Настройка PPPoE-сервера в Debian

Настройка PPPoE-сервера в Debian
Продолжаю тему пакета RP-PPPoE. На этот раз я решил настроить PPPoE-сервер из этого пакета. PPPoE-сервер, как и клиент, может работать в двух режимах: в режиме ядра и в пространстве пользователя.

Но прежде чем перейти к описанию настройки сервера, опишу настройки клиента. И клиент и сервер находятся в пакете pppoe Debian. Таким образом установка сервера и клиента весьма просты:

```
aptitude install pppoe
```

1. Настройка тестового PPPoE-клиента

Клиент настраивался в режиме ядра, маршрут по-умолчанию не устанавливался, номер создаваемого интерфейса был прописан постоянным - 2.

Ниже - содержимое файла /etc/ppp/peers/dsl-provider, содержащий настройки тестового клиента:

```
user test
plugin rp-pppoe.so
eth1
noipdefault
usepeerdns
persist
noauth
#defaultroute
unit 2
```

Для пользователя test в файл /etc/ppp/chap-secrets добавим такую строчку:

```
test            *               testpasswd      *
```

На этом настройка клиента закончена. После того, как будет настроен сервер, мы установим подключение при помощи команды:

```
pon dsl-provider
```

2. Настройка PPPoE-сервера в пространстве пользователя

Тут особых сложностей тоже не возникло. В файле /etc/ppp/pppoe-server-options были прописаны следующие опции:

```
auth
require-chap
ms-dns 81.30.199.5
ms-dns 81.30.199.94
noipdefault
noipx
nodefaultroute
noproxyarp
netmask 255.255.255.255
logfile /var/log/pppoe-server.log
```

Опция auth заставляет потребовать у удалённого клиента аутентификации, опция require-chap указывает способ аутентификации CHAP.

Опции ms-dns передают клиенту IP-адреса DNS-серверов, которые клиент воспринимает если в его настройках указазана опция usepeerdns.

Опция noipdefault не позвояет соглашаться с настройками IP-адресов, полученных от клиента, noipx отключает согласование протокола IPX, nodefaultroute запрещает добавлять маршрут по умолчанию через клиента, noproxyarp запрещает серверу создавать для клиентов ARP-записи на Ethernet-интерфейсах (эта опция нужна только если клиентам выдаются IP-адреса, пересекающиеся с IP-адресами Ethernet-сети, чтобы PPPoE-клиенты и компьютеры в локальной сети могли беспрепятственно обмениваться трафиком).

Опция "netmask 255.255.255.255" задаёт маску PPP-соединения у клиента и сервера, опция "logfile /var/log/pppoe-server.log" предписывает вести серверу журнал. Пользы от этого журнала не очень много, т.к. в нём нет отметок времени.

По-умолчанию сервер передаёт в pppd также ещё и следующие опции: nodetach, noaccomp, nobsdcomp, nodeflate, nopcomp, novj, novjccomp, default-asyncmap.

Все опции по-умолчанию имеют приоритет над опциями из файла /etc/ppp/pppoe-server-options. Все опции, кроме первой и последней, отключают различные виды сжатия. Первая и последняя опции, на мой взгляд, спорные. Первая указывает pppd не отделяться от управляющего терминала. Какой может быть управляющий терминал, если pppd запущен из демона, то есть из программы, которая сама не имеет управляющего терминала? Последняя опция предписывает pppd не согласовывать параметр asyncmap и принудительно экранировать все управляющие символы, что не имеет особого смысла, поскольку Ethernet-среда ни коим образом не интерпретирует управляющие символы. Возможно эта опция предназначена для DSL-линии. Она, будучи последовательной линией, может использовать асинхронный режим приёма/передачи, в котором приёмник сообщает передатчику о готовности или неготовности к приёму информации с помощью специальных символов XON или XOFF.

Заведём тестового пользователя в файле /etc/ppp/chap-secrets на PPPoE-сервере, добавив строчку:

```
test            *               testpasswd              192.168.0.2
```

Последнее поле сообщает IP-адрес, который будет назначен этому клиенту в случае успешной аутентификации.

