



Опыт сообщества в кратком изложении

Освоение OpenVPN

Освоение создания и интеграции защищенных частных сетей с использованием OpenVPN

Эрик Ф Крист

Ян Юст Кейзер

[PACKT] open source*
PUBLISHING

Освоение OpenVPN

Оглавление

Освоение OpenVPN.....	6
Кредиты.....	7
Об авторах.....	8
О рецензентах.....	9
www.PacktPub.com.....	10
Зачем подписываться?.....	10
Бесплатный доступ для владельцев аккаунтов Packt.....	10
Предисловие.....	11
О чём рассказывает эта книга.....	11
Что вам нужно для этой книги.....	13
Для кого предназначена эта книга.....	13
Конвенции.....	13
Загрузка примеров кода.....	14
Обратная связь с читателем.....	14
Поддержка клиентов.....	14
Загрузка примера кода.....	14
Список опечаток.....	14
Пиратство.....	15
Вопросы.....	15
Глава 1. Введение в OpenVPN.....	16
Что такое VPN?.....	16
Типы VPN.....	19
PPTP.....	19
IPSec.....	19
VPN на основе SSL.....	20
OpenVPN.....	21
Сравнение VPN.....	21
Преимущества и недостатки PPTP.....	21
Преимущества и недостатки IPSec.....	22
Преимущества и недостатки VPN на основе SSL.....	22
Преимущества и недостатки OpenVPN.....	22
История OpenVPN.....	22
Пакеты OpenVPN.....	26
Версия с открытым исходным кодом (общество).....	26
Закрытые исходники (коммерческий) Access Server.....	26
Мобильная платформа (смешанная) OpenVPN/OpenVPN Connect.....	26
Другие платформы.....	27
Внутренности OpenVPN.....	27
Драйвер tun/tap.....	27
Режимы UDP и TCP.....	29
Протокол шифрования.....	29
Каналы управления и передачи данных.....	30
Алгоритмы шифрования и хеширования.....	30
OpenSSL против PolarSSL.....	32
Резюме.....	32
Глава 2. Режим точка-точка.....	33
Плюсы и минусы ключевого режима.....	33
Первый пример.....	34
Протокол TCP и разные порты.....	36
Режим TAP.....	36

Топология subnet.....	36
Туннель открытого текста.....	37
Секретные ключи OpenVPN.....	38
Использование нескольких ключей.....	40
Использование разных алгоритмов шифрования и аутентификации.....	41
Маршрутизация.....	42
Конфигурационные файлы и командная строка.....	44
Полная настройка.....	44
Расширенная настройка без-IP.....	47
Трехсторонняя маршрутизация.....	49
Маршрут, net_gateway, vpn_gateway и метрики.....	52
Мостовой tap-переходник на обоих концах.....	52
Удаление мостов.....	55
Совмещение режима точка-точка с сертификатами.....	55
Резюме.....	57
Глава 3. PKI и сертификаты.....	58
Обзор PKI.....	58
PKI с использованием EasyRSA.....	60
Создание СА.....	62
Список отзыва сертификатов.....	63
Серверные сертификаты.....	64
Клиентские сертификаты.....	65
PKI с использованием ssl-admin.....	66
Серверные сертификаты OpenVPN.....	70
Клиентские сертификаты OpenVPN.....	72
Другие преимущества.....	74
Несколько СА и CRL.....	74
Дополнительная безопасность - аппаратные токены, смарт-карты и PKCS #11.....	74
Исходная информация.....	74
Поддерживаемые платформы.....	75
Инициализация аппаратного токена.....	76
Генерация пары сертификат/приватный ключ.....	76
Генерация приватного ключа на токене.....	77
Генерация запроса на сертификат.....	77
Запись сертификата X.509 в токен.....	78
Получение идентификатора аппаратного токена.....	79
Использование аппаратного токена с OpenVPN.....	79
Резюме.....	79
Глава 4. Режим клиент-сервер с устройствами tun.....	80
Понимание режима клиент-сервер.....	80
Настройка инфраструктуры открытых ключей.....	81
Начальная настройка режима клиент-сервер.....	82
Подробное объяснение файлов конфигурации.....	84
Topology subnets против topology net30.....	87
Добавление повышенной безопасности.....	88
Использование ключей tls-auth.....	88
Генерация ключа tls-auth.....	88
Проверка атрибутов использования ключа сертификата.....	89
Основные конфигурационные файлы производственного уровня.....	90
Конфигурация на основе TCP.....	91
Конфигурационные файлы для Windows.....	92
Маршрутизация и маршрутизация на стороне сервера.....	93
Специальные параметры для вариантов маршрута.....	95

Маскарадинг.....	96
Перенаправление шлюза по умолчанию.....	96
Специфичная для клиента конфигурация - файлы CCD.....	98
Как определить, правильно ли обрабатывается файл CCD.....	99
CCD файлы и топология net30.....	100
Клиентская маршрутизация.....	100
Подробное объяснение конфигурации client-config-dir.....	101
Трафик клиент-клиент.....	102
Файл состояния OpenVPN.....	102
Надежное отслеживание соединения в режиме UDP.....	104
Интерфейс управления OpenVPN.....	104
Пересогласование ключей сеанса.....	106
Замечание об устройствах PKCS#11.....	107
Использование IPv6.....	107
Защищенный трафик IPv6.....	108
Использование IPv6 в качестве транзита.....	109
Расширенные параметры конфигурации.....	110
ARP-прокси.....	110
Назначение публичных IP-адресов клиентам.....	113
Резюме.....	116
Глава 5. Расширенные сценарии развертывания в tun режиме.....	117
Включение общего доступа к файлам через VPN.....	117
Использование имен NetBIOS.....	120
Использование nbtstat для устранения проблем с подключением.....	121
Использование LDAP в качестве механизма внутренней аутентификации.....	122
Устранение неполадок в аутентификации LDAP.....	124
Фильтрация OpenVPN.....	125
Пример FreeBSD.....	126
Пример Windows.....	127
Маршрутизация на основе политик.....	134
Сетевые расположения Windows - общественные или частные.....	135
Бэкграунд.....	135
Изменение местоположения адаптера TAP-Win с помощью redirect-gateway.....	136
Использование OpenVPN с HTTP или SOCKS прокси.....	143
HTTP прокси.....	144
SOCKS прокси.....	144
Резюме.....	145
Глава 6. "Режим клиент-сервер" с tap-устройствами.....	146
Базовая настройка.....	146
Включение трафика клиент-клиент.....	148
Фильтрация трафика между клиентами.....	149
Использование устройства tap (мост).....	153
Соединение в Linux.....	155
Соединение в Windows.....	158
Использование внешнего DHCP-сервера.....	162
Проверка широковещательного и не IP-трафика.....	164
Протокол разрешения адресов трафика.....	164
Трафик NetBIOS.....	165
Сравнение режима tun с режимом tap.....	166
Слой 2 против слоя 3.....	167
Маршрутные различия и iroute.....	167
Фильтрация клиент-клиент.....	168

Широковещание трафика и "болтливость" сети.....	168
Мост.....	168
Резюме.....	168
Глава 7. Скрипты и плагины.....	170
Скриптинг.....	170
Серверные скрипты.....	171
Клиентские скрипты.....	173
Примеры серверных скриптов.....	174
Плагины.....	194
down-root.....	194
Плагин auth-pam.....	194
Резюме.....	195
Глава 8. Использование OpenVPN на мобильных устройствах и домашних маршрутизаторах.	197
Использование OpenVPN для приложения Android.....	197
Создание профиля приложения OpenVPN.....	198
Использование файла PKCS#12.....	202
Использование приложения OpenVPN Connect для Android.....	203
Использование приложения OpenVPN Connect для iOS.....	204
Интеграция смартфонов в существующую настройку VPN.....	209
Использование домашнего маршрутизатора в качестве VPN-клиента.....	209
Использование домашнего маршрутизатора в качестве VPN-сервера.....	212
Резюме.....	215
Глава 9. Устранение неполадок и настройка.....	216
Как читать файлы журнала.....	216
Обнаружение неработающей установки.....	220
Исправление распространенных ошибок конфигурации.....	221
Неправильный сертификат CA в конфигурации клиента.....	221
Сертификат клиента не распознан сервером.....	222
Несоответствие сертификата клиента и закрытого ключа.....	223
Несоответствие ключей auth и tls-auth.....	224
Несоответствие размера MTU.....	226
Несоответствие шифрования.....	227
Несоответствие сжатия.....	228
Несоответствие фрагмента.....	229
Несоответствие tun и tap.....	229
Проблемы с client-config-dir.....	230
Нет доступа к устройству tun в Linux.....	231
Отсутствие повышенных привилегий в Windows.....	233
Устранение проблем с маршрутизацией.....	234
Рисование детальной картины.....	234
Начните с середины и двигайтесь наружу.....	236
Найдите время, чтобы временно отключить брандмауэр.....	239
Если ничего не помогает – используйте tcpdump.....	239
Как оптимизировать производительность с помощью ping и iperf.....	239
Использование ping.....	240
Использование iperf.....	240
Гигабитная сеть.....	242
Анализ трафика OpenVPN с помощью tcpdump.....	242
Резюме.....	245
Глава 10. Будущие направления.....	247
Сильные стороны.....	247
Текущие недостатки.....	247
Масштабирование на гигабитных скоростях и выше.....	247

Куда идем.....	249
Улучшение поддержки сжатия.....	249
Сжатие на клиенте.....	249
Новые криптографические процедуры.....	250
Смешанная аутентификация сертификата/имени пользователя.....	250
Поддержка IPv6.....	250
Разделение привилегий Windows.....	250
Резюме.....	252

Освоение OpenVPN

Copyright © 2015 Packt Publishing Все права защищены. Никакая часть этой книги не может быть воспроизведена, сохранена в поисковой системе или передана в любой форме или любым способом без предварительного письменного разрешения издателя, за исключением кратких цитат, вложенных в критические статьи или обзоры.

При подготовке этой книги были приложены все усилия для обеспечения точности представленной информации. Однако информация, содержащаяся в этой книге, преподается без каких-либо гарантий, явных или подразумеваемых. Ни авторы, ни издательство Packt Publishing, а также его дилеры и дистрибуторы не несут ответственности за любой ущерб, причиненный или предположительно причиненный прямо или косвенно этой книгой.

Packt Publishing стремилось предоставить информацию о товарных знаках всех компаний и продуктов, упомянутых в этой книге, путем надлежащего использования капиталов. Однако Packt Publishing не может гарантировать точность этой информации.

Впервые опубликовано: август 2015 г.

Ссылка на производство: 1260815

Опубликовано компанией Packt Publishing Ltd.

Livery Place
35 Livery Street
Birmingham B3 2PB, UK.
ISBN 978-1-78355-313-6

www.packtpub.com

Кредиты

Авторы

Эрик Ф Крист
Ян Просто Кейсер

Рецензенты

Стефан Агнер
Эммануэль Бретель
Майкл А Коссенас
Гийом Дестюндер

Выпускающий редактор

Амарабха Банерджи

Редакторы комплектования

Ричард Брукс-Блэнд
Лариса Пинто

Редактор разработки контента

Пуджа Наир

Технический редактор

Митали Сомайя

Редактор

Рошни Банерджи
Рахи Савант Рахи Савант

Координатор проекта

Джуди Хоце
Корректор
Сафис Эдитинг

Индексатор

Гемангини Бари

Графика

Sheetal Aute

Координатор производства

Нитеш Тхакур

Лицензированная разработка

Нитеш Тхакур

Перевод и верстка перевода

Дмитрий Кузнецов



Об авторах

Эрик Ф Крист - ИТ-специалист с опытом работы в области интеграции аппаратных и программных систем. Вместе с несколькими другими он сыграл ключевую роль в создании сообщества OpenVPN таким, каким оно является сегодня. Он работает в области исследований и развития как основной специалист компьютерной системы для Санкт-Джуд Медикал. Его роль включает в себя системную инженерию, управление конфигурациями и анализ кибербезопасности для продуктов, связанных с отделом технологий сердечно-сосудистой аблации.

