Математические структуры и моделирование
2006, вып. 16, с. 11–20

УДК 681.3

ОРГАНИЗАЦИЯ БИЛЛИНГОВОЙ СИСТЕМЫ

А.С. Дрозд

Рассматривается создание биллинговой системы и безопасного шлюза для домашних локальных сетей, а также для небольших и средних организаций.

Существует множество способов реализовать биллинговую систему (систему для подсчета трафика). Учет можно вести как на клиенте, так и на маршрутизаторе. В свою очередь, маршрутизатор может быть как аппаратный, так и программный, построенный на базе операционных систем типа Unix или Windows.

Перед автором была поставлена задача построения биллинговой системы, а точнее, подсчета Интернет-трафика для пользователей домашней локальной сети.

Попытки реализовать стандартными средствами, такими, как прокси сервер (Squid, Socks), NAT, не удовлетворяли ни пользователей, ни заказчика, ни автора. Основные требования реализации проекта были следующими:

- точный подсчет трафика без ограничения возможностей пользователя (клиенту должны быть доступны все порты, любые протоколы);
- система должна автоматически осуществлять блокировку доступа пользователя в случае превышения лимита по трафику;
- возможность получения информации о состоянии счета в режиме реального времени;
- необходимость получения пользователем информации о посещении сетевых ресурсов;
- невысокая стоимость проекта;
- использование некоммерческого программного обеспечения.

Результатом работы должен оказаться шлюз с высоким уровнем безопасности и программа, написанная на языке C. Программа должна работать под

Омский государственный университет.
E-mail: ADrozd@kompas-m.ru

Unix-подобными операционными системами (с поддержкой pcap). Также создание версии программы для операционных систем типа Windows возможно в рамках реализации проекта.

Поставленная автором задача не нова, однако в рамках ее рассмотрения проблема информационной безопасности проекта является актуальной, но она широко не исследовалась.

Один из первых вариантов, который приходит в голову, – построить локальную сеть, поставить сервер, который будет считать приходящий, при необходимости и исходящий, трафик. Установить на него Unix-подобную операционную, настроить ipfw, iptables или ipchains. Данный вариант достаточно хорош в малых коммерческих организациях, где каждый друг друга знает, где администратор знает возможности пользователей, где компьютеры пользователей настраивает сам администратор (запрещает изменять IP-адрес клиенту), а самое главное — где есть административные механизмы борьбы со злоумышленниками. Данный же проект был разработан для домашних локальных сетей, в которых описанный способ борьбы неприменим.

1. Решение

Использование VPN для безопасности выхода и аутентификацию на основе Radius-сервера. Если пользователь захотел выйти в Интернет, то достаточно создать сетевое подключение типа «Подключение к виртуальной частной сети через Интернет» (в Windows-е, а у пользователей, скорее всего, именно эта операционная система), что не сложнее создания обыкновенного модемного соединения.

На сервере создана база данных, в которой хранится информация о пользователе, его платежах, информация о прошедшем трафике, ошибки соединения (удобно для нахождения злоумышленника).

Сама программа всегда «работает» на интерфейсе и по необходимости добавляет информацию о прошедшем трафике.

Предусмотрены скрипты для администратора и рядового пользователя, с помощью которых можно добавлять, изменять информацию о пользователе, просматривать ошибки и просматривать пользователем статистику.

1.1. Выбор операционной системы

Аппаратный маршрутизатор не подходит для решения поставленной задачи (основная проблема – цена), в связи с этим возникает новая подзадача – выбор программного маршрутизатора, а точнее, выбор операционной системы, она должна удовлетворять нескольким основным требованиям:

- безопасность;
- стабильность;
- гибкость в настройке.

Предпочтение было отдано программному серверу платформы Unix (FreeBSD), так как этот программный продукт зарекомендовал себя как наиболее стабильная и защищенная система из существующих серверных операционных систем.

1.2. Аутентификация и авторизация

В качестве аутентификации была использована технология RADIUS.

Radius (Remote Authentication Dial In User Service) – служба удаленной аутентификации входящих звонков пользователей. Он является протоколом для передачи информации по аутентификации, авторизации и конфигурации между сервером доступа, далее NAS, и сервером аутентификации – RADIUS.

