

《信息安全综合实践》实验报告

实验名称: snort 实验

姓名: 黎锦灏 学号: 518021910771 邮箱: 1197993966@qq.com 实验时长: 75 分钟

一、实验目的

1. 加深对入侵检测技术的理解;
2. 了解 snort 的系统结构、配置要求和工作原理;
3. 了解和体验 mysql 数据库操作及其安全设置。

二、实验内容 (5 分)

序	内容	项目	截图要求
1)	体验 Snort 三种工作模式	配置文件修改	截图见思考题 1
2)		Snort 三种工作模式	无
3)	规则编写	按要求编写规则并测试	截图见思考题 1
4)	日志写入数据库	将日志写入本地数据库	无
5)		将日志写入远端数据库 远端数据库 IP: 192.168.1.211 远端数据库用户名: idsr	截图见思考题

步骤 0. 实验准备

启动虚拟机, 登录 ubuntu 操作系统 (name/pw): test/test, 开始实验;
查看网络设置, 记录本机的 IP 地址及子网掩码。

根据实验环境对 snort 配置文件 (/etc/snort/snort.conf) 进行修改, 设置网络变量, 示例如下:

```
# ipvar HOME_NET any

var HOME_NET XXX.XXX.XXX.XXX/XX (根据本机的 IP 地址及子网掩码修改, 本次实验子网掩码为 23)

var LOCAL_HOST XXX.XXX.XXX.XXX (根据本机的 IP 地址修改)

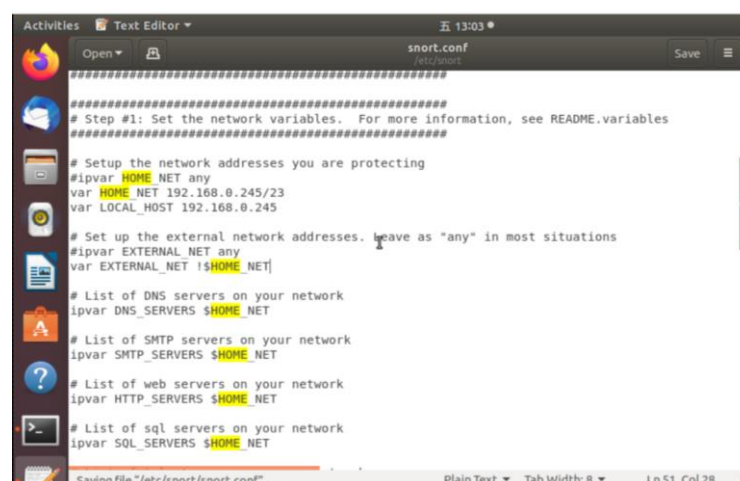
# ipvar EXTERNAL_NET any

var EXTERNAL_NET !$HOME_NET
```

这里 HOME_NET 代表本机所在网段, 即本地网络, LOCAL_HOST 代表本机 IP 地址, EXTERNAL_NET 代表外部网络。步骤 2 在 local.rules 中添加规则时, 可

:

以通过\$HOME_NET 的方式使用这些变量。修改如下图所示：



步骤 1. 体验 Snort 的三种工作模式

分别体验 snort 的三种工作模式：嗅探器、数据包记录器、网络入侵检测系统，查看系统输出/日志，数据包记录器的默认输出日志位置为 /var/log/snort/snort.log.XXX(随机数字)。

嗅探器： `sudo snort -vde`。嗅探器模式将抓到的数据包直接输出在终端中。

数据包记录器： `sudo snort -l /var/log/snort`。数据包记录器将抓到的数据包输出到默认日志中。

网络入侵检测系统(ids)： `sudo snort -c /etc/snort/snort.conf`。IDS 模式为阻塞性进程，使用 CTRL+C 中止，中止时程序会对运行期间抓到的数据包按照配置文件 snort.conf 中引入的规则进行统计，可以通过上滑滚轮进行查看。

步骤 2. 规则编写

按照如下要求编写规则，写入/etc/rules/local.rules 文件中。要求：

- 1) 记录本地网络对本机的 ftp 访问。
- 2) 记录本地网络中的 DNS 查询数据包。
- 3) 对所有主机对本机的 ping 包发出警告 “Got ping!!!”。

进行网络入侵检测系统(`snort -c /etc/snort/snort.conf`)模式实验，利用 wireshark 以及 snort 日志文件 (alert、snort.log)，检查规则是否生效。（默认日志文件夹为/var/log/snort）

步骤 3. 日志输出到数据库

步骤 3.1 日志输出到本地数据库

1) 修改 snort 配置文件 (/etc/snort/snort.conf), 在 Step #6: Configure output plugins 加入以下语句 output unified2:filename snort.u2,limit 128。

2) 查看 mysql 数据库相关情况, 如用户信息、相关权限、数据库情况等。(数据库管理员 root 口令为 123456, ids 本地写入用户 ids1 口令为 ids1)