Вы можете найти его онлайн в IRC-сетях Freenode и EFNet как `ecrist`. Он называет город-побратим Миннесотой своим домом и живет с женой Диidi, сыном Лансом и дочерью Тейлор.

Ян Джаст Кейсер - профессионал по работе с открытым исходным кодом из Утрехта, Нидерланды. Он имеет широкий спектр опыта в области ИТ, начиная от предоставления поддержки пользователям, системного администрирования и системного программирования и заканчивая сетевым программированием. С 1989 года он работал в различных ИТ-компаниях. С 1995 года он работает в основном на платформах Unix/Linux. Он был активным участником USENET в начале 1990-х годов.

В настоящее время он работает старшим научным программистом в Амстердаме, Нидерланды, в Нихэфе, Институте субатомной физики голландского фонда фундаментальных исследований материи (FOM - ФИМ). Он работает над многоядерными вычислительными системами, грид-вычислениями, а также приложениями для смарт-карт. Его интересы с открытым исходным кодом включают все типы виртуальных частных сетей, включая IPSec, PPTP и, конечно же, OpenVPN. В 2004 году он открыл OpenVPN и с тех пор использует его.

Его первой книгой была *Книга рецептов OpenVPN 2*, Издательства Packt.

О рецензентах

Стефан Агнер получил степень бакалавра информационных технологий в Люцернском Университете прикладных наук и искусств в 2009 году и с тех пор работает в области встраиваемых систем в качестве инженера-программиста. Он специализируется на разработке драйверов и системном программировании и предпочитает работать со стеком программного обеспечения с открытым исходным кодом. В настоящее время он работает над восходящей поддержкой Linux для ARM-based Freescale Vybrid SoC для своего работодателя Toradex AG.

Он описывает себя как энтузиаста с открытым исходным кодом, который работает с Linux и другими свободными программами не только в своей профессиональной жизни инженера-программиста, но и в свободное время. В нескольких небольших компаниях он успешно развернул и управлял OpenVPN в качестве основного VPN-решения. Для своей частной ИТ-инфраструктуры он запускает маршрутизаторы на базе OpenWrt, которые служат OpenVPN серверами. Он также любит вести блог о технических вещах, таких как увлекательные проекты и интересные проблемы, с которыми он сталкивается.

Эммануэль Бретель имеет 10-летний опыт работы в области devops, системного и сетевого администрирования. Он использовал OpenVPN, его возможности плагинов и кросс-платформенную совместимость чтобы помочь подключить сотрудников по всему миру к корпоративным сетям.

Он также разработал и плагины с открытым исходным кодом для OpenVPN: openvpn-mysql-auth и openvpn-ldap-auth.

Когда он не возится с новыми технологиями или не автоматизирует свой выход - он любит путешествовать и отдыхать.

Майкл Коссенас - администратор Linux/network из Афин, Греция.

Он работал специалистом по сетевой безопасности в Digital Sima, компании, специализирующейся на сетях LAN/WAN. Сейчас он работает субподрядчиком в IBM Greece и управляет более чем 50 серверами Linux на базе SUSE как один из их клиентов.

Его первый опыт работы с Linux произошел еще в 1998 году, когда он использовал RedHat 5.2. Затем он работал над различными проектами с открытым исходным кодом, включая Zimbra, DRBD, KVM и Postfix.

Он также является модератором форума OpenVPN.

Он работает субподрядчиком в IBM Greece в отделе SO (Strategic Outsourcing).

Я хотел бы поблагодарить мою семью (мою жену Фрозо, моего сына Антония и мою дочь Кейт) за поддержку в трудные времена.

www.PacktPub.com

Файлы поддержки, электронные книги, скидки и многое другое

Для получения файлов поддержки и загрузок, связанных с вашей книгой, пожалуйста, посетите www.PacktPub.com.

Знаете ли вы, что Packt предлагает электронные версии каждой опубликованной книги с доступными файлами PDF и ePub? Вы можете перейти на версию электронной книги по адресу www.PacktPub.com и как клиент печатной книги, вы имеете право на скидку на копию электронной книги. Свяжитесь с нами по адресу <service@packtpub.com> для получения более подробной информации.

На www.PacktPub.com вы также можете прочитать коллекцию бесплатных технических статей, подписаться на ряд бесплатных информационных бюллетеней и получить эксклюзивные скидки и предложения на книги Packt и электронные книги.



<https://www2.packtpub.com/books/subscription/packtlib>

Вам нужны мгновенные решения ваших ИТ-вопросов? PacktLib - это онлайн-библиотека цифровых книг Packt. Здесь вы можете искать, получать доступ и читать всю библиотеку книг Packt.

Зачем подписываться?

- Возможность поиска по каждой книге, изданной Packt
- Копирование и вставка, печать и закладка содержимого
- По запросу и доступ через веб-браузер

Бесплатный доступ для владельцев аккаунтов Packt

Если у вас есть учетная запись в Packt at www.PacktPub.com, вы можете использовать её для доступа к PacktLib сегодня и просмотра 9 полностью бесплатных книг. Просто используйте свои учетные данные для немедленного доступа.

Предисловие

Конфиденциальность и безопасность в интернете и в частных сетях вызывает все большую озабоченность и все чаще встречается в новостях, где есть нарушения любого из них. Виртуальные частные сети (VPN) были созданы из потребности в защищенной связи. Наиболее популярным и широко используемым программным обеспечением с открытым исходным кодом на сегодняшний день является OpenVPN. Освоение OpenVPN направлена на обучение вас развертыванию, устранению неполадок и настройке OpenVPN, а также предоставляет надежные варианты использования для различных сценариев.

О чем рассказывает эта книга

[В главе 1.](#) *Введение в OpenVPN* рассматривают различные типы виртуальных частных сетей и некоторые их сильные и слабые стороны. PPTP, OpenVPN, IPSec и другие протоколы также обсуждаются в этой главе.

[Глава 2.](#) *Режим точка-точка* охватывает корни OpenVPN - режим точка-точка и первоначально только поддерживаемый режим. Она также охватывает tap-режим в мостовом сценарии и необычной конфигурации.

[Глава 3.](#) *PKI и сертификаты* объясняет сложную концепцию сертификатов X.509 и PKI с примерами и демонстрацией нескольких утилит. Она также описывает, как создать цепочку сертификатов и развернуть эту цепочку в их VPN.

[Глава 4.](#) *Режим клиент/сервер с tun-устройствами* проведет вас через наиболее распространенный режим развертывания - tun или маршрутизуемый, и его настройку. В ней также обсуждается передача поддерживаемых клиентами маршрутов наряду с IPv4 и IPv6.

[Глава 5.](#) *Расширенные сценарии развертывания в туннельном режиме* охватывает маршрутизацию на основе политик и настройку OpenVPN для интеграции Ваших VPN-клиентов с остальной частью локальной сети. Рассматриваются сложные примеры tun-режима, показывающие, что они подходят даже в продвинутых сценариях.

[Глава 6.](#) *"Режим клиент/сервер" с tap-устройством* рассматривает часто не менее широко используемые развернутые tap или мостовой режим VPN. Солидные примеры широковещательного и OSI-трафика уровня 2 продемонстрированы в этой главе.

[Глава 7.](#) *Сценарии и плагины* поможет вам получить представление о методах расширения VPN, включая аутентификацию, маршрутизацию и усовершенствования протокола. Эта глава помогает администратору создать локальное взаимодействие для работника или пользователя в дороге.

[Глава 8.](#) *Использование OpenVPN на мобильных устройствах и домашних маршрутизаторах* поможет вам узнать, как использовать операционные системы и функции домашнего маршрутизатора для развертывания OpenVPN. Мы понимаем, что это не просто корпоративные или коммерческие пользователи, желающие защитить свою конфиденциальность и данные. Все чаще домашние пользователи стремятся установить безопасные соединения со своими домашними ресурсами.

[Глава 9.](#) *Устранение неполадок и настройка* поможет вам стать экспертом в развертывании OpenVPN, научившись устранять неполадки и ошибки. Возможность идентифицировать проблемы создает стабильную и надежную установку и доверие ваших пользователей.

[Глава 10.](#) *Будущие направления*, даст вам краткую историю и более подробное обсуждение будущего направления OpenVPN, а также раскрывает мышление разработчиков. Она также поможет вам понять причины и историю различных решений, лежащих в основе функций и ошибок.

Что вам нужно для этой книги

Вы должны иметь следующие вещи для полного опыта чтения и последующего Освоения OpenVPN:

- Система Unix, Linux или Mac OS X
- Система Windows
- Сервер (Windows или Linux, в зависимости от предпочтения, Linux или FreeBSD)
- Четкое понимание (101 или 201 уровень) сетей (UDP и TCP через IP)

IRC-клиент или веб-браузер также будут полезны. Когда у вас возникнут проблемы или слишком много вопросов - зайдите в #openvpn на irc.freenode.net и ищите @janjust или @esgrist. Мы с нетерпением ждем возможности поговорить с вами!

Для кого предназначена эта книга

Эта книга действительно предназначена для тех, кто хочет развернуть решение VPN в любой частной или корпоративной сети. OpenVPN можно использовать для туннелей точка-точка, внутрисетевых соединений и дорожных воинов. Концепции, описанные в этой книге, могут быть применены в целом не только к развертываниям OpenVPN, за исключением специфики аргументов конфигурации.

Конвенции

В этой книге вы найдете ряд стилей текста, различающих виды информации. Вот некоторые примеры этих стилей и объяснения их значений.

Кодовые слова в тексте, имена таблиц базы данных, имена папок, имена файлов, расширения файлов, пути, фиктивные URL-адреса, вводимые пользователем и дескрипторы Twitter отображаются следующим образом: "Вы можете указать дайджест сообщения в качестве параметра для параметра `--auth`."

Блок кода задается следующим образом:

```
proto udp
port 1194
dev tun
server 10.200.0.0 255.255.255.0
```

Когда мы хотим привлечь ваше внимание к определенной части блока кода, соответствующие строки или элементы выделяются полужирным шрифтом:

```
proto udp
port 1194
dev tun
server 10.200.0.0 255.255.255.0
```

Любой ввод или вывод командной строки записывается следующим образом:

```
# mkdir -p /etc/openvpn/movpn
```

Обратите внимание, что первый символ (приглашение) используется для обозначения оболочки root (#) или оболочки пользователя (\$).

Новые термины и важные слова выделены жирным шрифтом. Слова, которые вы видите на экране, например в меню или диалоговых окнах, появляются в тексте следующим образом: "Запустите приложение OpenVPN GUI, выберите конфигурацию `basic-udp-client` и нажмите кнопку **Connect**."

Заметка

Предупреждения или важные заметки появляются в таком поле.

Подсказка

Подсказки и рекомендации выглядят так.

Загрузка примеров кода

Вы можете скачать примеры файлов кода для всех книг Packt, которые вы приобрели в своем аккаунте по адресу <http://www.packtpub.com>, если вы приобрели эту книгу в другом месте, то можете посетить <http://www.packtpub.com/support> и зарегистрироваться, чтобы файлы отправлялись вам по электронной почте.

Обратная связь с читателем

Обратная связь от наших читателей всегда приветствуется. Дайте нам знать, что вы думаете об этой книге — что вам понравилось или, возможно, не понравилось. Обратная связь с читателями важна для нас для разработки наименований, от которых вы действительно получите максимальную отдачу.

Чтобы отправить нам общую обратную связь - просто отправьте электронное письмо по адресу <feedback@packtpub.com>, и упомяните название книги в теме Вашего сообщения.

Если есть тема, в которой у вас есть опыт, и вы заинтересованы либо в написании книги, либо в ее участии, обратитесь к нашему авторскому руководству по этой теме www.packtpub.com/authors.

Поддержка клиентов

Теперь, когда вы являетесь гордым владельцем книги Packt, у нас есть ряд вещей, которые помогут вам получить максимальную отдачу от вашей покупки.