Стандартно существует два шага в процессе аутентификационного запроса, приходящего от NAS (Network Access Server): авторизация и аутентификация.

Аутентификация должна быть разрешена в период авторизации. Основная причина этого состоит в том, что пользователь может быть не допущен до метода аутентификации в период авторизации.

В данном проекте Radius (программа freeradius) использует базу данных для хранения имен пользователей, IP-адрес, который будет выдаваться пользователю, его пароль. Этот способ хранения информации о пользователе, очевидно, очень удобен. Можно будет добавлять пользователя с помощью Perl-скрипта через Web-интерфейс, что будет показано ниже. В качестве базы данных была выбрана СУБД MySQL.

1.3. Выбор СУБД. Создание базы данных

В настоящее время существует множество бесплатно распространяемых СУБД, таких, как PostageSQL, mSQL, MySQL и так далее. Конечно, эти открытые СУБД не сравнятся с такими коммерческими гигантами, как Oracle или MS SQL Server. Но при правильной настройке, при малых и средних задачах они будут выглядеть достаточно достойно. Журналы «PC Magazine», «eWEEK» проводили тестирование пяти СУБД: DB2 7.2, MySQL 4.01, Oracle 9i, SQL Server 2000, ASE 12.5.0.1.

«Из пяти баз данных, которые мы тестировали, только Oracle 9i и MySQL были способны выполнить наше приложение 8 часов без каких-либо проблем».

«SQL Server и MySQL были самыми простыми в настройке, Oracle 9i – был самым сложным, потому что у него слишком много разделенных кешей памяти, которые можно регулировать».

В качестве СУБД был выбран MySQL. Как было показано выше, ряд его преимуществ впечатляет. Второй причиной нашего выбора был опыт работы с этой СУБД и разработка нескольких простых баз данных для компании «Компас-М». Как уже говорилось, база данных, назовем ее billing, состоит из четырех таблиц:

- Users – информация о пользователе;
- Traffic – информация о проходящем через роутер трафике;

- Money – информация о поступивших взносах от пользователей;
- Rejects – информация о всех попытках, которые отброшены по той или иной причине.

Очевидно, что самой большой таблицей в базе данных является Traffic, у MySQL есть ограничение на размер таблицы, в версии 3.22 имеет предел по размеру 4 Гб. В MySQL версии 3.23, где используется новый тип таблиц, максимальный размер таблицы доведен до 8 миллионов терабайтов (2^{63} bytes).

Доступ к MySQL осуществляется на языке C и Perl, так как сама билинговая система написана на C, а web интерфейс на языке Perl. У СУБД MySQL идет поддержка большинства операционных систем (так как проект MySQL – является «open source»). В дальнейшем, естественно, будет рассмотрено программирование под Unix.

Рассмотрим, как писать программы на языке C, взаимодействующие с сервером MySQL. При работе с клиентской библиотекой MySQL необходимо включить в программу файл mysql.h, а затем подключить файл библиотеке на этапе компоновки программы.

Листинг 1

```
#include <stdio.h>
# include <mysql/mysql.h>
int main (int argc, char *argv[])
{
MYSQL mysql;
MYSQL_RES *result;
MYSQL_ROW row;
uint num_fields, i;
ulong *lenghs;
if (!mysql_init(&mysql))
{
printf ("Невозможно инициализировать стуктуру !);
exit();
}
if (!mysql_real_connect(&mysql, "localhost",
"kursuser", "mypassword", "billing", 0, NULL,0))
{
printf ("%d: %s", mysql_errno(&mysql),
mysql_errno(&mysql));
exit();
}
if (mysql_query(&mysql, "SELECT username, ip
FROM users ORDER BY 1,2")){
printf ("%d: %s", mysql_errno(&mysql),
mysql_errno(&mysql));
```

```
}
else
{
result = mysql_store_result(&mysql);
num_fields = mysql_num_fields(result);
while (row = mysq_fetch_row(result))
{
length = mysql_fetch_length(result);
for (i=0;i < num_fields;i++)
{
printf (“[%.*s]”, (int) length[i], row[i] ? row[i] : “NULL”)
}
printf (“\n”);
}
mysql_free_result (result) ;
}
mysql_close(&mysql);
}
```

1.4. Написание программы

За основу написания программы была взята программа tcpdump, а точнее, использовалась библиотека pcap (Packet Capture library), которая, в свою очередь, основана на bpf (Berkley Packet Filter).