Теперь запустим сервер. У сервера есть несколько важных опций:
-I - задаёт имя интерфейса, на котором сервер будет ожидать соединений,
-L - задаёт собственный IP-адрес сервера внутри канала PPP,
-p - позволяет задать файл, содержащий список IP-адресов, выдаваемых клиентам внутри канала PPP,

-O - позволяет задать другой файл вместо используемого по умолчанию /etc/ppp/pppoe-server-options,
-N - позволяет указать максимальное количество одновременных соединений, по умолчанию 64.
Я запустил так:

```
pppoe-server -I eth1 -L 192.168.0.1
```

Сервер запускается и переходит в фоновый режим. После этого запускаем на другом компьютере PPPoE-клиента:

```
pon dsl-provider
```

Связь успешно установилась и на клиенте был создан интерфейс ppp2 с IP-адресами 192.168.0.2 и 192.168.0.1,
на на сервере был создан интерфейс ppp0 с IP-адресами 192.168.0.1 и 192.168.0.2.
Теперь можно проверить работу связи. Для этого я на клиенте запустил команду:

```
ping -S 192.168.0.2 192.168.0.1
```

А на сервере запустил команду:

```
tcpdump -i ppp0
```

Можно было увидеть, что клиент получает от сервера отклики на эхо-запросы, а сервер принимает и отправляет ICMP-сообщения "echo request" и "echo reply".

3. Настройка PPPoE-сервера в режиме ядра

Здесь, казалось бы, всё просто: добавляем серверу опцию -k и вперёд, но оказалось всё не так просто. В Debian pppoe-server собран без поддержки режима ядра
и наотрез отказывается принимать опцию -k. Я решил собрать pppoe самостоятельно из deb-src.
Направление для мыслей задал вот этот багрепорт: #298185 - pppoe-server: invalid option -- k - Debian Bug report logs. После долгого процесса копания по
опциям получилось такое mini-how-to:

Устанавливаем зависимости для построения pppoe:

```
apt-get build-dep pppoe
```

Скачиваем и распаковываем в текущий каталог исходники:

```
apt-get source pppoe
```

Переходим в каталог rp-pppoe-3.8/src и выполняем конфигурирование:

```
cd rp-pppoe-3.8/src
./configure
```

Теперь в только что созданном файле config.h нужно заменить строчку

```
/* #undef HAVE_LINUX_KERNEL_PPPOE */
```

на строчку

```
#define HAVE_LINUX_KERNEL_PPPOE 1
```

Теперь выходим из каталога src и выполняем сборку, указав путь к плагину rp-pppoe.so, который как раз и реализует поддержку PPPoE-сеансов на уровне ядра:

```
cd ..
./debian/rules PLUGIN_PATH=/usr/lib/pppd/2.4.4/rp-pppoe.so
```

Собираем пакет и устанавливаем его:

```
./debian/rules binary
cd ..
dpkg -i pppoe_3.8-3_i386.deb
```

Заморозим пакет в системе, чтобы предотвратить его автоматическое обновление:

```
echo pppoe hold | dpkg --set-selections
```

С этого момента нужно как минимум при каждом обновлении системы просматривать список обновлений. Если для pppoe будет выпущено обновление, закрывающее уязвимость, необходимо будет собрать пакет из свежих исходников заново, снять фиксацию пакета в системе с помощью команды:

```
echo pppoe install | dpkg --set-selections
```

Затем заменить пакет новым и заново зафиксировать. Всё это - неизбежная плата за использование самосборного пакета. Конечно можно автоматизировать пересборку пакета, написав shell-скрипт, запускающий в нужной последовательности все указанные команды и заменяющий нужную строчку с помощью sed.

Так или иначе, теперь в системе установлен пакет с PPPoE-сервером, позволяющим поддерживать PPPoE-соединения на уровне ядра.

Для запуска сервера в этом режиме, к команде описанной в предыдущем разделе нужно добавить опцию -k:

```
pppoe-server -I eth1 -L 192.168.0.1 -k
```

Можно проверить связь так же, как это было описано выше, у меня всё заработало.

Источник статьи