Загрузка примера кода

Вы можете скачать примеры файлов кода для всех книг Packt, которые вы приобрели в своей учетной записи по адресу <http://www.packtpub.com> если вы приобрели эту книгу в другом месте - вы можете посетить <http://www.packtpub.com/support> и зарегистрироваться для отправки файлов по электронной почте непосредственно вам.

Список опечаток

Хотя мы позаботились о том, чтобы обеспечить точность нашего контента, ошибки все же случаются. Если вы обнаружите ошибку в одной из наших книг — возможно ошибку в тексте или коде — мы будем признательны, если вы сообщите нам об этом. Поступая таким образом вы можете избавить других читателей от разочарования и помочь нам улучшить последующие версии этой книги. Если вы обнаружите какие-либо ошибки, пожалуйста, сообщите о них, посетив сайт <http://www.packtpub.com/submit-errata>, выбрав свою книгу, нажав на ссылку **формы подачи ошибок (errata submission form)** и введя сведения о ваших ошибках. Как только ваши ошибки будут проверены - ваша заявка будет принята и эти ошибки будут загружены на наш веб-сайт или добавлены в любой список существующих ошибок в разделе ошибок этого заголовка.

Чтобы просмотреть ранее отправленные ошибки, перейдите по ссылке <https://www.packtpub.com/books/content/support> и введите название книги в поле поиска. Необходимая информация появится в разделе **Errata**.

Пиратство

Пиратство авторских материалов в Интернете является постоянной проблемой во всех средствах массовой информации. В компании Packt мы очень серьезно относимся к защите наших авторских прав и лицензий. Если вы наткнетесь на какие-либо незаконные копии наших работ, в любой форме в Интернете, пожалуйста, немедленно сообщите нам адрес местонахождения или название веб-сайта, чтобы мы могли обратиться за помощью. Пожалуйста, свяжитесь с нами по адресу <copyright@packtpub.com> со ссылкой на предполагаемый пиратский материал.

Мы ценим вашу помощь в защите наших авторов и нашу способность предоставить вам ценный контент.

Вопросы

Вы можете связаться с нами по адресу <questions@packtpub.com> если у вас возникли проблемы с каким-либо аспектом книги, и мы сделаем все возможное, чтобы решить их.

Глава 1. Введение в OpenVPN

Интернет в современном обществе так же широко распространен как и любое коммунальное сооружение. Когда кто-то покупает дом или переезжает в новую квартиру, или бизнес переезжает в новое место, интернет является первой услугой в списке, за которой следует электричество, тепло, мусор и, возможно (но маловероятно) проводная телефонная линия. Вы можете даже возразить, что современный квалифиликатор даже не нужен. С помощью таких программ, как One Laptop per Child, в сочетании с усилиями таких компаний как Facebook и Google, так называемые страны третьего мира, где нет водопроводов, канализации или даже телефонной связи, имеют Интернет.

Когда у вас есть развитая служба с большим количеством людей настанет момент, когда необходимо будет обеспечить безопасность и защиту данных, передаваемых по этой сети. В большинстве толп и больших скоплений людей есть более гнусный элемент, стремящийся воспользоваться теми, у кого меньше знаний. **Виртуальные частные сети (VPN)** были созданы из-за большой потребности в защищенной связи через незащищенную инфраструктуру. Первоначальная крупномасштабная сеть ARPANET имела очень низкую (если вообще имела) защиту и аутентификацию, а все остальные узлы были изначально доверенными. Сетевые ландшафты сегодня очень разные, и даже многие случайные, нетехнические пользователи осознают отсутствие безопасности своих соединений.

Правительственные учреждения уже давно стали мишениями для разведки. На протяжении тысячелетий методы и процедуры медленно совершенствовались и настраивались для защиты конфиденциальной информации от врагов и других любопытных глаз. Первоначально запечатанные воском письма, которые носили доверенные лица, означали что вы и получатель можете быть уверены в том, что сообщение прибыло безопасно и беспрепятственно. С течением времени и развитием технологий стало легче перехватывать эти сообщения, читать или изменять их и отправлять далее по пути.

Вторая мировая война дала развитие многим величайшим достижениям в криптографии и защищенной связи. От таких устройств, как немецкая машина "Энигма" до Кодовых говорунов Навахо, надежная связь между войсками и командованием была бесконечной гонкой вооружений. Сегодня правительства и военные - не единственные группы, стремящиеся к уединению. Корпорации хотят поддерживать целостность данных и защиту стандартов **индустрии платежных карт (PCI)** для защиты потребителей. Члены семьи хотят обсуждать семейные дела по частным каналам, где их никто не сможет подслушивать. Другие хотят прорваться через национальные брандмауэры, предназначенные для наблюдения за населением и ограничения контента, считающегося спорным или противоречащим политике партии.

Каждый день большинство людей используют VPN или имеют возможность использовать VPN независимо от того, осознают они это или нет. Существует множество различных технологий VPN как от коммерческих поставщиков, так и в виде проектов с открытым исходным кодом. Одним из самых популярных программных проектов для VPN с открытым исходным кодом является OpenVPN. Цель этой книги - сделать вас мастером OpenVPN; вы узнаете не только технологию, лежащую в ее основе, но и рассуждения, логику и логистику всего, что связано с этим. В то время как эта книга будет упоминать и касаться коммерческого предложения от OpenVPN Technologies Inc. - Access Server, основной акцент будет сделан на open source/community версию OpenVPN.

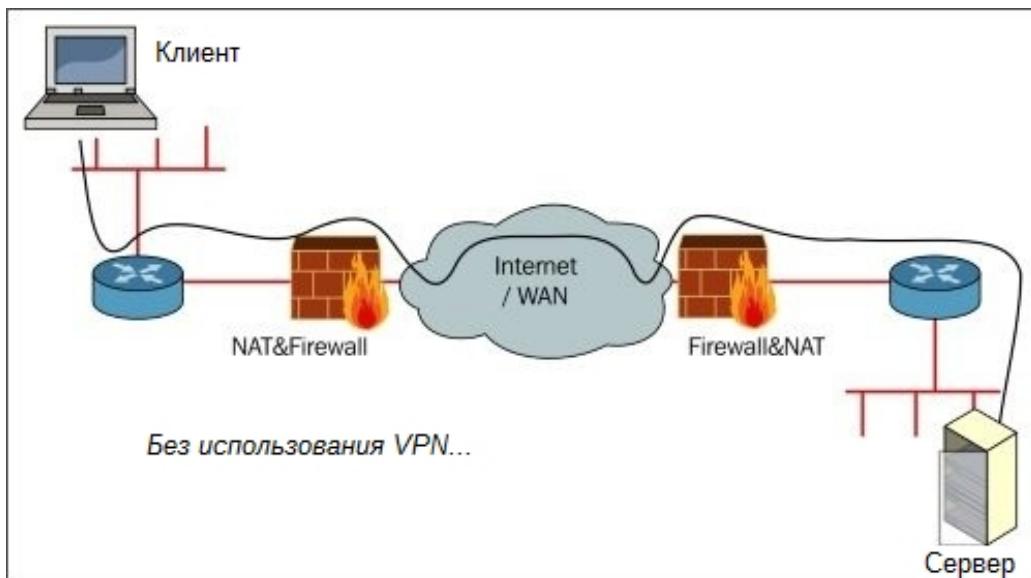
Что такое VPN?

Проще говоря, VPN позволяет администратору создавать "локальную" сеть между несколькими компьютерами в разных сегментах сети. В некоторых случаях эти машины могут находиться в

одной и той же локальной сети, они могут быть удалены друг от друга через общедоступный Интернет или даже могут быть подключены через множество соединительных сред, таких как беспроводные восходящие каналы, спутниковая связь, коммутируемая сеть, и так далее. Р в VPN происходит от дополнительной защиты (private), чтобы сделать эту виртуальную сеть приватной. Сетевой трафик, проходящий через VPN, часто называют *внутри туннеля* (VPN) по сравнению со всем другим трафиком *за пределами туннеля*.

На следующем рисунке показан сетевой трафик, который традиционно проходит через несколько сегментов сети и общий Интернет. Здесь этот трафик относительно открыт для проверки и анализа. Хотя защищенные протоколы, такие как HTTPS и SSH, менее уязвимы - их все же можно идентифицировать; если злоумышленник отслеживает сетевой трафик - он все еще может видеть какой тип соединения установлен, с какого компьютера к какому серверу.

Когда используется VPN - трафик *внутри туннеля* больше не может быть идентифицирован.



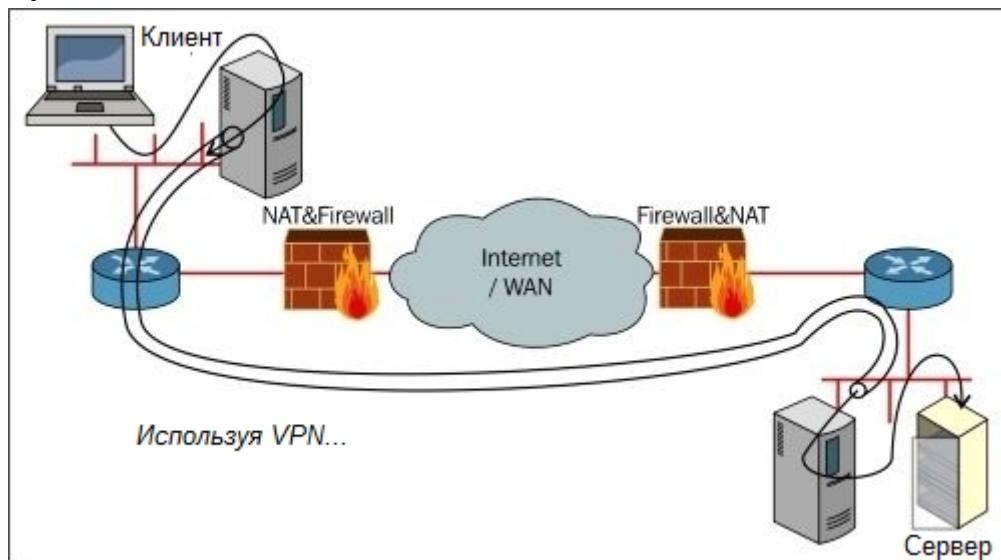
Трафик внутри VPN может быть любым, какой бы вы не отправили через локальную или глобальную сеть: веб-трафик, электронная почта, текст, графика и т.д. Примеры некоторых приложений включают следующее:

- Банкоматы: банкоматы могут использовать VPN для более безопасного подключения к банковским системам.
- Открытый/бесплатный Wi-Fi: С распространением бесплатных или публичных беспроводных сетей обычные пользователи могут использовать VPN для защиты своего интернет-серфинга.
- Корпоративные сети: корпорации и другие организации могут использовать VPN для подключения нескольких офисов или даже целых центров обработки данных.
- Сервисы GeoIP / Геолокации: некоторые веб-сайты предоставляют данные, основанные на географическом местоположении, используя базы данных GeoIP и другие записи. VPN может позволить вам "прыгать" через другую машину в месте, ближе к контенту, который вы действительно хотите. Интернет-videосервисы, такие как Hulu, YouTube и Netflix, являются типичными примерами этого.
- Обход цензуры / политической свободы: некоторые режимы, такие как в Северной Корее или Китае, имеют чрезвычайно ограничительные правила цензуры. "Великий брандмауэр Китая" - один из крайних примеров. Запрет доступа к Интернету во время политических восстаний, таких как "Арабская весна", пытается сдерживать и

контролировать сообщения вне конфликта. Виртуальные частные сети могут помочь выйти за пределы этих ограничительных правил в более широкий Интернет.

Вот пример трафика внутри VPN. В то время как сама VPN маршрутизируется через Интернет, как на предыдущем рисунке, устройства по сетевому пути видят только трафик VPN; эти устройства совершенно не знают о том, что передается внутри приватного туннеля.

Защищенные протоколы, такие как HTTPS и SSH, по-прежнему будут защищены внутри туннеля от других пользователей VPN, но будут дополнительно неопознаваемы извне туннеля. VPN не только шифрует трафик внутри, но и скрывает и защищает отдельные потоки данных от потоков вне туннеля.