3) 使用 ids1 用户登录 mysql, 或登录 web 端的 phpmyadmin, 查看日志数据库 ids_local 的情况。

4) 再次进行网络入侵检测系统 (snort -c /etc/snort/snort.conf) 模式 ping 实验, 生成日志信息。

5) 修改 barnyard 配置文件 (/usr/local/etc/barnyard2.conf), 设置数据库目标地址。修改配置文件中默认 output database 为如下样式。

```
output database: log, mysql, user=ids1 password=ids1 dbname=ids_local
host=localhost
```

6) 通过以下命令将日志写入数据库

```
$ sudo barnyard2 -c /usr/local/etc/barnyard2.conf -d /var/log/snort -f snort.u2
-w /var/log/snort/barnyard2.waldo
```

出现 Waiting for new spool file 即为成功写入, CTRL+C 停止进程。

7) 进入 ids_local 数据库查看日志数据, 如 tcp_hdr、udp_hdr、event 等表, 如果有数据记录则说明写入成功。

步骤 3.2 日志输出到异地数据库

1) 异地数据库操作

a) 查看 mysql 数据库相关情况, 如用户信息、相关权限、数据库情况等 (数据库用户 root 口令为 123456)

b) mysql 新增远程写入用户 idsr/idsr, 使其可以本地及远程登录 (' idsr' @' %'), 并且授予该用户数据库写入权限。

c) 编辑 mysql 配置文件 (/etc/mysql/mysql.conf.d/mysqld.cnf), 将 bind-address=127.0.0.1 注释掉, 开启 mysql 数据库远程访问。

2) 本地 IDS 操作

a) 修改本地的 barnyard 配置文件 (/usr/local/etc/barnyard2.conf), 设置数据库目标地址、数据库用户名及口令。

b) 通过 barnyard2 相关命令(与步骤 3.1 同), 将 snort 日志写入远端 mysql 数据库, 并查看日志数据库 ids_remote 内容。

三、分析和思考 (85 分)

1. 列出实验 2 规则编写中所用规则, 并就其生效情况给出截图说明 (不超过 8 张图片)。(30 分)

```
#rule 1

log tcp $HOME_NET any <> $LOCAL_HOST 21 (sid:10000000;rev:1)

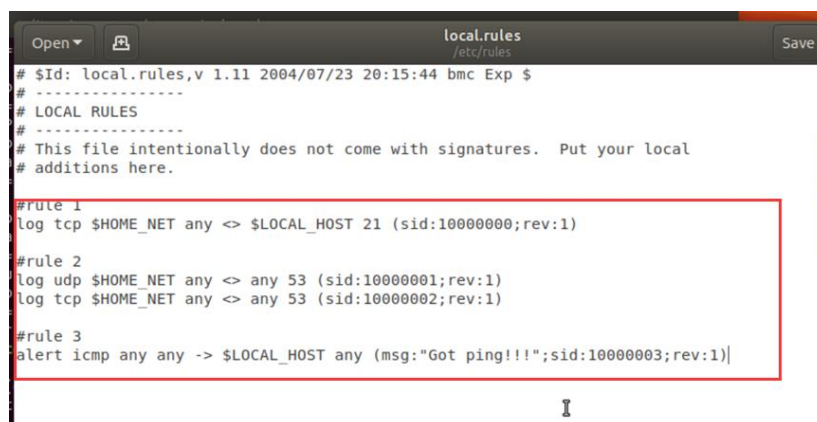
#rule 2

log udp $HOME_NET any <> any 53 (sid:10000001;rev:1)
log tcp $HOME_NET any <> any 53 (sid:10000002;rev:1)

#rule 3

alert icmp any any ->
$LOCAL_HOST any (msg:"Got ping!!!";sid:10000003;rev:1)
```

按照要求编写以上三条规则, 写入/etc/rules/local.rules 文件, 如下图所示:



```
local.rules
/etc/rules
Save

# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# -----
# LOCAL RULES
# -----
# This file intentionally does not come with signatures. Put your local
# additions here.

#rule 1
log tcp $HOME_NET any <> $LOCAL_HOST 21 (sid:10000000;rev:1)

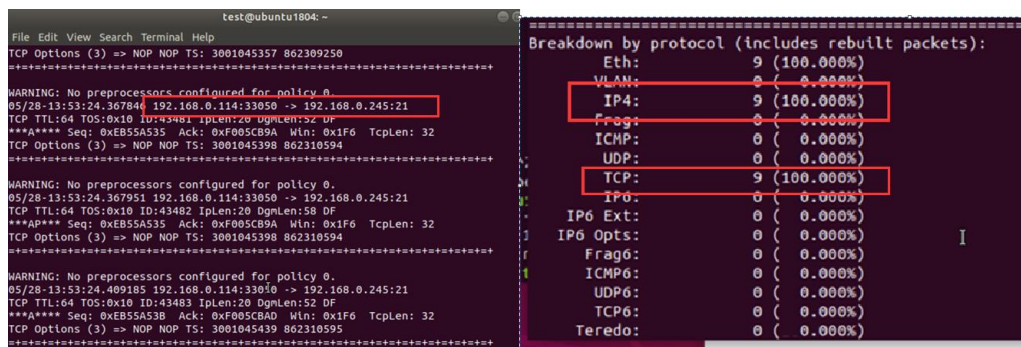
#rule 2
log udp $HOME_NET any <> any 53 (sid:10000001;rev:1)
log tcp $HOME_NET any <> any 53 (sid:10000002;rev:1)

#rule 3
alert icmp any any -> $LOCAL_HOST any (msg:"Got ping!!!";sid:10000003;rev:1)
```

