ipacctd - подсчёт траффика через IPFW

Автор: lissyara.
Оригинал: http://www.lissyara.su/articles/freebsd/traffic_count/ipacctd/

Купили новый сервант, на AMD64 - (AMD - форева!) - под архивацию пользовательских документов, и прочего подобного хламу. Накатил на него, разумеется, фряху - 6-ю версию, платформа - x64. В-общем-то всё хорошо, но, как оказалось некоторые злостные программеры пишут строго под определённую архитектуру - x86 и всё тут... Первым таким приложением стал trafd - он компилился, запускался, но вот вместо статистики в файлы сохранялся какой-то бред... Пересобрал - бесполезно... Тогда собрал пакет на другой машине, тоже 6-я FreeBSD, но x86. Раскатал на AMD64 - не запускается... Что и ожидал (однако где-то в глубине души копошилась надежда - вдруг поднимется :))). Полез в порты - искать чем считать. Нашёл - bpft, сайт проекта мёртв, но вещщь родственная trafd. К сожалению родство оказалось слишком близким - у них одна наследственная болезнь - бред в логах. Снёс, полез дальше. Нашёл - ipacctd - сайт не указан.... Фомат логов очень похож на тот, что у trafd - потому переделки будут мелкими.
Ставим из портов:/usr/home/lissyara/>cd /usr/ports
/usr/ports/>make search name=ipacctd
Port: ipacctd-1.46_1
Path: /usr/ports/net-mgmt/ipacctd
Info: IP accounting using divert socket
Maint: skv@FreeBSD.org
B-deps:
R-deps:
WWW:

/usr/ports/>cd /usr/ports/net-mgmt/ipacctd
/usr/ports/net-mgmt/ipacctd/>make && make install && make clean

Программа абсолютно копеечного размера (15кб), но самый главный плюс оказался не в этом - если у Вы удосужились прикрутить русский язык к консоли, то сможете лицезреть ман по этой программе на русском :) Собственно по этой причине не буду расписывать ключи, и прочее.
Работает через IPFW - пришлось пересобрать ядро и поднять файрволл. Извращаться особо не стал, забил совсем немного правил - машина с одним интерфейсом, смотрящим в локалку:
/etc/rc.firewall#!/bin/sh -xv

# переменные
FwCMD="/sbin/ipfw"

```
${FwCMD} -f flush

${FwCMD} add divert 10001 ip from any to any via sk0
${FwCMD} add divert 10002 ip from any to any via lo0
#${FwCMD} add deny ip from not 192.168.0.0/16 to me
${FwCMD} add allow ip from any to any
```

Учтите, что правила надо добавлять в самом верху файрволла до остальных (сразу после -f flush). Также добавляем следующие строки в /etc/rc.conf (для двух интерфейсов - lo0 и sk0)

```
ipacctd_enable="YES"
ipacctd_flags="-v"
ipacctd_rules="sk0 lo0"
ipacctd_rule_sk0_flags="-p 10001 -f /var/log/traffic_sk0.log"
ipacctd_rule_sk0_pid="/var/run/ipacctd.sk0"
ipacctd_rule_lo0_flags="-p 10002 -f /var/log/traffic_lo0.log"
ipacctd_rule_lo0_pid="/var/run/ipacctd.lo0"
```

После чего запускаем ipacctd:/usr/local/etc/rc.d/>./ipacctd.sh start

и перезагружаем правила файрволла:/usr/local/etc/rc.d/>sh /etc/rc.firewall > /dev/null &

```
[1] 16480
/usr/local/etc/rc.d/>
[1]    Done                    sh /etc/rc.firewall > /dev/null
/usr/local/etc/rc.d/>
```