Следует отметить, что на предыдущем рисунке показаны как сильные стороны, так и одна из величайших угроз технологии VPN. VPN-туннель пустили через маршрутизаторы и межсетевые экраны с обеих сторон. Таким образом, весь сетевой трафик, проходящий через VPN-туннель, обходит обычную сетевую защиту, если не приняты специальные меры для контроля VPN-трафика.

В большинстве реализаций VPN используется некоторая форма шифрования и, кроме того, аутентификация. Шифрование VPN гарантирует что другие стороны, которые могут отслеживать трафик между системами, не смогут декодировать и дополнительно анализировать конфиденциальные данные. Аутентификация состоит из двух компонентов, каждый в своем контексте.

Во-первых, существует аутентификация пользователя или системы, обеспечивающая авторизацию тех, кто подключается к сервису. Этот тип аутентификации может быть в форме сертификатов для каждого пользователя или комбинации имени пользователя и пароля. Кроме того, могут быть согласованы правила, специфичные для данного пользователя, такие как конкретные маршруты, правила брандмауэра или другие сценарии и утилиты. Как правило, они уникальны для каждого экземпляра, хотя даже это можно настроить (когда используется OpenVPN, см. --duplicate-cn).

Вторым компонентом аутентификации является дополнительная защита потока связи. В этом случае устанавливается метод подписи каждого отправленного пакета. Каждая система проверяет, что полученные VPN-пакеты правильно подписаны, прежде чем расшифровывать полезную нагрузку. За счет аутентификации пакетов, которые уже зашифрованы, система может сэкономить время обработки, даже не расшифровывая пакеты, несоответствующие правилам аутентификации. В конце концов, это серьезно мешает потенциальной атаке **Отказ в обслуживании (Denial of Service - DoS)**, а также срыве **Атаки посредника (Man in the middle - MITM)**, предполагая что ключи подписи хранятся в безопасности!

Типы VPN

Есть множество продуктов VPN, доступных на рынке, как коммерческих, так и с открытым исходным кодом. Почти всех их можно разделить на следующие четыре категории:

- протокол PPTP на основе VPN
- протокол IPSec на основе VPN
- SSL на основе VPN
- OpenVPN

Некоторые люди утверждают что OpenVPN - это также VPN на основе SSL, поскольку он использует протокол SSL или TLS для установления безопасного соединения. Тем не менее, мы сделали отдельную категорию для OpenVPN, так как она отличается от любого другого SSL-решения VPN.

Теперь мы рассмотрим более подробно каждый из четырех типов VPN:

PPTP

Одним из самых старых протоколов VPN является **Туннельный протокол типа точка-точка (Point-to-Point Tunneling Protocol - PPTP)**, разработанный Microsoft и Ascend в 1999 году. Он официально зарегистрирован как RFC2637 (полный стандарт см.

<https://www.ietf.org/rfc/rfc2637.txt>). Клиент PPTP был включен в Windows с 1995 года и до сих пор входит в большинство операционных систем.

В настоящее время протокол PPTP считается принципиально небезопасным, так как степень безопасности соединения напрямую связана с силой выбранного механизма аутентификации (например, пароля). Таким образом, небезопасный пароль приводит к небезопасному VPN-соединению. Большинство настроек PPTP используют протокол MS-CHAPv2 для шифрования паролей и именно этот протокол существенно небезопасен. Безопасность протокола PPTP, включая расширения Microsoft MS-CHAPv2, обсуждалась в статье, доступной по адресу <https://www.schneier.com/paper-pptpv2.html>.

Также можно использовать сертификаты X.509 для защиты соединения PPTP, что приводит к довольно безопасному соединению. Однако не все клиенты PPTP поддерживают EAP-TLS, что необходимо для использования сертификатов X.509.

PPTP использует два канала: канал управления для настройки соединения и другой канал для передачи данных. Канал управления инициируется через TCP-порт 1723. Канал данных использует протокол **General Routing Encapsulation (GRE)**, который является IP-протоколом 47. Для сравнения, "обычный" трафик TCP/IP передается с использованием IP-протокола 6 (TCP) и 17 (UDP).

Клиенты PPTP доступны практически во всех операционных системах от Windows до Linux и производных Unix, для устройств iOS и Android.

IPSec

Стандарт IPSec является официальным стандартом IEEE/IETF для защиты IP. Он официально зарегистрирован как RFC2411 (полный стандарт см. <https://www.ietf.org/rfc/rfc2411.txt>). IPSec также встроен в стандарт IPv6.

IPSec работает на уровне 2 и 3 сетевого стека модели OSI. Он вводит концепцию политик безопасности, что делает его чрезвычайно гибким и мощным, а также чрезвычайно сложным в настройке и устранении неполадок. Политики безопасности позволяют администратору

шифровать трафик между двумя конечными точками на основе многих параметров, таких как IP-адрес источника и назначения, а также TCP и UDP-порты источника и назначения.

IPSec может быть настроен на использование предустановленных общих ключей или сертификатов X.509 для защиты VPN-подключения. Кроме того, для аутентификации VPN-подключения используются сертификаты X.509, одноразовые пароли или протоколы имени_пользователя/пароля.

В IPSec есть два режима работы: туннельный и транспортный режим. Транспортный режим чаще всего используется в сочетании с **протоколом туннелирования 2 уровня (Layer 2 Tunneling Protocol - L2TP)**. Этот протокол L2TP выполняет аутентификацию пользователя, как описано в предыдущем разделе. Клиенты IPSec, встроенные в большинство операционных систем, обычно выполняют IPSec + L2TP, хотя также возможно установить соединение только для IPSec. VPN-клиент IPSec, встроенный в Microsoft Windows, по умолчанию использует IPSec + L2TP, но его можно отключить или обойти. Тем не менее он включает в себя команды шифрования и изменения политики безопасности.

Как и PPTP, IPSec также использует два канала: канал управления для настройки соединения и канал для передачи данных. Канал управления инициируется через UDP-порт 500 или 4500. Канал данных использует протокол **Encapsulated Security Payload (ESP)**, который является IP-протоколом 50. Для сравнения, «обычный» трафик TCP/IP передается с использованием IP-протокола 6 (TCP) и 17 (UDP). Целостность пакетов IPSec обеспечивается с помощью **кода аутентификации сообщений на основе хэша (Hash-based Message Authentication Code - HMAC)**, являющийся тем же методом, что используется в OpenVPN.

Одним из основных недостатков IPSec является внедрение многими производителями оборудования собственных расширений в стандарт, что затрудняет (если не делает невозможным) подключение двух конечных точек IPSec от разных производителей.

Программное обеспечение IPSec включено практически во все операционные системы, а также в микропрограммы межсетевого экрана, маршрутизатора и коммутатора.

VPN на основе SSL

В настоящее время, наиболее часто используемый VPN - это VPN на основе SSL, основанный на протоколе SSL/TLS. VPN на основе SSL часто называют VPN без клиентского доступа или VPN на основе Интернета, хотя есть некоторые поставщики, предоставляющие отдельное клиентское программное обеспечение, например Cisco AnyConnect и Microsoft SSTP.

Большинство VPN на основе SSL используют тот же сетевой протокол, что используется для безопасности веб-сайтов (HTTPS), тогда как OpenVPN использует собственный формат для шифрования и подписи трафика данных. Это основная причина почему OpenVPN указан как отдельная категория VPN.

Не существует четко определенного стандарта для VPN на основе SSL, но большинство используют протокол SSL/TLS для настройки и защиты соединения. В большинстве случаев соединение защищено с помощью сертификатов X.509 с одноразовым паролем или протоколами имени_пользователя/пароля для аутентификации соединения. VPN на основе SSL очень похожи на соединения, используемые для защиты веб-сайтов (HTTPS) и часто используется один и тот же протокол и канал (TCP и порт 443).

Несмотря на то, что VPN на основе SSL часто называют веб-интерфейсом или бесклиентским, существует довольно много производителей, использующих *плагин для браузера* или элемент управления ActiveX для “улучшения” VPN-соединения. Это делает VPN несовместимым с неподдерживаемыми браузерами или операционными системами.

OpenVPN

OpenVPN часто называют VPN на основе SSL, так как он использует протокол SSL/TLS для защиты соединения. Однако OpenVPN также использует HMAC в сочетании с алгоритмом дайджеста (или хеширования) для обеспечения целостности доставляемых пакетов. Он может быть настроен на использование предустановленных ключей, а также сертификатов X.509. Эти функции обычно не предлагаются другими VPN на основе SSL.

Кроме того, OpenVPN использует виртуальный сетевой адаптер (устройство tun или tap) в качестве интерфейса между программным обеспечением OpenVPN пользовательского уровня и операционной системой. В общем, любая операционная система, поддерживающая устройства tun/tap, может запускать OpenVPN. В настоящее время это ОС на основе Linux, Free/Open/NetBSD, Solaris, AIX, Windows и Mac OS, а также устройства iOS/Android. Для всех этих платформ необходимо установить клиентское программное обеспечение, которое отличает OpenVPN от клиентских или веб-сетей VPN.

Протокол OpenVPN не определен в стандарте RFC, но протокол общедоступен, поскольку OpenVPN является частью программного обеспечения с открытым исходным кодом. Тот факт что это открытый исходный код, на самом деле делает OpenVPN более безопасным, чем VPN с закрытым исходным кодом, так как код постоянно проверяется разными людьми. Кроме того, очень мало шансов что в OpenVPN будут встроены секретные бэкдоры.

OpenVPN имеет понятие канала управления и канала данных, зашифрованных и защищенных по-разному. Однако весь трафик проходит через одно UDP или TCP-соединение. Канал управления зашифрован и защищен с использованием SSL/TLS, канал данных зашифрован с использованием специального протокола шифрования.

Протокол и порт по умолчанию для OpenVPN - это UDP и порт 1194. Прежде чем IANA предоставила OpenVPN официальное назначение порта, старые клиенты (2.0-beta16 и старше) по умолчанию использовали порт 5000.

Сравнение VPN

Каждая из различных технологий VPN имеет свои особенности, преимущества и недостатки. Несмотря на то, что эта книга посвящена OpenVPN, существуют случаи, когда, например, VPN на основе IPSec подходит больше, в зависимости от требований пользователей.

Преимущества и недостатки PPTP

Основным преимуществом VPN на основе PPTP является встроенность программного обеспечения VPN-клиента в большинство операционных систем. Кроме того, время запуска для настройки и инициализации PPTP VPN-соединения довольно мало.

Недостатками VPN на основе PPTP являются отсутствие безопасности и параметров конфигурации как на стороне клиента, так и на стороне сервера. Кроме того, расширение EAP-TLS, позволяющее использовать сертификаты X.509, полностью поддерживается только в Microsoft Windows, хотя существует патч для пакета rpppd с открытым исходным кодом для включения поддержки EAP-TLS. Пакет rpppd входит почти в каждый дистрибутив Linux. Кроме того, если нужно использовать EAP-TLS, то простота настройки PPTP VPN значительно уменьшается. Это связано с тем, что EAP-TLS требует настройки инфраструктуры открытого ключа, как IPSec и OpenVPN.

Другим существенным недостатком PPTP является использование протокола GRE, который плохо интегрируется с устройствами NAT.

Преимущества и недостатки IPSec

Преимуществами протокола IPSec являются его высокая безопасность, хорошая поддержка от различных производителей и платформ, включая маршрутизаторы xDSL и Wi-Fi, а также возможность использовать детализированные политики безопасности для управления потоком трафика.

Недостатки IPSec заключаются в том, что его общеизвестно сложно настраивать и устранять неисправности, разные реализации IPSec от разных поставщиков оборудования плохо работают вместе, а IPSec плохо интегрируется с сетями с NAT. В частности, не рекомендуется, а иногда даже невозможно, запустить сервер IPSec, находящийся в сети с NAT.

Преимущества и недостатки VPN на основе SSL

Преимущество VPN на основе SSL или веб-VPN заключается в том, что клиентское программное обеспечение не задействовано или почти не используется. Это делает установку и инициализацию на стороне клиента очень простой.