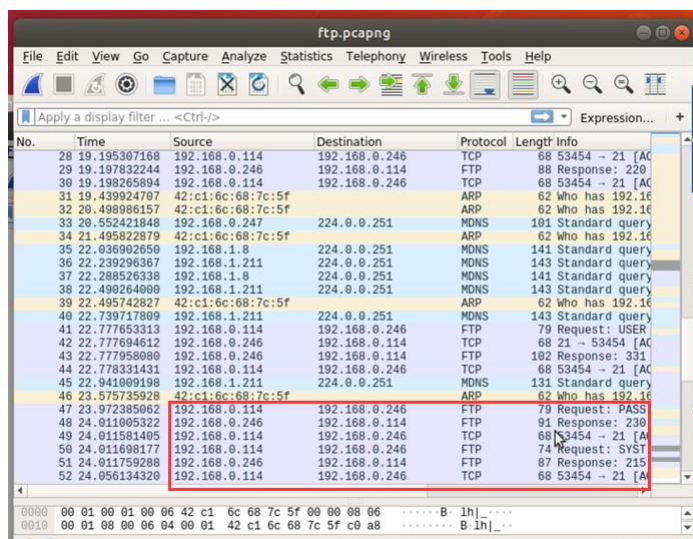
进行网络入侵检测系统(snort -c /etc/snort/snort.conf)模式实验, 分别检查上述设置的三条规则是否生效。

规则一：记录本地网络对本机 FTP 的访问

本机 IP 地址为 192.168.0.245，合作者 192.168.0.114 访问了我的 FTP 后，查看 Snort 日志可看到有 **192.168.0.114** 的 **33050** 端口连接本机 **21** 端口的记录。同时，由于 FTP 基于传输层 TCP 协议，所以捕捉到 **TCP 数据包**，如下图所示：

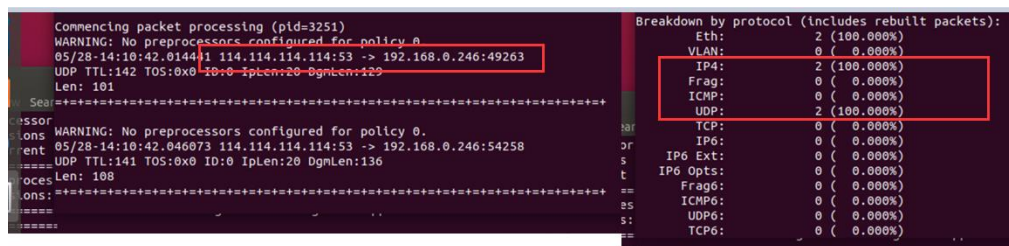


使用 wireshark 抓包分析可以看到有 FTP 数据包，如下图所示：

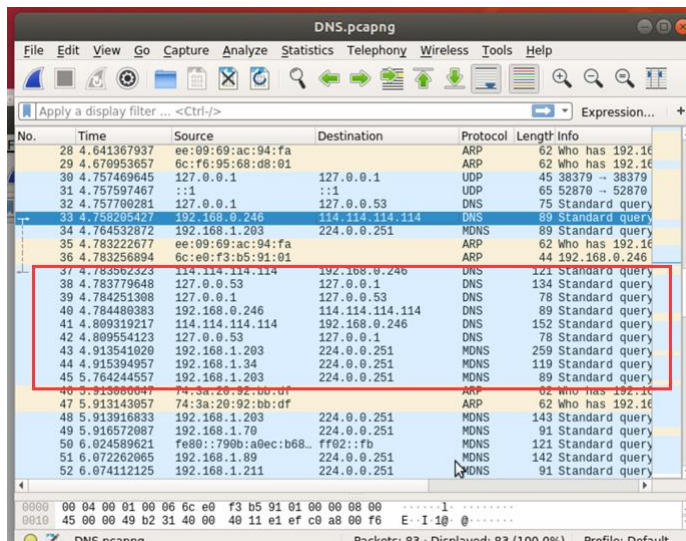


规则二：记录本地网络中的 DNS 查询数据包

在本机的终端输入 nslookup www.baidu.com，执行 DNS 查询服务，然后查看 Snort 日志文件，发现记录了 DNS 数据包，解析到 www.baidu.com 的 IP 为 114.114.114.114。由于 DNS 是基于传输层协议 UDP 的，故捕捉到了 **UDP 数据包**，如下图所示：