Всё. Траффик считается. Осталось привенуть скрипт, который, будет всё это складывать в БД. Я чуть-чуть модифицировал свой скрипт, написанный для trafd:

```
#!/bin/sh -xv
#
#
##########   Вводим данные для подключения к MySQL серверу   ##################
# IP адрес MySQL сервера
IP_MySQL_servera="localhost"
# Имя пользователя для доступа к БД в которой храниться траффик
username="ipacctd"
# Пароль пользователя MySQL
user_passw="ipacctd"
# Имя базы данных
db_name="ipacctd"

##########
```

```
# Сегодяшний день
day="`date +%Y-%m-%d`"
# Текущий год
year="`date +%Y`"
# Текущий месяц
month="`date +%m`"
# Текущее время (секунды специально сделаны 00 - иногда cron запускает скрипт не
# в 00 секунд а позже (максимум что я видел - в 13), если машина очень загружена -
# как итог в логах начинает фигурировать разное число секунд.
# Мне это непонравилось :)
curr_time="`date +%H:%M:00`"
# Директория в которой будут храниться текстовые файлы с логами trafd
NewDir="/var/traffic/${year}/${month}"
# Пытаемся создать эту самую директорию на случай если это первый запуск
# или произошла смена месяца (года)
mkdir -p ${NewDir}
# Ну и топаем туда
cd ${NewDir}

# Местоположение исполняемого файла клиента MySQL
mysql="/usr/local/bin/mysql"
# Префикс для команд (лень же каждый раз набивать параметры подключения)
sql_preffix="${mysql} --host=${IP_MySQL_servera}
--user=${username} --password=${user_passw} --database=${db_name}"

# Считываем все переменные из файла /etc/rc.conf с целью извлечь оттуда
# строчку с названиями интерфейсов по которым работает trafd
# (У меня три сетевых платы и lo0 - просто интереса ради)
. /etc/rc.conf

# Сохраняем статистику по всем интерфейсам
# sleep введён по причине, что иногда не успевает траффик
# в текстовый файл сохраниться - подумываю ещё sync воткнуть
killall -1 ipacctd && sleep 1

# Для всех интерфейсов выковырнутых из rc.conf (висят в ${trafd_ifaces})
# выполняем один и тот же набор действий по разбору логов и запихиванию
# их в базу данных
for iface in ${ipacctd_rules}
do

# Дозаписываем логи в текстовый файл (пусть лежат на всякий случай...)
echo " >> /var/log/traffic_${iface}.log
cat /var/log/traffic_${iface}.log >> ${NewDir}/summary.${iface}
# Далее - загоняем траффик в БД
#
```

```
${sql_preffix} --execute="CREATE TABLE `traffic_tmp`
(`date` DATE NOT NULL, `time` TIME NOT NULL,
`from_IP` CHAR(16) NOT NULL, `port_from_IP` CHAR(8) NOT NULL,
`to_IP` CHAR(16) NOT NULL, `port_to_IP` CHAR(8) NOT NULL,
`protocol` ENUM('icmp','tcp','udp') NOT NULL, `bytes` CHAR(16) NOT NULL,
`paketov` CHAR(16) NOT NULL) TYPE=MyISAM COMMENT='tmp_table'" 2>/dev/null

########### Лопатим данные для интерфейса ${iface}  ####################