Недостаток веб-VPN заключается в том, что она часто не является *полноценной* VPN и обеспечивает доступ к одному серверу или набору серверов. Также сложнее обмениваться локальными данными с удаленными точками или сервером.

Преимущества и недостатки OpenVPN

Преимуществами OpenVPN являются простота развертывания, его конфигурируемость и возможность развертывания OpenVPN в сетях с ограниченным доступом, включая сети с поддержкой NAT. Кроме того, OpenVPN включает функции безопасности, которые столь же сильны, как и решения на основе IPSec, в том числе безопасность аппаратного токена и поддержка различных механизмов аутентификации пользователей.

Недостатками OpenVPN являются отсутствие масштабируемости и зависимость от установки клиентского программного обеспечения. Еще одним недостатком является отсутствие графического интерфейса для настройки и управления. В частности, драйвер интерфейса tap для Microsoft Windows часто вызывал проблемы развертывания при выпуске новой версии Windows.

История OpenVPN

OpenVPN был первоначально написан Джеймсом Йонаном с первоначальным выпуском - версией 0.90 в 2001 году под лицензией GPL. Первый выпуск позволял пользователям создавать простые VPN типа «точка-точка» по UDP с использованием шифрования Blowfish и, дополнительно, подписи SHA1 HMAC. С версией 1.0, были добавлены аутентификация на основе TLS, обмен ключами, а так же страница man.

Улучшения для OpenVPN 1.x включали улучшенную поддержку TLS, защиту от повторов и перенос на другие операционные системы. Некоторые порты были включены для OpenBSD, Mac OS и улучшены пакеты для RedHat. До версии 1.1.1 устройство tun должно было настраиваться вручную вне OpenVPN. В этом выпуске добавлена опция --ifconfig, автоматически настраивающая устройство tun, значительно упрощая общую настройку.

Серия 1.x была относительно сырой по сравнению с текущей версией OpenVPN 2.3.8, как и следовало ожидать от нового проекта. Одним из основных препятствий была интеграция OpenSSL. Поскольку OpenSSL был известен своей плохой или полностью отсутствующей документацией - разработчик должен был перейти непосредственно к исходному коду, чтобы интегрировать проект с OpenVPN. Так же на ранних этапах требовались изменения лицензии, чтобы позволить более специальному общедоступному лицензионному коду GNU связываться с библиотекой OpenSSL не-GPL. Эти проблемы были проработаны и в журнале изменений на

протяжении серии 1.x были добавлены новые функции.

Некоторые заметные обновления в серии 1.x включают в себя:

- 2001.05.13 (0.90): это был первый выпуск
- 2002.03.23 (1.0): позволил TLS-аутентификацию и обмен ключами
- 2002.04.09 (1.1.0): появился порт OpenBSD и соединение OpenSSL
- 2002.04.22 (1.1.1): появилась опция `--ifconfig`
- 2002.05.22 (1.2.0): здесь появились файлы конфигурации (вместо просто параметров командной строки, поддержка `pthread` и порт Solaris)
- 2002.07.10 (1.3.0): улучшена поддержка FreeBSD и улучшена регистрация
- 2002.10.23 (1.3.2): начальная поддержка IPv6 и больше улучшений для FreeBSD
- 2003.05.07 (1.4.0): включены функции MTU
- 2003.07.24 (1.5-beta1): поддержка TCP
- 2003.11.03 (1.5-beta13): в нем появилась поддержка параметров конфигурации `--http-proxy`, `--redirect-gateway` и `--crl-verify`
- 2004.02.01 (1.6-beta5): прокси SOCKS5 и IPv6 на FreeBSD
- 2004.05.09 (1.6.0): это финальная версия 1.x

OpenVPN 2.0 видел большие успехи от выпусков 1.x. В версии 2.0 были предприняты усилия для обеспечения многоклиентских экземпляров сервера, улучшенной работы с потоками и улучшенного tun/tap адаптера Windows. Разработка для 2.0 пересекалась с 1.x более года, с начальными тестовыми выпусками для 2.0, датируемыми ноябрем 2003 года и финальной версией 1.x не выходившей до 9 мая 2004 года. Пока она была окончательно выпущена, 2.0 увидела 29 тестовых выпусков, 20 бета-релизов и 21 релиз-кандидат за полтора года усилий (с ноября 2003 года по апрель 2005 года).

Некоторые ключевые особенности релиза 2.0 по сравнению с 1.6.0 следующие:

- Позволяет серверу принимать соединения от нескольких клиентов
- Включает опцию `config` на стороне сервера `push` для клиентов (`--push/-pull`)
- Позволяет аутентификацию по имени_пользователя/паролю
- Поддерживает `chroot` и понижение привилегий демона (`--user/-group/-chroot`)
- Поддерживает сценарии подключения клиента
- Имеет интерфейс управления
- Появление Easy-RSA

Разработка с 2.0 до 2.0.9 в основном состояла из исправлений ошибок и исправлений для нескольких уязвимостей безопасности. Помимо некоторых случайных вкладов от сторонних разработчиков, OpenVPN разрабатывался Джеймсом до выпуска 2.1. 2.0.9 оставался неизменным официальным выпуском с октября 2006 года до версии 2.1.0 в декабре 2009 года.

OpenVPN 2.1 был первым крупным выпуском с заметным количеством кода, написанного кем-то кроме Джеймса Йонана. Алон Бар-Лев внес значительный вклад, начиная с 2.1-beta3 со многими исправлениями для поддержки криптографии и поправками. Рассматривая первый

реальный выпуск сообщества, 2.1 увидел большую работу в базовом ядре кода, включая интерфейс управления и сетевую адресацию. Некоторые заметные примечания к выпуску включают следующее:

- 2005.11.12 (2.1-beta7): файлы ca, cert, key и dh могут быть указаны в файле конфигурации.
- 2006.01.03 (2.1-beta8): добавлена подсеть --topology.
- 2006.02.16 (2.1-beta9): было разрешено совместное использование портов, чтобы OpenVPN и HTTPS могли совместно использовать порт.
- 2008.09.10 (2.1_rc10): предупреждает если используется общая подсеть 192.168.0.0/24 или 192.168.1.0/24. --server-bridge был добавлен для поддержки DHCP-прокси.
- 2010.08.09 (2.1.2): у него была система сборки Windows на основе Python с улучшенной обработкой AUTH_FAIL для интерфейса управления.
- 2010.11.09 (2.1.4): это был последний выпуск серии 2.1.

В августе 2008 года официального релиза с 2.0.9 не было. Кроме того, было мало поддержки со стороны сообщества, кроме списка рассылки. Был интерес к созданию сообщества и Кризе Кинг и Эрик Крист подталкивали к созданию сообщества вокруг проекта. Изначально все усилия были направлены на поддержку пользователей.

Поскольку группа людей, поддерживающих OpenVPN, росла - это привлекало людей, которые могли писать хороший код. Был установлен контакт с OpenVPN Inc. с целью не только обеспечить более высокий уровень поддержки OpenVPN, но также создать и расширить программное обеспечение, написанное Джеймсом, но попытки сотрудничества были отвергнуты.

Начались переговоры по **Internet Relay Chat (IRC)**, являющимся предпочтаемым разработчиками средством коммуникации, для переноса проекта с целью добиться прогресса. Разработка началась; некоторые участники управляли IRC и помогали в списках рассылки. Другие - создали репозиторий, вики и веб-форум. Среднее использование было примерно 2 сообщения в день на форуме и около 8 пользователей в IRC.

В начале 2009 года OpenVPN Technologies наняли Самули Сеппанена чтобы помочь создать сообщество с открытым исходным кодом и взаимодействовать с ним. Самули способствовал установлению прочных отношений между корпорацией, энтузиастами и волонтерами. Было построено сильное сообщество вокруг проекта. Сегодня на форуме в среднем 16 сообщений в день (всего более 35 000 сообщений), а IRC колеблется от 150 до 250 пользователей в любой день.

OpenVPN 2.2 был первым выпуском после перехода к более ориентированной на сообщество модели разработки. После выяснения модели развития и направления, сообщество решило двигаться с проектом и сразу же началась работа.

Первоначально для OpenVPN 2.2 Джеймс по-прежнему полностью контролировал то, что будет объединено с основным исходным деревом, так как дерево все еще управлялось с использованием подверсий у OpenVPN Technologies. Позже дерево исходных текстов было перенесено в GIT, а роли поменялись местами, где изменения Джеймса были приняты и объединены в дерево проектов с открытым исходным кодом.

Заметные изменения в OpenVPN 2.2:

- Аутентификация открытым текстом для SOCKS
- Улучшена поддержка платформы для подсети --topology

- Режим tap для Solaris
- Сборка Windows скомпилирована с включением ENABLE_PASSWORD_SAVE
- Поддержка Windows IPv6 tun
- Клиентские сертификаты могут быть опущены с поведением, аналогичным веб-браузеру (--client-cert-not-required)
- Клиентские сертификаты теперь могут указывать отдельное имя пользователя вместо использования общего имени сертификата (- -x509-username-field)
- Была удалена поддержка для Windows 2000 и более ранних выпусков
- 2011.04.26 была выпущена версия 2.2.0
- 2011.07.06 версия 2.2.1 была выпущена с небольшими изменениями, в основном связанными со сборкой/установкой
- 2011.12.22 версия 2.2.2 была выпущена с изменениями tap-драйвера Windows

OpenVPN 2.3 - начало серьезного поворота в структуре сборки OpenVPN. Вкратце, конечная цель - создать более расширяемый и удобный источник для плагинов. Поскольку сборка для мобильных платформ, таких как Android и iOS, уже требует переписывания с нуля, Джеймс и другие разработчики почистили старый код в пользу более компактных и нормализованных функций. Эти переписывания сделаны на C++, в отличие от используемого языка С.

Хотя она и упомянута в журнале изменений предыдущих версий, поддержка IPv6, как полезной нагрузки, так и транзита в OpenVPN, действительно не достигла зрелости до выпуска 2.3. Подавляющее большинство вкладов в поддержку IPv6 было результатом тяжелой работы Герт Деринг.

Еще одной важной особенностью выпуска 2.3 было добавление поддержки PolarSSL. PolarSSL - это альтернативная криптографическая библиотека для OpenSSL и теперь OpenVPN может быть создан на основе любой библиотеки. Эта тема более подробно обсуждается далее в этой главе.

Список улучшений и дополнений для выпуска 2.3 огромен, но основные моменты заключаются в следующем (полный журнал изменений находится по адресу

<https://community.openvpn.net/openvpn/wiki/ChangesInOpenvpn23>):

- Кроссплатформенная поддержка IPv6 (транзит и полезная нагрузка)
- API нового плагина
- Поддержка создания PolarSSL и подготовка других потенциальных альтернатив
- Теперь клиенты могут информировать сервер о поддержке LZO и сервер может автоматически отключить LZO для этого клиента
- Обходной путь для локальных конфликтов маршрутизации (--client-nat)
- Новый режим каталога --crl-verify, файлы с одинаковыми именами отключают сертификаты, как если бы они были отозваны
- Поддержка сертификатами UTF-8 для полей сертификата
- Разделение проекта по различным подпроектам:
 - Основной проект OpenVPN
 - tap-Windows

- Easy-RSA
 - Система сборки OpenVPN
- Завершение клиентских соединений из интерфейса управления

Версия 2.3.8 была самой последней версией на момент написания.

Пакеты OpenVPN

В Интернете доступно несколько пакетов OpenVPN:

- Версия OpenVPN с открытым исходным кодом или версия сообщества
- OpenVPN Access Server - коммерческое предложение с закрытым исходным кодом от OpenVPN Inc.
- Версии OpenVPN для мобильных платформ Android и iOS (часть кода закрыта по требованию Apple)

Версия с открытым исходным кодом (сообщество)

Версии OpenVPN с открытым исходным кодом становятся доступными после публикации каждого выпуска. Сообщество располагает ресурсами для создания бинарных пакетов для нескольких платформ, включая как 32-х, так и 64-разрядных клиентов Windows. Доступные в настоящее время варианты загрузки находятся по адресу <http://openvpn.net/index.php/download/community-downloads.html>.