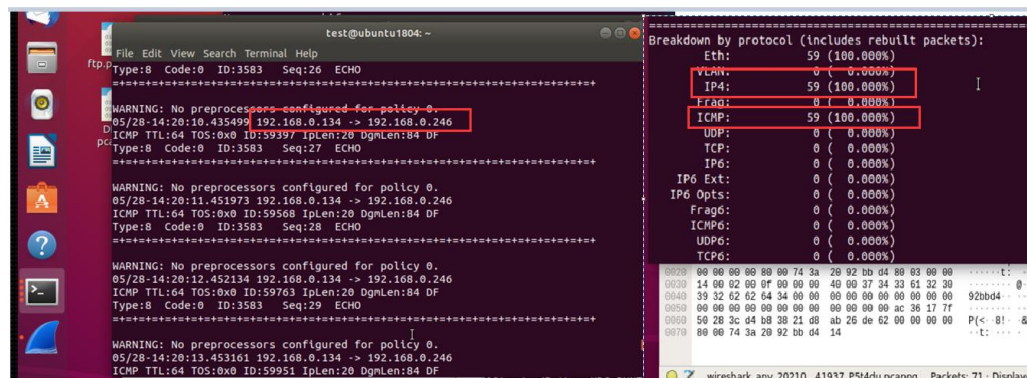


使用 wireshark 抓包分析，可见 DNS 信息，如下图所示：



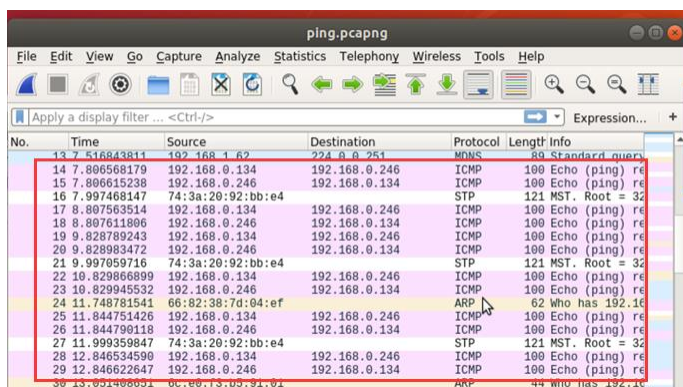
规则三：对所有主机对本机的 ping 包发出警告 “Got ping!!!”

当同伴（192.168.0.134）对我的主机执行 ping 命令后，查看 Snort 日志可以看到记录了 ping 记录。Ping 命令是基于 ICMP 协议的，故 Snort 捕捉到了 ICMP 数据包，如下图所示：

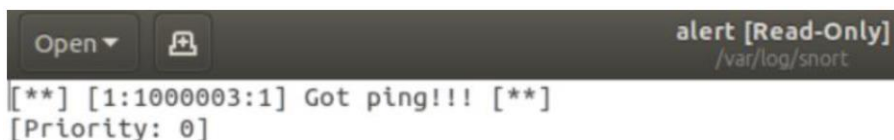


使用 wireshark 抓包分析，可以看到 ping 命令发送的 ICMP 数据包，

如下图所示：



打开 alert 日志，可以看到记录了“Got ping!!!”的提示信息：



2. 日志写入从远端写入数据库时，需要分别在远端和数据库端进行怎样的操作，按步骤说明清楚，并给出相应截图（不超过 6 张图片）（10 分）

一、远端数据库操作：（IP：192.168.1.211）

1、首先在同伴（即远端数据库）的 mysql 中**新增用户 idsr**，方便后面远程写入。口令也设为 idsr，定义 idsr 可以本地登录、远程登录（‘idsr’@’%’）。

```
mysql> create user 'idsr'@'%' IDENTIFIED BY 'idsr'
-> ;
Query OK, 0 rows affected (0.02 sec)

mysql>
```

2、授予 idsr 用户权限来操作数据库：

```
grant all privileges on ids_remote to 'idsr'@'%';
```

3、在授予权限后需要刷新用户权限，来保证授予生效：

```
mysql > flush privileges;
```

4、为了开启 mysql 数据库远程访问，还需编辑 mysql 配置文件 (/etc/mysql/mysql.conf.d/mysqld.cnf)，将 bind-address=127.0.0.1 注释掉：

```
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
#bind-address          = 127.0.0.1
```