Для понимания гибкости и функциональных возможностей pcap приведем простейший пример с пояснениями.

Листинг 2

```
#include <pcap.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>
// Вызывается каждый раз, когда получен пакет
void analize (u_char *useless,const struct pcap_pkthdr* pkthdr,const u_char*
packet)
{
static int count = 1;
fprintf(stdout,"Пришел пакет - %d, ",count);
if(count == 7)
fprintf(stdout,"Конец эксперементам!! ");
```

```
fflush(stdout);
count++;
}
int main(int argc,char **argv)
{
int i;
char *dev;
char errbuf[PCAP_ERRBUF_SIZE];
pcap_t* descr;
const u_char *packet;
struct pcap_pkthdr hdr; // pcap.h
struct ether_header *eptr; // net/ethernet.h
if(argc != 2)
{
fprintf(stdout,"Используйте: %s количество пакетов \n",argv[0]);
return 0;
}
// какой интерфейс использовать
dev = pcap_lookupdev(errbuf);
if(dev == NULL)
{
printf("%s\n",errbuf);
exit(1);
}
// открывает для чтения
descr = pcap_open_live(dev,BUFSIZ,0,-1,errbuf);
if(descr == NULL)
{
printf("pcap_open_live(): %s\n",errbuf);
exit(1);
}
pcap_loop(descr,atoi(argv[1]),analize,NULL);
fprintf(stdout,"\nУстановленное число пакетов принято!\n");
return 0;
}
```

1.5. Web-интерфейс и написание программы

При проектировании биллинговой системы еще ни разу не говорилось об удобстве пользователя. Пользователь должен в любой момент знать, сколько трафика у него осталось, сколько он уже использовал. Естественно, что пользователь должен иметь доступ к СУБД. Был выбран Web-интерфейс, так как у него есть ряд преимуществ:

- нет необходимости в написании множества клиентов для разных операционных систем (вариант с Java не рассматриваем в связи с медлительностью приложений, написанных на нем);
- у пользователя должен быть только web-браузер, который существует во всех современных операционных системах, что придает гибкость системе;
- нет необходимости в установке клиентской части, что достаточно большой плюс для начинающих пользователей.

Первое, что нужно сделать для поставленной задачи, – выбрать web-сервера. Web-серверы, бесспорно, являются самыми известными в мире сервисами системы Unix. Существует много других установленных сервисов, но ни один из них не известен так, как web-сервер (или HTTP-сервер). В качестве Web-сервера был выбран Apache, так как он зарекомендовал себя с наилучшей стороны.

Некоторые Web-серверы содержат встроенный интерпретатор Perl, что позволяет генерировать документы на Perl без запуска нового процесса. Системные издержки на чтение не изменившейся страницы пренебрежимо малы для страниц с редкими обращениями (даже порядка нескольких обращений в секунду). Однако вызовы CGI существенно замедляют компьютер, на котором работает Web-сервер.

Приведем простейший сценарий для проверки пользователем своего баланса. Большая часть скрипта должна быть уже понятна (простой доступ к базе данных).

Листинг 3

```
#!/usr/bin/perl
use DBI;
use CGI qw (:standart : html3);
$name = param (“Name”);
# использование модуля CGI делает разработку программы более удобной.
print header(),
start_html (“Limit Query”),
start_form(),
p(“Введите имя пользователя”, textfield (“Name”)),
submit(),
end_form();
if (defined $name)
{
$dbh = DBI->connect (“DBI:mysql:billing:localhost:3306”,”root”,”mypassword”);
or die “Connecting: $DBI::errstr”;
$sth=$dbh->prepare (“SELECT username, limit FROM users WHERE username
= $name”)
or die “Preparing: ”, $dbh->errstr;
$sth->execute
```

```
or die “Executing: ”, $sth->errstr;
print h1(“Результат запроса”), “<TABLE BORDER =1>”;
while (@row = $sth->fetchrow())
{
print Tr( td(\@row));
}
print “</TABLE> \n”;
$sth->finish;
$dbh->disconnect;
}
print end_html();
```