Некоторые хранилища пакетов операционных систем отслеживают разработку и делают доступными выпуски моментальных снимков. FreeBSD, например, имеет порт security/openvpn-devel, отслеживающий еженедельные снимки тарболла из разработки OpenVPN. Если вы хотите запустить последнюю и самую передовую версию OpenVPN - сначала посмотрите на хранилище ваших пакетов. В противном случае - вы всегда можете собрать напрямую из исходников.

Версия OpenVPN от сообщества может выступать как в качестве VPN-сервера, так и в качестве VPN-клиента. Отдельной - только клиентской версии не существует.

Закрытые исходники (коммерческий) Access Server

OpenVPN Technologies Inc. предлагает коммерческую версию OpenVPN под названием Access Server. По сравнению с проектом с открытым исходным кодом, Access Server предлагает множество функций и вариантов развертывания, которые могут понравиться некоторым организациям. Access Server является платным продуктом, но на сайте доступна пробная версия с двумя лицензионными ключами.

Пакеты программ, виртуальные устройства и облачные сервисы, все доступны от OpenVPN Technologies Inc. на <https://openvpn.net/index.php/access-server/overview.html>.

OpenVPN Access Server включает в себя собственный клиент OpenVPN - OpenVPN Connect для Windows и Mac OS. Это клиентское программное обеспечение обычно работает только с OpenVPN Access Server. Также можно использовать версию сообщества в качестве клиента для OpenVPN Access Server.

Мобильная платформа (смешанная) OpenVPN/OpenVPN Connect

Для мобильных устройств, таких как iPhone/iPad и Android, OpenVPN Technologies Inc. предоставляет специальный OpenVPN Connect Client. OpenVPN Technologies Inc. и Джеймс специально приложили немало усилий и юридических споров с такими компаниями как Google

и Apple, чтобы получить доступ к используемому VPN API на каждой платформе.

Из-за особенностей NDA Apple, в настоящее время исходные коды OpenVPN Connect недоступны и не могут быть открыты для общего доступа. Клиент iOS OpenVPN Connection можно загрузить из Apple App Store по адресу <https://itunes.apple.com/us/app/openvpn-connect/id590379981?mt=8>.

Есть Android-клиенты, написанные несколькими разработчиками, но официально поддерживается только версия OpenVPN for Android, написанная Арне Швабе, которую можно найти по адресу <https://play.google.com/store/apps/details?id=de.blinkt.openvpn&hl=ru>.

OpenVPN Connect, написанный OpenVPN Technologies Inc. также доступен. Вы можете загрузить клиент Android OpenVPN Connect по адресу <https://play.google.com/store/apps/details?id=net.openvpn.openvpn&hl=ru>.

Одним из серьезных преимуществ OpenVPN Connect является то, что он поддерживает/поддерживается как общедоступной версией OpenVPN, так и OpenVPN Access Server с закрытым исходным кодом. Если вам нужен доступ к обоим типам серверов - рекомендуется OpenVPN Connect.

Другие платформы

Некоторые производители оборудования пытаются интегрировать поддержку OpenVPN в свои устройства. Некоторые предлагают версии прошивки для VoIP-телефонов, включающие более старую версию OpenVPN. Другие проекты микропрограмм, такие как DD-WRT для маршрутизаторов Linksys, а также проекты FreeNAS, pfSense и другие также интегрируют OpenVPN.

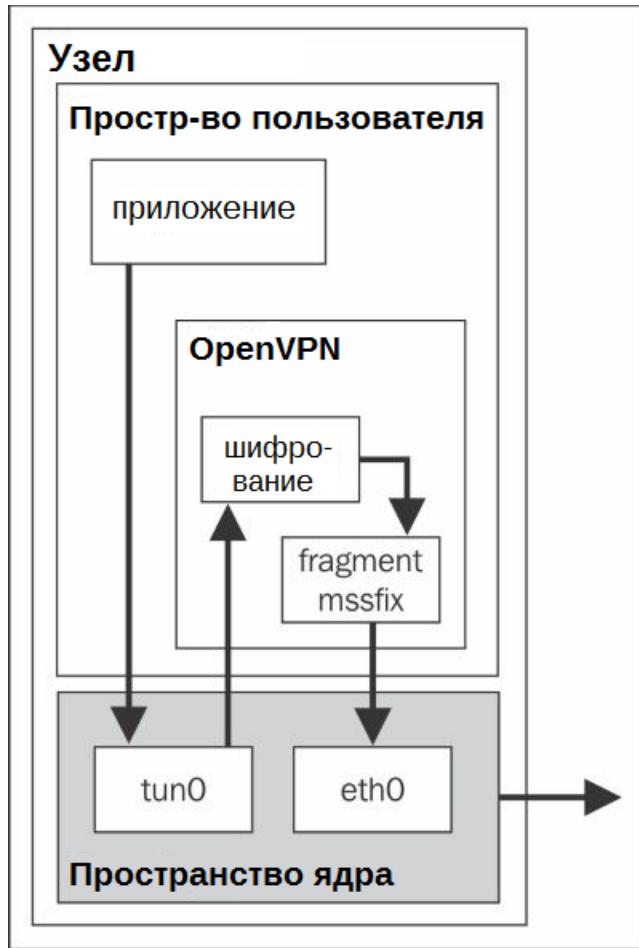
Внутренности OpenVPN

Конструкция OpenVPN не документирована, но большинство внутренних возможностей OpenVPN можно обнаружить взглянув на исходный код.

Драйвер tun/tap

Одним из основных составных блоков OpenVPN является драйвер tun/tap. Концепция драйвера tun/tap происходит из мира Unix/Linux, где он часто доступен как часть операционной системы. Это виртуальный сетевой адаптер, обрабатываемый операционной системой как двухточечный адаптер (*в режиме tun*) для трафика только по IP или как полноценный виртуальный адаптер Ethernet для всех типов трафика (*в режиме tap*). В основе этого адаптера находится приложение, такое как OpenVPN, для обработки входящего и исходящего трафика. Linux, Free/Open/NetBSD, Solaris и Mac OS включают в себя драйвер ядра tun, который может работать как в режиме tun, так и в режиме tap. Недавно аналогичный драйвер был добавлен в AIX - производную Unix от IBM.

Для Microsoft Windows Джеймс Йонан написал специальный драйвер NDIS, называемый адаптером TAP-WIN32. На данный момент доступны версии драйверов NDIS5 и NDIS6, поддерживающие Windows XP через Windows 8.1. Разработка этого адаптера теперь официально отделена от основной разработки OpenVPN, но OpenVPN по-прежнему сильно зависит от него.



Поток трафика из пользовательского приложения через OpenVPN изображен на предыдущей диаграмме. На схеме приложение отправляет трафик на адрес, доступный через туннель OpenVPN. Шаги следующие:

1. Приложение передает пакет операционной системе.
2. ОС решает, используя *обычные* правила маршрутизации, что пакет должен маршрутизоваться через VPN.
3. Затем пакет пересыпается на устройство ядра - tun.
4. Устройство ядра tun пересыпает пакет в процесс OpenVPN (в пользовательском пространстве).
5. Процесс OpenVPN шифрует и подписывает пакет, фрагментирует его при необходимости, а затем снова передает его ядру, чтобы отправить на адрес удаленной конечной точки VPN.
6. Ядро забирает зашифрованный пакет и перенаправляет его на удаленную конечную точку VPN, где происходит обратный процесс.

На этой диаграмме также видно, что производительность OpenVPN всегда будет ниже, чем у обычного сетевого подключения. Для большинства приложений потеря производительности минимальна и/или приемлема. Однако для скоростей, превышающих 1 Гбит/с, существует узкое место в производительности как с точки зрения пропускной способности, так и с точки зрения задержки.

Следует отметить, что производительность драйвера Windows намного ниже, чем производительность встроенных адаптеров tun/tap в других операционных системах. Это верно даже для самой последней реализации драйвера TAP-Win32 в NDIS6. Для одного клиента OpenVPN влияние довольно мало, но для крупномасштабного сервера OpenVPN,

обслуживающего множество клиентов, легко может вызвать проблемы с производительностью. Это одна из основных причин того, почему сообщество разработчиков исходного кода обычно рекомендует использовать хост на основе Unix или Linux в качестве сервера OpenVPN.

Режимы UDP и TCP

OpenVPN в настоящее время поддерживает два способа обмена данными между конечными точками: используя UDP-пакеты или используя TCP-пакеты. UDP - это протокол без установления соединения или с потерями; если пакет отбрасывается при передаче, то сетевой стек не может прозрачно исправить это. TCP-пакеты - это протокол, ориентированный на соединение; пакеты отправляются и доставляются с использованием протокола квитирования, обеспечивая доставку каждого пакета на другую сторону.

Оба способа общения имеют свои преимущества и недостатки. На самом деле это зависит от типа трафика, отправляемого через VPN-туннель, для определения более подходящего режима связи. Использование приложения на основе TCP через VPN на основе TCP может привести к двойной потере производительности, особенно если основное сетевое соединение не работает. В этом случае повторная передача потерянных пакетов выполняется для пакетов, потерянных как внутри, так и вне туннеля, что приводит к удвоению нагрузки. Это хорошо объясняется в статье «*Почему TCP через TCP - плохая идея*» на <http://sites.inka.de/~W1011/devel/tcp-tcp.html>.

Однако аналогичным образом можно утверждать, что отправка UDP через UDP также не является хорошей идеей. Если приложение, использующее UDP для своего трафика, подвержено атакам потери сообщений или переупорядочивания пакетов, то основное зашифрованное TCP-соединение повысит безопасность таких приложений даже в большей степени, чем базовая VPN на основе UDP. Если большая часть трафика через VPN основана на UDP, то иногда лучше использовать TCP-соединение между конечными точками VPN.

При выборе между транспортным протоколом UDP или TCP общее практическое правило следующее: *если UDP (режим `udp`) работает для вас - используйте его; если нет, то попробуйте TCP (режим `tcp-server` и режим `tcp-client`)*. Некоторые коммутаторы и маршрутизаторы неправильно перенаправляют трафик UDP, что может быть проблемой, особенно если к одному коммутатору или маршрутизатору подключено несколько клиентов OpenVPN. Точно так же на производительность OpenVPN через TCP может сильно повлиять выбор **интернет-провайдеров (ISP)**: некоторые интернет-провайдеры используют нечетные размеры MTU или правила фрагментации пакетов, что приводит к крайне низкой производительности OpenVPN через TCP по сравнению с незашифрованным TCP-трафиком.

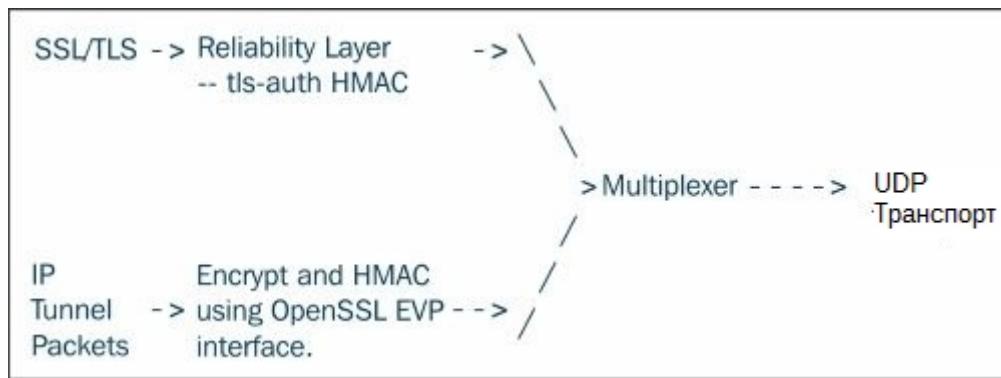
Протокол шифрования

Было сказано, что OpenVPN реализует TLS через UDP. Это более или менее верно, но то, как OpenVPN использует TLS, отличается от того, как его использует веб-браузер. Таким образом, когда OpenVPN запускается по протоколу TCP (использование порта 443 является распространенным методом для защиты межсетевых экранов), трафик можно отличить от обычного трафика TLS. Брандмауэр, использующий **Deep Packet Inspection (DPI)**, может легко отфильтровать трафик OpenVPN.