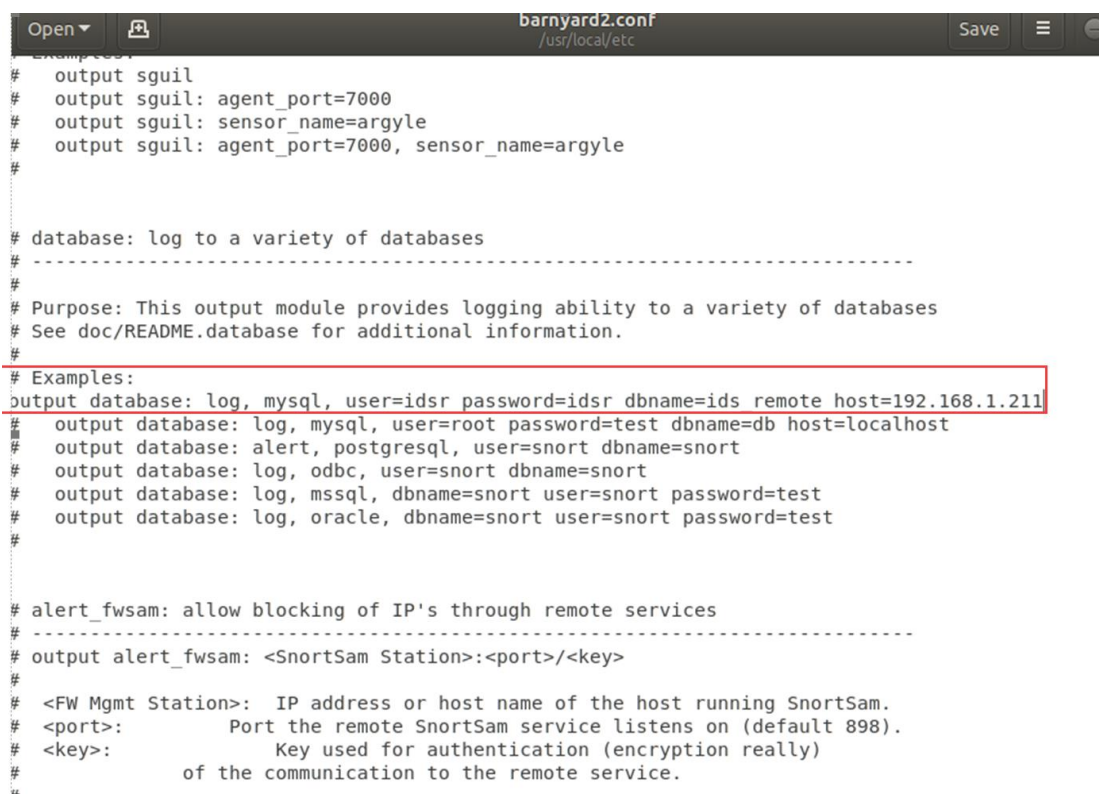
5、最后需要重启 mysql 服务来让修改生效，如下图所示：

service mysql restart

```
test@ubuntu1804:/$
test@ubuntu1804:/$ service mysql restart
test@ubuntu1804:/$
```

二、本地数据库端操作：

1、首先修改本地数据库的 barnyard 配置文件 (/usr/local/etc/barnyard2.conf)，设置数据库目标地址为 192.168.1.211、远端数据库用户名及口令都为 idsr。



```
Open  barnyard2.conf  Save
/usr/local/etc

# output sgul
# output sgul: agent_port=7000
# output sgul: sensor_name=argyle
# output sgul: agent_port=7000, sensor_name=argyle
#

# database: log to a variety of databases
# -----
#
# Purpose: This output module provides logging ability to a variety of databases
# See doc/README.database for additional information.
#
# Examples:
output database: log, mysql, user=idsr password=idsr dbname=ids remote host=192.168.1.211
# output database: log, mysql, user=root password=test dbname=db host=localhost
# output database: alert, postgresql, user=snort dbname=snort
# output database: log, odbc, user=snort dbname=snort
# output database: log, mssql, dbname=snort user=snort password=test
# output database: log, oracle, dbname=snort user=snort password=test
#

# alert_fwsm: allow blocking of IP's through remote services
# -----
# output alert_fwsm: <SnortSam Station>:<port>/<key>
#
# <FW Mgmt Station>: IP address or host name of the host running SnortSam.
# <port>:          Port the remote SnortSam service listens on (default 898).
# <key>:          Key used for authentication (encryption really)
#                  of the communication to the remote service.
```

2、输入 barnyard2 相关命令，并再次 ping 本地主机，得到 Snort 日志后再将其写入同伴的远端数据库，并查看数据库 ids_remote 内容。

