存在sql注入,也让入侵者没法通过跨表来查网站后台管理的账号和密码hash,有些权限对普通用户来讲是完全没必要的,如,file,如果让普通用户都有file权限,也就意味着入侵者可以通过mysql往你服务器本地文件系统中读写文件,虽然,我们是可以对本地文件系统进行详细权限控制,但还是会造成一部分信息泄露,毕竟有些权限,我们是不太好动的,比如,/tmp下,所以,这些危险权限统统的不要,当然,一些非常重要的业务数据表,也可以单独授权给另一个用户进行相互隔离,如果业务逻辑比较复杂,这样做确实麻烦,可以尝试慢慢把业务整理拆分出来,虽然,我们可以利用mysql轻松把权限控制到表级别,但实际中还是非常建议'一站一库',这样后续维护管理起来也非常方便规整

```
mysql> create database sec_list;
mysql> create user 'klion'@'192.168.3.70' identified by 'admin';
mysql> create user 'sec'@'192.168.3.70' identified by 'admin';
mysql> grant insert,delete,update,select,create,drop on sec_list.* to
klion@'192.168.3.70' identified by 'admin';
mysql> grant insert,delete,update,select,create,drop on sec_list.admin to
sec@'192.168.3.70' identified by 'admin';
mysql> flush  privileges;
mysql> show grants for 'klion'@'192.168.3.70';       查询指定数据库用户的系统
权限
mysql> revoke select on sec_list.* from 'sec'@'192.168.3.70'; 撤销指定用户
的指定权限
```

严禁允许root用户外连,正常来讲,不仅仅是root不允许外连,有条件的情况下,mysql服务端口都不要对外开放,只允许特定的内网ip段来连接,另外,所有的实际业务严禁直接用mysql的root用户身份来处理,强烈建议,不同的业务需求,直接创建对应的数据库普通账户来处理即可

```
mysql> grant all  on *.* to  'root'@'%' identified by  'admin'  with grant option;flush
privileges; 严禁用语句对root重新授权
```

把root用户改个比较另类的名字,越看不出来是干啥的越好,嘿嘿.....说实话个人觉得没啥用,如果真的存在sql注入,随便用sql语句查下权限就知道了,另外,sqlmap里的–is-dba也不是白给的,不过这样做的好处,倒是可以一定程度上防爆破

```
mysql> use mysql;
mysql> update user set user='root' where user='guest';
mysql> flush privileges;
mysql> show variables like 'log_%';  查看各类日志存放位置和开启情况
```

```
# history -w       先把当前所有的历史记录写到命令历史文件中去
# vi .bash_history     然后编辑该文件,把里面所有的关于mysql的操作全部删除
# history -w && history    最后,再更新文件,看看刚刚删掉的那些记录还在不在
# rm -fr .mysql_history    删掉mysql操作历史
```