Основное различие между OpenVPN-TLS и браузер-TLS заключается в способе подписи пакетов. OpenVPN предлагает функции для защиты от DoS-атак, подписывая пакеты канала управления с помощью специального статического ключа (`-t ls-auth ta.key 0|1`). Пакеты канала данных, отправляемые через одно и то же соединение UDP или TCP, подписываются совершенно по-разному и очень легко отличаются от трафика HTTPS. На сайте OpenVPN (<http://openvpn.net>) показано, как зашифрованы пакеты для транспорта UDP, что показано ниже.

Тот же механизм используется для транспорта TCP

(<http://openvpn.net/index.php/open-source/documentation/security-overview.html>).



Это также основная причина, по которой совместное использование портов, при использовании OpenVPN и защищенным веб-сервером одного и того же IP-адреса и порта, может работать.

Каналы управления и передачи данных

OpenVPN использует два виртуальных канала для связи между клиентом и сервером:

- Канал управления TLS для обмена информацией о конфигурации и шифрованием между клиентом и сервером. Этот канал используется в основном при запуске VPN-подключения, а также для обмена новыми материалами ключей шифрования. Этот материал обновляется через определенный период (на основе параметров `--reneg-sec`, `--reneg-bytes` или `--reneg-pkts`).
- Канал данных, по которому осуществляется обмен зашифрованной полезной нагрузкой.

Исключением является старый двухточечный режим с общим ключом, в котором используется только канал данных.

Шифрование и аутентификация (подпись) для канала управления и канала данных определяются по-разному. Канал управления инициируется с использованием протокола TLS, аналогично тому, как инициируется безопасное подключение к веб-сайту. Во время инициализации канала управления, шифрование и алгоритм хеширования согласовываются между клиентом и сервером.

Алгоритмы шифрования и аутентификации для канала данных не подлежат обсуждению, но они устанавливаются в файлах конфигурации клиента и сервера для OpenVPN. Текущими настройками по умолчанию являются Blowfish в качестве алгоритма шифрования и SHA1 в качестве алгоритма хеширования. Возможность согласования алгоритмов шифрования и хеширования для канала данных занимает важное место в списке пожеланий группы разработчиков, но требует значительных изменений в коде.

Алгоритмы шифрования и хеширования

OpenVPN поддерживает широкий спектр алгоритмов шифрования и хеширования. Они используются для шифрования полезной нагрузки, а функция HMAC использует алгоритм дайджеста или хеширования для аутентификации входящих пакетов. Поскольку OpenVPN использует канал управления и канал данных - существует два набора алгоритмов шифрования и хеширования, которые можно настроить.

Алгоритмы шифрования канала управления и хеширования обычно согласовываются при запуске. Список доступных комбинаций шифрования и хеширования можно отобразить с помощью следующей команды:

```
$ openvpn --show-tls
```

Доступные варианты шифрования TLS перечислены в порядке предпочтения:

```
TLS-ECDHE-RSA-WITH-AES-256-GCM-SHA384
TLS-ECDHE-ECDSA-WITH-AES-256-GCM-SHA384
TLS-ECDHE-RSA-WITH-AES-256-CBC-SHA384
TLS-ECDHE-ECDSA-WITH-AES-256-CBC-SHA384
TLS-ECDHE-RSA-WITH-AES-256-CBC-SHA
TLS-ECDHE-ECDSA-WITH-AES-256-CBC-SHA
TLS-DHE-DSS-WITH-AES-256-GCM-SHA384
TLS-DHE-RSA-WITH-AES-256-GCM-SHA384
TLS-DHE-RSA-WITH-AES-256-CBC-SHA256
TLS-DHE-DSS-WITH-AES-256-CBC-SHA256
TLS-DHE-RSA-WITH-AES-256-CBC-SHA
TLS-DHE-DSS-WITH-AES-256-CBC-SHA
TLS-DHE-RSA-WITH-CAMELLIA-256-CBC-SHA
TLS-DHE-DSS-WITH-CAMELLIA-256-CBC-SHA
TLS-ECDH-RSA-WITH-AES-256-GCM-SHA384
[...]
```

Этот вывод был получен на хосте CentOS 6 с использованием библиотеки OpenSSL 1.0.1e.

Доступные комбинации во многом зависят от конкретной версии используемой библиотеки SSL. Вы можете указать список `tls-ciphers` в файле конфигурации OpenVPN способом, очень похожим на настройку модуля Apache `mod_ssl`:

```
tls-cipher TLS-ECDHE-RSA-WITH-AES-256-GCM-SHA384:TLS-ECDHE-ECDSAWITH-AES-
256-CBC-SHA384
:TLS-ECDH-RSA-WITH-AES-256-GCM-SHA384
```

Перечислите все шифры в одну строку; предыдущий вывод был изменен для удобства чтения.

Для канала данных алгоритмы шифрования и хеширования управляются с помощью параметров `--cipher` и `--auth`. Если алгоритмы шифрования и аутентификации не указаны, то используются значения по умолчанию `bf-cbc` и `sha1`, соответственно.

Чтобы получить список доступных алгоритмов шифрования используйте следующую команду:

```
$ openvpn --show-ciphers
```

Следующие алгоритмы и режимы шифрования доступны для использования с OpenVPN. Каждый показанный здесь шифр может использоваться в качестве параметра опции `--cipher`. Размер ключа по умолчанию отображается независимо от того, может ли он быть изменен с помощью директивы `--keysize`. Рекомендуется использовать режим CBC. В режиме статического ключа допускается только режим CBC:

```
[...]
BF-CBC 128 bit default key (variable)
BF-CFB 128 bit default key (variable) (TLS client/server mode)
BF-OFB 128 bit default key (variable) (TLS client/server mode)
[...]
AES-128-CBC 128 bit default key (fixed)
AES-128-OFB 128 bit default key (fixed) (TLS client/server mode)
AES-128-CFB 128 bit default key (fixed) (TLS client/server mode)
AES-192-CBC 192 bit default key (fixed)
AES-192-OFB 192 bit default key (fixed) (TLS client/server mode)
AES-192-CFB 192 bit default key (fixed) (TLS client/server mode)
AES-256-CBC 256 bit default key (fixed)
AES-256-OFB 256 bit default key (fixed) (TLS client/server mode)
AES-256-CFB 256 bit default key (fixed) (TLS client/server mode)
[...]
```

В этом выводе показаны только наиболее часто используемые шифры. Список доступных шифров снова зависит от точной версии базовой библиотеки шифрования. Однако в

большинстве случаев должны быть доступны шифры Blowfish (BF-*) и AES (AES-*).

Точно так же для алгоритмов аутентификации (HMAC-подписи) мы используем следующую команду, чтобы перечислить все доступные параметры:

```
$ openvpn --show-digests
```

Следующие дайджесты сообщений доступны для использования с OpenVPN. Дайджест сообщения используется вместе с функцией HMAC для аутентификации полученных пакетов. Вы можете указать дайджест сообщения в качестве параметра опции `--auth`:

```
[...]
SHA 160 bit digest size
SHA1 160 bit digest size
[...]
ecdsa-with-SHA1 160 bit digest size
[...]
SHA256 256 bit digest size
SHA384 384 bit digest size
SHA512 512 bit digest size
SHA224 224 bit digest size
```

В этом выводе показаны только наиболее часто используемые дайджесты или алгоритмы хеширования. Список доступных дайджестов зависит от точной версии базовой библиотеки шифрования. В большинстве случаев должны быть доступны алгоритмы хэширования семейства SHA-1 и SHA-2.

OpenSSL против PolarSSL

Начиная с OpenVPN 2.3, добавлена поддержка новой библиотеки SSL. Библиотека PolarSSL (<http://polarssl.org>) может быть скомпилирована вместо библиотеки OpenSSL по умолчанию. Основная причина добавления второй библиотеки состояла в том, чтобы обеспечить независимость лежащих в основе библиотек шифрования и гарантировать, что никаких проблем с авторским правом не возникнет, поскольку лицензия на авторские права OpenSSL отличается от той, которую использует OpenVPN.

Резюме

В этой главе мы начали с объяснения что такое VPN. Затем обсудили некоторые примеры различных типов протоколов VPN, включая PPTP, IPsec и OpenVPN. После краткого обзора истории OpenVPN мы приступили к более глубокому погружению в методы, используемые в OpenVPN. Эти методы включают адаптер tun/tap и используемые алгоритмы шифрования и подписывания пакетов.

После этого знакомства с VPN и самим OpenVPN пришло время узнать больше об OpenVPN. В следующей главе мы начнем с самого простого метода использования OpenVPN - режима «точка-точка» с использованием предустановленных общих ключей. По мере продвижения по этой книге вы получите более глубокие знания о том, как использовать OpenVPN в самых разных конфигурациях.

Глава 2. Режим точка-точка

Сначала режим «точка-точка» с использованием предустановленных ключей был единственным доступным вариантом при использовании OpenVPN. В настоящее время существует несколько способов использования OpenVPN, но режим «точка-точка» все еще имеет свое применение. Термин *двуточечный режим с использованием предварительно установленных общих ключей* часто сокращается до **предустановленных ключей**.

В режиме «точка-точка» OpenVPN настраивается с использованием предустановленных общих секретных ключей для предварительно определенных конечных точек, и только одна конечная точка может одновременно подключаться к экземпляру сервера. Термин *сервер* можно считать вводящим в заблуждение, поскольку обе конечные точки более или менее равны, когда речь заходит о функциональности. Конечная точка, инициирующая соединение, считается клиентом, а другая - рассматривается как сервер.

Мы начнем с демонстрации очень простого примера. После этого обсудим больше возможностей, предоставляемых OpenVPN. Мы рассмотрим следующие темы:

- Протокол TCP и разные порты
- Режим TAP
- Секретные ключи OpenVPN
- Маршрутизация
- Полная настройка, включая IPv6
- Настройка без IP
- Трехсторонняя маршрутизация
- Мостовые адAPTERЫ TAP на обоих концах
- Совмещение режима "точка-точка" с сертификатами

Плюсы и минусы ключевого режима

Основным вариантом использования режима предустановленного общего ключа является подключение двух удаленных сетей, например, главного офиса и удаленного для небольшой компании. Как только требуется более трех пользователей или конечных точек - гораздо проще использовать режим клиент/сервер, как описано в [Главе 4](#), *Режим клиент/сервер с туннельными устройствами*. Пример того, как соединить три местоположения вместе с помощью общих ключей, приведен ниже в этой главе и станет понятно - почему режим с предустановленным ключом не подходит для трех сторон или пользователей.

Основные преимущества использования режима предустановленного ключа:

- Его очень легко настроить
- Нет необходимости в **инфраструктуре открытого ключа (public key infrastructure - PKI)** или сертификатах X.509
- Может работать на ограниченном оборудовании, таком как коммутаторы или маршрутизаторы на основе Linux

Недостатками использования режима предустановленного ключа являются:

- Как указывает имя *точка-точка* - только две конечные точки могут пользоваться одним соединением. Поэтому этот режим плохо масштабируется.
- Некоторые оболочки GUI для OpenVPN (например, GNOME NetworkManager) не поддерживают предустановленные ключи. Тоже самое относится к клиентам Android и iOS.
- Секретный ключ должен быть скопирован в удаленную конечную точку с использованием безопасного канала, например с использованием SSH. Иногда это может быть угрозой безопасности.
- Невозможно зашифровать секретный ключ с помощью парольной фразы, как это возможно при использовании публичных/приватных ключей X.509.
- Он считается несколько менее безопасным, поскольку безопасность полностью зависит от безопасности и надежности предустановленного секретного ключа. Кроме того, в этом режиме нет **идеальной секретности пересылки (perfect forwarding secrecy - PFS)**. Без PFS злоумышленник может записать весь зашифрованный трафик VPN. Если злоумышленнику удастся в какой-то момент нарушить шифрование - тогда *весь* записанный трафик VPN может быть дешифрован. С PFS невозможно расшифровать *старые* данные.