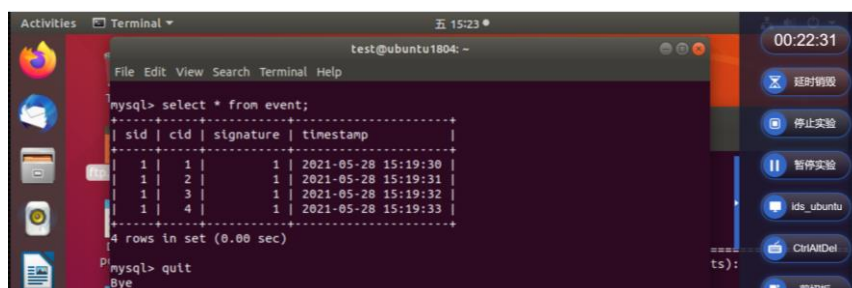
```
sudo barnyard2 -c /usr/local/etc/barnyard2.conf -d /var/log/snort -f snort.u2
-w /var/log/snort/barnyard2.waldo
```



```
test@ubuntu1804:~$ sudo barnyard2 -c /usr/local/etc/barnyard2.conf -d  
1-f snort.u2 -w /var/log/snort/barnyard2.waldo  
1Running in Continuous mode  
1  
1  
1==== Initializing Barnyard2 ====  
1Initializing Input Plugins!  
19Initializing Output Plugins!  
1acParsing config file "/usr/local/etc/barnyard2.conf"  
4  
@u  
1+[ Signature Suppress list ]+  
yt-----  
yt+[No entry in Signature Suppress List]+  
yt-----  
yt+[ Signature Suppress list ]+
```

三、远端写入结果

远端写入操作执行后，查看同伴数据库 ids_remote 中的 event 表，如下图所示：



```
mysql> select * from event;  
+-----+-----+-----+-----+  
| sid | cid | signature | timestamp |  
+-----+-----+-----+-----+  
| 1 | 1 | 1 | 2021-05-28 15:19:30 |  
| 1 | 2 | 1 | 2021-05-28 15:19:31 |  
| 1 | 3 | 1 | 2021-05-28 15:19:32 |  
| 1 | 4 | 1 | 2021-05-28 15:19:33 |  
+-----+-----+-----+-----+  
4 rows in set (0.00 sec)  
  
mysql> quit  
Bye
```

由此可见成功实现了远端写入。

3. 查看日志数据库的结构，给出简要说明（不超过 4 张图片），并分析日志写入数据库这一方式可能存在哪些安全隐患。（20 分）

在 mysql 中查看已有的数据库：

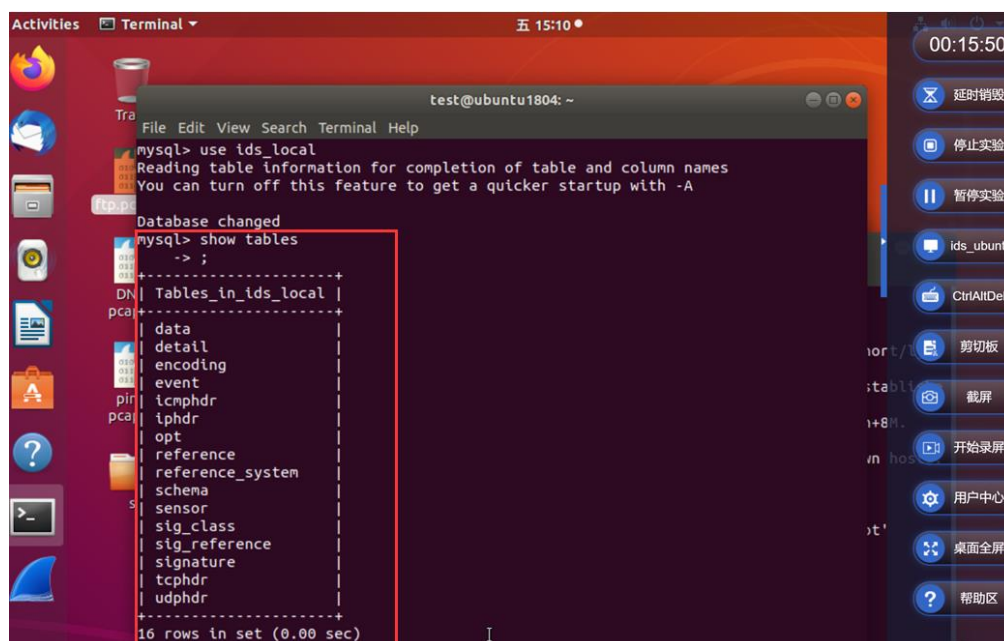
show databases;

当前已有 ids_local、ids_remote、information_schema 等数据库，如下图所示：

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| ids_local |  
| ids_remote |  
| mysql |  
| performance_schema |  
| suricata_remote |  
| suricata_local |  
| sys |  
+-----+
```

下面以 `ids_local` 为例具体分析日志数据库的结构：

输入 `show tables;` 可得日志数据库的表信息，如下图所示：