2. Результаты

В результате получилась достаточно стабильная система, которая проверялась на следующем оборудовании в течение нескольких месяцев: Celeron-1000/256Mb/20Gb, с пропускной способность канала 115.5 Кб/с, на 15 машин. Средняя нагрузка на канал — 4-5 Гб в месяц.

2.1. Примеры использования

Данная система применялась в компании «Компас-М» и в домашней локальной сети по улице Масленникова 45.

2.2. Перспективы развития

Данную систему можно серьезно улучшить как в управлении, так и в установке. Перспективы развития описаны в нескольких пунктах:

- изменение структуры базы данных, добавление возможности зон (например, местный трафик дешевле и тому подобное);
- улучшение web-интерфейса;
- добавление возможности конфигурационного файла;
- возможно построение e-mail рассылки пользователям (например, каждую неделю посылать в письме, сколько трафика у них осталась, или же если лимит стал равным критическому, то посылать письмо и администратору и пользователю);
- написание порта для ОС FreeBSD.

Полученный опыт и отдельные части программы могут использоваться в других задачах, таких, как:

- поиск злоумышленника внутри компании, использующего доступ в Интернет не для работы;

- обнаружение всевозможных атак, с помощью скриптов существует возможность блокирования злоумышленника;
- любые задачи, связанные с анализом сетевого трафика.

3. Заключение

Стабильная, надежная, безопасная биллинговая система является необходимым условием для успешной работы провайдинговых компаний. Разработанная система удовлетворяет основным требованиям, которые поставил заказчик. Можно выделить следующие плюсы данного проекта:

- не требует денежных затрат;
- не использовались коммерческие программы, что делает систему в какой-то мере независимой;
- возможно детальное протоколирование пакетов;
- гибкость системы;
- стабильность гарантируется операционной системы FreeBSD (у автора сервер на базе этой операционной системы пробыл без перезагрузки более пяти месяцев, что не является рекордом, перезагрузка осуществилась из-за долгосрочного отключения электричества);
- сложность в воровстве трафика у пользователя достигается за счет использования Radius-сервера, VPN-технологии (не рассматривается случай, когда пользователь сам выдаст пароль по собственной невнимательности);
- простота в настройке пользовательского компьютера.

В результате проведенной работы сформировалась биллинговая система, которая, к сожалению, неприменима в коммерческих целях из-за законодательства Российской Федерации, разрешающего использовать только сертифицированные биллинговые системы. В начале работы данный факт не был известен ни автору, ни заказчику. В итоге возможна сертификация данной системы, но существует большая проблема – деньги, которые нужно отдать за сертификацию. К сожалению, одному заказчику заниматься этим просто невыгодно, легче купить сертифицированный продукт.

Литература

1. Кристиансен Т., Торкингтон. Perl библиотека программиста. СПб.: Питер, 2001.
2. Аткинсон Т. MySQL. М.: Издательский дом «Тильямс», 2002.
3. Руководство FreeBSD. – http://www.freebsd.org.ua/handbook/p42.html

4. Manpage IPFW. – http://www.opennet.ru/docs/RUS/ipfw/ipfw.html
5. An Analysis of the RADIUS Authentication Protocol.
   – http://www.untruth.org/josh/security/radius/radius-auth.html
6. MySQL Manual. – http://www.mysql.com/doc/ru/index.html
7. Making a Connection with tcpdump.
   – http://www.linuxjournal.com/article.php?sid=6446&mode=thread&order=0
8. Manpage of PCAP. – http://www.tcpdump.org/pcap3_man.html
9. Programming with pcap. – http://www.tcpdump.org/pcap.htm
10. Packet Capture With libpcap and other Low Level Network Tricks.
    – http://www.cet.nau.edu/∼mc8/Socket/Tutorials/section1.html