# Очищаем временную таблицу
${sql_preffix} --execute="DELETE FROM `traffic_tmp`"
# Построчно превращаем файл со статистикой в набор переменных
cat /var/log/traffic_${iface}.log |
{
while read stroka
do
from_IP=`echo "${stroka}" | awk '{print $1}'`
port_from_IP=`echo "${stroka}" | awk '{print $2}'`
to_IP=`echo "${stroka}" | awk '{print $3}'`
port_to_IP=`echo "${stroka}" | awk '{print $4}'`
protocol=`echo "${stroka}" | awk '{print $5}'`
bytes=`echo "${stroka}" | awk '{print $6}'`
paketov=`echo "${stroka}" | awk '{print $7}'`
# Загоняем полученный набор во временную таблицу
${sql_preffix} --execute="insert into `traffic_tmp` (date, time, from_IP,
port_from_IP, to_IP, port_to_IP, protocol, bytes, paketov)
values ('${day}', '${curr_time}', '${from_IP}',
'${port_from_IP}', '${to_IP}', '${port_to_IP}',
'${protocol}', '${bytes}', '${paketov}')"
done
}
# Стираем пустые строки (а вот откуда они вылазиют я так и непонял....)
${sql_preffix} --execute="DELETE FROM `traffic_tmp` WHERE from_IP='' AND
port_from_IP='' AND to_IP='' AND port_to_IP='' AND protocol=""
# Стираем строки в которых полное число байт (вместе с технической инфой)
# равно нулю (тоже непойми откуда берутся - раз в статистику trafd попали -
# значит соединение было и байты должны были б быть...)
${sql_preffix} --execute="DELETE FROM `traffic_tmp` WHERE paketov='0'"
# Создаём таблицу для окончательного хранения траффика
# (на тот случай если она не создана - хотя конечно тоже дурацкий вариант -
# пытаться создать таблицу при каждом запуске скрипта, но другой вариант -
# проверять существование и если нету её - то создавать. А какая разница? Так
# как сделано сейчас - проще и менее ресурсоёмко)
${sql_preffix} --execute="CREATE TABLE `${iface}_${year}`
(`date` DATE NOT NULL, `time` TIME NOT NULL,
`from_IP` CHAR(16) NOT NULL, `port_from_IP` CHAR(8) NOT NULL,
`to_IP` CHAR(16) NOT NULL, `port_to_IP` CHAR(8) NOT NULL,
```

```
`protocol` ENUM('icmp','tcp','udp') NOT NULL, `bytes` CHAR(16) NOT NULL,
`paketov` CHAR(16) NOT NULL) TYPE=MyISAM COMMENT='База
данных траффика по (${iface}) интерфейсу за ${year} год'" 2>/dev/null
# Перекидываем траффик из временной таблицы в окончательную, при этом
# объединяем строки в которых совпадает ВСЁ кроме числа байт.
${sql_preffix} --execute="INSERT INTO `${iface}_${year}` (date, time, from_IP,
port_from_IP, to_IP, port_to_IP, protocol, bytes, paketov)
SELECT date, time, from_IP, port_from_IP, to_IP, port_to_IP,
protocol, sum(bytes) as bytes, sum(paketov) as paketov FROM
traffic_tmp GROUP BY date, time, from_IP, port_from_IP, to_IP,
port_to_IP, protocol"

# Очищаем файл с логами о том когда и по какому интерфейсу сохранялась статистика
cat /dev/null > /var/log/traffic_${iface}.log


done
```