```
test@ubuntu1804: ~  
mysql> use ids_local  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
Database changed  
mysql> show tables  
+-----+  
Tables_in_ids_local |  
+-----+  
data  
detail  
encoding  
event  
icmphdr  
iphdr  
opt  
reference  
reference_system  
schema  
sensor  
sig_class  
sig_reference  
signature  
tcphdr  
udphdr  
+-----+  
16 rows in set (0.00 sec)
```

简要说明 `ids_local` 中各表的作用：

- `data`: 记录数据包的载荷内容
- `detail`: 存储 Snort 记录的粒度
- `event`: 存储检测到警告的元数据
- `icmphdr`: ICMP 协议字段
- `iphdr`: IP 协议字段
- `opt`: 存储 IP、TCP 协议选项
- `sensor`: Snort 传感器信息
- `signature`: 有形式的告警信息
- `tcphdr`: TCP 协议字段
- `udphdr`: UDP 协议字段

日志写入数据库带来的安全隐患：

1、由于 Snort 是单线程应用程序，在触发警报或日志事件时，Snort 首先需要将警报或日志条目发送到远程数据库，然后才返回传入的数据流。此时单线程可能对 Snort 正常工作造成影响：若 Snort 等待数据库插入过程完成相对较长的

时间，则 Snort 警报的及时性遭到破坏，当有潜在威胁来临时用户不能立刻接收到警报。

2、具有写入权限则可以将数据通过 `SELECT...INTO OUTFILE...` 的方式写到服务器上有写权限的目录下，作为文本格式存放，然后通过 `LOAD DATA INFILE...` 将文本文件数据导入到数据表中。不难看出，可能造成数据泄露。

3、为了完成写入远端数据库操作，我们需要先修改配置文件，而目的数据库 IP、账号和口令在配置文件中都是明文存储的，若此文件的权限管理不当或被攻击者获取，则可能导致远程数据库的账号、口令泄露，造成数据安全风险。

4、本地的日志文件是驻留在本地数据库端的，写入远端数据库会有传输过程，这一过程可能造成数据泄露，攻击者会截获数据并由此获取数据库的结构信息。由此可见写入操作带来很大的安全隐患。

5、假设可从浏览器通过链接 `http://ServerIP/test.php?id=1` 访问本地一网站，其中仅有 `id` 参数，且参数类型为数字。若该链接可被实施 SQL 注入攻击，请通过正则表达式编写 snort 规则，对此类 SQL 注入攻击（数字型和字符型）进行检测，并发出警告“SQL Injection Alert”（假设后台数据库为 MySQL）。（25 分）

一、数字型注入

当输入的参数为整形时且存在注入漏洞，则认为是数字型注入。

检测方式：

1、输入 URL：

`http://ServerIP/test.php?id=1 and 1=1`

即执行 SQL 语句: `select * from table where id=1' and 1=1`

此时语句执行正常, 与原命令结果完全一致;

2、输入 URL:

`http://ServerIP/test.php?id=1 and 1=2`

即执行 SQL 语句: `select * from table where id=1 and 1=2`

此时语句正常执行, 但由于条件为 false 无法查询, 返回数据与原 URL 不同。

3、输入 URL:

<http://ServerIP/test.php?id=1'>

即执行 SQL 语句: `select * from table where id=1'`

此时 SQL 语句报错, 程序无法正常从数据库中查询出数据。

若存在上述情况, 则可以判断存在数字型注入。不难看出, 对于数字型 SQL 注入的关键点在于对非数字字符的检测。我们需要保证输入参数全是数字, 通过正则表达式编写 Snort 规则如下:

```
# rule1 数字型 sql
alert tcp !192.168.125.140/24 any ->
$HTTP_SERVERS $HTTP_PORTS(msg: "SQL Injection Alert";
uricontent: "/test.php?id=; pcre:" ^((?!([1-9][0-9]*[0-9])).)*$");
```

二、字符型注入

当输入的参数为字符串时且存在注入漏洞, 称为字符型注入。字符型和数字型最大的一个区别在于, 数字型不需要单引号来闭合, 而字符串一般需要通过单引号来闭合的。

下面对比数字型与字符型输入:

数字型: `select * from table where id =1`

字符型: `select * from table where id ='1'`

因此, 在构造 payload 时只需构造闭合单引号, 即可成功执行语句。

检测方式:

1、执行 SQL 语句:

```
select * from table where name='admin''
```

由于存在三个单引号, 无法执行且程序报错。

2、执行 SQL 语句:

```
select * from table where name='admin' and 1=1'
```

此时与 1 类似, 无法进行注入, 还需要通过注释符号将其绕过;

3、执行 SQL 语句:

```
select * from table where name='admin' and 1=2 --'
```

将会报错, 需要注释掉后面的单引号。

如果满足以上三点, 可以判断为字符型注入。字符型注入的显著特点是参数中会有单引号 "'", 因此我们可以针对单引号来构建 Snort 规则。注意到单引号对应的十六进制是%27, 我们通过正则表达式构造 Snort 规则如下:

```
# rule2 字符型 sql
alert tcp !192.168.125.140/24 any ->
$HTTP_SERVERS $HTTP_PORTS(msg: "SQL Injection Alert";
uricontent: "/test.php?id="; pcre:" ^((?!([1-9][0-9]*).)*$");)
```

一旦输入参数含有单引号, 都会发出警告, 可以有效防止字符型注入。

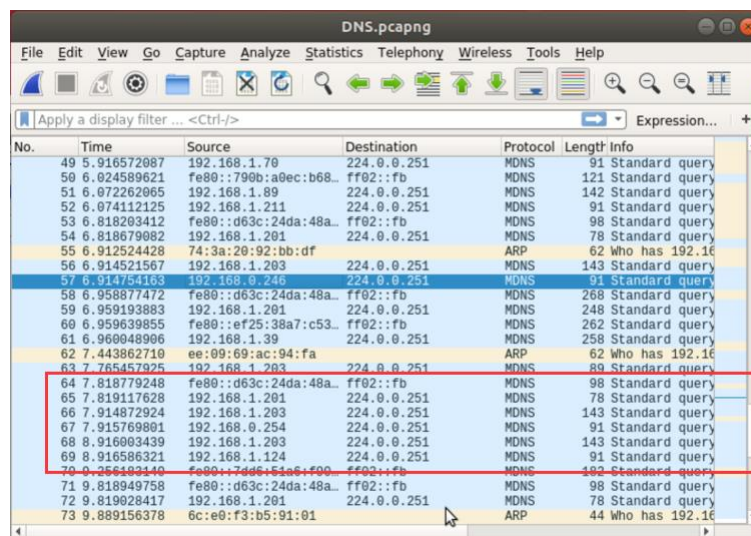
四、实验总结（收获和心得）（5 分）

本次实验让我首次接触入侵检测系统，对监听、匹配再到响应的处理流程有了深入了解，对 IDS 的框架更加熟悉，加深了对入侵检测技术的理解，对系统网络防御有了基本认识。通过预习和实验操作，我了解了 Snort 的系统结构、配置要求和工作原理。实验三的内容是 MySQL 数据库和安全设置，实现了 MySQL 的远程写入 Snort 日志文件，让我对 MySQL 的操作更加熟悉。

五、尚存问题或疑问、建议（5 分）

问题

- 1、在 wireshark 抓包分析中，我注意到有多个本地网段下的主机（应该是同时做实验的其他同学）与 224.0.0.251 的交互数据包，想请问这些数据包是执行什么操作的呢？



No.	Time	Source	Destination	Protocol	Length	Info
49	5.916572087	192.168.1.70	224.0.0.251	MDNS	91	Standard query
50	6.024589621	fe80::790b:a9ec:b68...	ff02::fb	MDNS	121	Standard query
51	6.072262065	192.168.1.89	224.0.0.251	MDNS	142	Standard query
52	6.074112125	192.168.1.211	224.0.0.251	MDNS	91	Standard query
53	6.818203412	fe80::d63c:24da:48a...	ff02::fb	MDNS	98	Standard query
54	6.818679082	192.168.1.201	224.0.0.251	MDNS	78	Standard query
55	6.912524428	74:3a:20:92:bb:df	224.0.0.251	ARP	62	Who has 192.16...
56	6.914521567	192.168.1.203	224.0.0.251	MDNS	143	Standard query
57	6.914754163	192.168.0.246	224.0.0.251	MDNS	91	Standard query
58	6.958877472	fe80::d63c:24da:48a...	ff02::fb	MDNS	268	Standard query
59	6.959193883	192.168.1.201	224.0.0.251	MDNS	248	Standard query
60	6.959639855	fe80::ef25:38a7:c53...	ff02::fb	MDNS	262	Standard query
61	6.960048906	192.168.1.39	224.0.0.251	MDNS	258	Standard query
62	7.443862710	ee:09:69:ac:94:fa	224.0.0.251	ARP	62	Who has 192.16...
63	7.765457925	192.168.1.203	224.0.0.251	MDNS	89	Standard query
64	7.818779248	fe80::d63c:24da:48a...	ff02::fb	MDNS	98	Standard query
65	7.819117628	192.168.1.201	224.0.0.251	MDNS	78	Standard query
66	7.914872924	192.168.1.203	224.0.0.251	MDNS	143	Standard query
67	7.915769801	192.168.0.254	224.0.0.251	MDNS	91	Standard query
68	8.916003439	192.168.1.203	224.0.0.251	MDNS	143	Standard query
69	8.916586321	192.168.1.124	224.0.0.251	MDNS	91	Standard query
70	8.916586321	fe80::d63c:24da:48a...	ff02::fb	MDNS	102	Standard query
71	9.818949758	fe80::d63c:24da:48a...	ff02::fb	MDNS	98	Standard query
72	9.819028417	192.168.1.201	224.0.0.251	MDNS	78	Standard query
73	9.889156378	6c:e0:f3:b5:91:01	224.0.0.251	ARP	44	Who has 192.16...

- 2、除了 Snort 之外，现在业界主要采用的入侵检测系统是什么呢？

建议

在实验过程中，常有修改配置文件、编辑权限后未生效的情况，后经助教提醒才知道是没有执行重启 SQL 服务、刷新 MySQL 数据库权限等更新操作，希望在实验手册中可以对容易碰到的问题进行提示（不用太具体，只需在某步骤以后稍加提示）。