Доработки минимальны - изменилась команда сохранения траффика, и одна колонка сменилась - вместо `all_bytes` стало `paketov`... После чего пихаем скрипт в планировщик - я всунул его на запуск раз в минуту (все звёздочки, кроме команды)...

P.S. По прошествии нескольких дней обратил внимание, что в моменты пиковой загрузки (когда всех припёрло лезть в инет) скрипт работает долго - 20-30 секунд на нененагруженной машине... Пару раз, когда машина была загруженна, даже не успевал отработать. Причина нашлась быстро - сильно возросло число строк в логах - trafd все порты больше 10000 обзывал client a ipacctd честно их писал... Подумавши, настругал такой скрипт на перл:#!/usr/bin/perl -w

```
# вводим переменные
# MySQL - хост где БД
$db_host = 'localhost';
# MySQL юзер
$db_user = 'ipacctd';
# MySQL пароль
$db_password = 'ipacctd';
# MySQL база данных
$db_database = 'ipacctd';
# подрубаем модуль для работы с MySQL
use Mysql;
# время - тока чтоб год достать...
use Time::localtime;

# достаём время
# Год
```

```
$year = localtime->year() + 1900;
# Месяц (идиотский язык, чтобы достать месяц в виде
# двузначного числа приходиться изгаляться через жопу...)
# Если знаете способ лучше - подскажите, поменяю...
$month = `date '+%m'`;
$month = substr($month,0,2);

# Коннектимся к MySQL
$dbh = Mysql->Connect($db_host,$db_database,$db_user,$db_password);

# Вызываем внешние программы по сохранению траффика
system("killall -1 ipacctd && sleep 2");

#use strict;
if(open(RC_CONF,"/etc/rc.conf")){
my @data = reverse <RC_CONF>;
chomp @data;
close RC_CONF;
foreach my $str (@data)
 {
 # разбираем rc.conf
 next if $str =~ /^#/ or $str =~ /^s*$/;
 $str =~ /^s+/;
 $str =~ /s+$/;
 my($var_name,$var_value) = split(/=/, $str);
 if($var_name eq 'ipacctd_rules')
 {
 $var_value =~ s#^s*(["']?)(.*)1#$2#;
 foreach my $interface (split (/s+/, $var_value))
 {
 # шуршим по интерфейсам
 # Создаём таблицу для постоянного хранения траффика
 # строим кверю к MySQL
$MySQL_query = "CREATE TABLE IF NOT EXISTS `" . $interface . "_" . $year . "`(
 `unic_id` INT(16) NOT NULL auto_increment,
 `date` DATE NOT NULL,
 `time` TIME NOT NULL,
 `unix_time` INT(12) NOT NULL,
 `from_IP` CHAR(16) NOT NULL,
 `port_from_IP` INT(8) NOT NULL,
 `to_IP` CHAR(16) NOT NULL,
 `port_to_IP` INT(8) NOT NULL,
 `protocol` CHAR(12) NOT NULL,
 `bytes` INT(16) NOT NULL,
 `paketov` INT(8) NOT NULL,
 PRIMARY KEY (`unic_id`),
```

```
 KEY `date`(`date`),
 KEY `unix_time`(`unix_time`)
 ) ENGINE=MyISAM COMMENT='Traffic for " . $interface . "-interface'";
 # Делаем запрос к БД, если неудачный - помираем с ошибкой
 $dbh->Query("$MySQL_query") or die $Mysql::db_errstr;
 # строим путь к файлу с траффиком
 $file_path = "/var/log/traffic_" . $interface . ".log";
 # открываем файло
 open TRAFFIC,"$file_path";
 # Разбираем построчно
 while (<TRAFFIC>)
 {
 # убираем лишние пробелы
 #tr/s+/ /s;
 # Разбиваем по пробелам на переменные
 ($from_IP,$port_from_IP,$to_IP,$port_to_IP,$protocol,
 $bytes,$paketov) = split(/s+/,$_);
 # пихаем траффик в БД

 $MySQL_query = "INSERT INTO `" . $interface . "_" . $year . "`
 (`date`,`time`,`unix_time`,`from_IP`,`port_from_IP`,`to_IP`,
 `port_to_IP`,`protocol`,`bytes`,`paketov`) VALUES (DATE(NOW()),
 TIME(NOW()),UNIX_TIMESTAMP(),'" . $from_IP . "',
 '" . $port_from_IP . "','" . $to_IP . "','" . $port_to_IP . "',
 '" . $protocol . "','" . $bytes . "','" . $paketov . "')";
 # Делаем запрос к БД, если неудачный - помираем с ошибкой
 $dbh->Query("$MySQL_query") or die $Mysql::db_errstr;
 }
 # создаём директории
 system("mkdir -p /var/traffic/" . $year . "/" . $month);
 # переносим траффик
 $otkuda = "/var/log/traffic_" . $interface . ".log";
 $kuda = "/var/traffic/" . $year . "/" . $month . "/summary." . $interface;
 system("cat $otkuda >> $kuda");
 # очищаем файло
 system("cat /dev/null > $otkuda");


 # создаём таблицу, где будет храниться траффик

 }
 }
 }
}

# выходим
```

1;

Он прекрасно заменяет тот же шелловый скрипт. Тока работает раз в 10-15 быстрей :)))
Также есть и нововведения (колонка unix_time и unic_id) - понадобились для работы. Если заменять существующий shell скрипт колонку unix_time надо добавить, а если с нуля - то сам всё создаст. Также убрана временная таблица. Тут она не нужна.

P.S. Ненавижу перл.