Важно понимать, что OpenVPN на самом деле работает по-разному при использовании предустановленных ключей по сравнению с использованием сертификатов и настройкой клиент/сервер. Пути кода, которые следуют в OpenVPN, на самом деле сильно отличаются, например, согласование канала управления не требуется. Среднестатистический конечный пользователь не увидит эти различия, но их важно знать, когда необходимо устраниТЬ неполадки при соединении OpenVPN. Кроме того, при чтении файла журнала OpenVPN с высокой детальностью вывода (т.е. что-либо выше 5), выходные данные подключения с предустановленным ключом будут выглядеть совершенно иначе по сравнению с выходными данными подключения на основе сертификатов.

Если не указано иное - все примеры в этой главе основаны на конечных точках под управлением CentOS 6 64bit. Версия установленного программного обеспечения OpenVPN v2.3.2 взята из репозитория CentOS-EPEL.

Первый пример

Давайте посмотрим на наш самый первый пример:

1. Самый простой и краткий пример подключения двух компьютеров с использованием OpenVPN - это запуск первой конечной точки в режиме прослушивания с использованием фиксированных IP-адресов и сети в режиме tun:

```
[root@server] # openvpn \
    --ifconfig 10.200.0.1 10.200.0.2 \
    --dev tun
```

- Затем запустите клиент OpenVPN:

```
[root@client] # openvpn \
    --ifconfig 10.200.0.2 10.200.0.1 \
    --dev tun \
    --remote openvpnserver.example.com
```

- В другом окне терминала отобразите сетевое устройство:

```
[root@client] # ip addr show tun0
7: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500
    qdisc pfifo_fast state UNKNOWN group default
        qlen 100 link/none

        inet 10.200.0.2 peer 10.200.0.1/32 scope global tun0
            valid_lft forever preferred_lft forever
```

На следующих снимках экрана показано как устанавливается соединение:

```
root@centos6-kvm:movpn# openvpn --ifconfig 10.200.0.2 10.200.0.1 --dev tun --remote openvpnserver.example.com
Mon Sep  8 18:27:29 2014 OpenVPN 2.3.2 x86_64-redhat-linux-gnu [SSL (OpenSSL)] [LZO] [EPOLL] [PKCS11] [eurephia]
[MH] [IPv6] built on Sep 12 2013
Mon Sep  8 18:27:29 2014 ***** WARNING *****: all encryption and authentication features disabled -- all data
will be tunnelled as cleartext
Mon Sep  8 18:27:29 2014 TUN/TAP device tun0 opened
Mon Sep  8 18:27:29 2014 do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
Mon Sep  8 18:27:29 2014 /sbin/ip link set dev tun0 up mtu 1500
Mon Sep  8 18:27:29 2014 /sbin/ip addr add dev tun0 local 10.200.0.2 peer 10.200.0.1
Mon Sep  8 18:27:29 2014 UDPv4 link local (bound): [undef]
Mon Sep  8 18:27:29 2014 UDPv4 link remote: [AF_INET]192.168.122.1:1194
Mon Sep  8 18:27:39 2014 Peer Connection Initiated with [AF_INET]192.168.122.1:1194
Mon Sep  8 18:27:40 2014 Initialization Sequence Completed
```

- Теперь мы можем пропинговать конечные точки OpenVPN с любого конца, при условии что правила брандмауэра и SELinux позволяют это.

Журнал подключения показывает некоторые интересные детали. Версия OpenVPN на стороне клиента: 2.3.2 x86_64-redhat-linux-gnu. Это подтверждает, что мы запускаем v2.3.2 на 64-битной производной RedHat Linux.

Журнал подключения показывает предупреждение:

```
Mon Sep  8 18:27:29 2014 ***** WARNING *****: all encryption and
authentication features disabled -- all data will be tunnelled as
cleartext
```

Это предупреждение выводится, поскольку не был указан секретный ключ для шифрования соединения, что делает этот пример небезопасным.

- Устройство Linux tun0 открыто для подключения. Мы указали --dev tun, что сообщает OpenVPN открыть первый доступный адаптер tun. Если теперь запустить второе соединение OpenVPN - следующий экземпляр будет использовать tun1.
- Команда Linux iproute2 /sbin/ip используется для настройки сетевого адаптера tun0. Указанный IP-адрес назначается вместе с **максимальной единицей передачи по умолчанию (maximum transfer unit - MTU)** 1500 байтов.
- По умолчанию OpenVPN будет использовать UDP-порт 1194 для установления соединения. Если требуется протокол TCP, то аргументы командной строки на обоих концах немного различаются (это показано в следующем разделе).
- Из временных меток, напечатанных в начале каждой строки, видно, что для установления начального соединения требуется 10 секунд.

Если выдается следующее сообщение, то соединение было успешно установлено. Однако в следующих примерах мы увидим, что это необязательно означает что VPN работает нормально.

```
Mon Sep  8 18:27:40 2014 Initialization Sequence Completed
```

Протокол TCP и разные порты

Протоколом по умолчанию, используемым OpenVPN, является UDP, так как обычно он больше подходит для VPN-подключений. Однако, если требуется протокол TCP - предыдущий пример необходимо изменить лишь незначительно: на конце прослушивания запустите экземпляр сервера OpenVPN:

```
[root@server] # openvpn \
--ifconfig 10.200.0.1 10.200.0.2 \
--dev tun \
--proto tcp-server
```

На стороне клиента команда выглядит следующим образом:

```
[root@client] # openvpn \
--ifconfig 10.200.0.2 10.200.0.1 \
--dev tun \
--proto tcp-client \
--remote openvpnserver.example.com
```

OpenVPN теперь будет подключаться через TCP-порт 1194. Также возможно изменить номер порта, используя параметр `--port`, например `--port 5000`.

Режим TAP

Если через VPN-туннель необходимо передавать трафик не-TCP/IP (например устаревший трафик AppleTalk или IPX), то требуется *tap*-устройство. *tap*-устройство позволяет интерфейсу передавать полные кадры Ethernet через VPN-туннель. Затраты при прохождении полных кадров Ethernet незначительны. Назначение IP для устройства *tap* отличается от устройства *tun*, поскольку устройство *tap* действует как обычный сетевой адаптер, которому необходимо назначить один IP-адрес и сетевую маску.

Предыдущий пример теперь изменен. На конце прослушивания запустите процесс сервера OpenVPN:

```
[root@server] # openvpn \
--ifconfig 10.200.0.1 255.255.255.0 \
--dev tap
```

На стороне клиента команда выглядит следующим образом:

```
[root@client] # openvpn \
--ifconfig 10.200.0.2 255.255.255.0 \
--dev tap \
--remote openvpnserver.example.com
```

Опять же, мы приводим конфигурацию сетевого устройства:

```
[root@client] # ip addr show tap0
8: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
    qdisc pfifo_fast state UNKNOWN group default qlen 100
        link/ether 6e:ea:0e:47:a3:d8 brd ff:ff:ff:ff:ff:ff
        inet 10.200.0.2/24 brd 10.200.0.255 scope global tap0
            valid_lft forever preferred_lft forever
        inet6 fe80::6cea:eff:fe47:a3d8/64 scope link
            valid_lft forever preferred_lft forever
```

Сравните её с конфигурацией из самого первого примера.

Топология subnet

OpenVPN 2.1 и более поздние версии поддерживают новую топологию - топологию *subnet* для назначения IP-адресов в сетях в режиме *tun*, которая очень похожа на IP-адреса, используемые в сетях в режиме *tap*. При использовании `--topology subnet` одному интерфейсу туннеля

назначаются один IP-адрес и маска сети, а адрес партнера не указывается.

Хотя использовать этот режим топологии для выделенной двухточечной связи не имеет большого смысла - этот параметр можно использовать чтобы сделать настройку режима точка-точка в режиме tun практически такой же, как соответствующая настройка tap. Чтобы использовать этот режим топологии, используйте настройку, описанную ниже.

Начните на конце прослушивания:

```
[root@server] # openvpn \
--ifconfig 10.200.0.1 255.255.255.0 \
--dev tun \
--topology subnet
```

На стороне клиента команда выглядит следующим образом:

```
[root@client] # openvpn \
--ifconfig 10.200.0.2 255.255.255.0 \
--dev tun \
--topology subnet \
--remote openvpnserver.example.com
```

Строка --ifconfig теперь такая же, как для примера tap. Единственное изменение - добавление --topology subnet на обоих концах.

Туннель открытого текста

Предыдущий пример не использует шифрование или ключи аутентификации; следовательно, вы получаете следующее предупреждение:

```
Mon Sep 8 18:27:29 2014 ***** WARNING *****: all encryption and
authentication features disabled -- all data will be tunnelled as cleartext
```

Тем не менее, туннель открытого текста имеет свои применения. В доверенной среде, где безопасность обрабатывается на другом уровне (например, с использованием выделенного оптоволоконного кабеля), туннель с открытым текстом обеспечивает лучшую производительность по зашифрованному туннелю и легче отслеживается поток трафика по туннелю.

Кроме того, если вы заранее знаете что весь трафик, который будет проходить через туннель, зашифрован сам (например, весь трафик строго HTTPS), то можно использовать туннель открытого текста для избежания двойного шифрования, что может привести к снижению производительности. Особенно при работе OpenVPN на небольшом или встроенном оборудовании (например, Raspberry Pi или даже на некоторых платах Arduino) шифрование наносит большой удар по производительности.

Туннель открытого текста может быть настроен с использованием примеров, приведенных в предыдущем разделе. Если секретный ключ не указан - шифрование и аутентификация (подпись HMAC) автоматически отключаются. Также возможно отключить их явно:

На конце прослушивания запустите:

```
[root@server] # openvpn \
--ifconfig 10.200.0.1 10.200.0.2 \
--dev tun \
--cipher none --auth none
```

На стороне клиента код выглядит следующим образом:

```
[root@client] # openvpn \
--ifconfig 10.200.0.2 10.200.0.1 \
--dev tun \
--cipher none --auth none \
--remote openvpnserver.example.com
```

После того, как соединение установлено, мы можем проверить что содержимое действительно отправлено в виде открытого текста, используя команду `tcpdump` (или эквивалентную, например, Wireshark):

1. Запустите соединение.
2. Запустите `tcpdump` и прослушивайте *обычный* сетевой интерфейс, а не туннельный и отфильтруйте пакеты OpenVPN (UDP port 1194) с помощью следующей команды:

```
[root@server] # tcpdump -l -w - eth0 udp port 1194 | strings
```

3. Теперь отправьте текст через туннель, например, `nc` (netcat):

- На стороне сервера:

```
$ nc -l -p 31000
```

- На стороне клиента:

```
$ nc 10.200.0.1 31000
hello from openvpn client
goodbye
```

4. Вывод `tcpdump` теперь должен показать что-то вроде этого:

```
tcpdump: listening on eth0, link-type EN10MB (Ethernet),
capture size 65535 bytes
V~hello from openvpn client
5goodbye
```

Символы, показанные в виде текстовых сообщений, являются артефактами инкапсуляции пакетов OpenVPN.

Секретные ключи OpenVPN

Для защиты соединения OpenVPN необходим секретный ключ. Сначала мы сгенерируем такой ключ. Затем его необходимо скопировать на удаленную конечную точку с использованием безопасного канала (например, SCP):

```
$ openvpn --genkey --secret secret.key
```

Обратите внимание, что нет необходимости запускать эту команду от пользователя root (отсюда и приглашение \$). Полученный файл секретного ключа имеет следующий формат:

```
##
2048 bit OpenVPN static key
#-
-----BEGIN OpenVPN Static key V1-----
1393ae687606c1f7d465d70227bf63e8
8963e9d1401450002d073d6eab1bffd
b06d1a33cc5c45d4a667016339e921d3
3ac36b1a949eb52e9217e41e4b035a7b
987ddfa9d6766d3b5e4c952dc27f518d
12ccff6b2f0966284382ddc0f62b824a
f576f0982beec9d6a4728d0788499a75
0fd7055ef681404fd463d9862d3a40a9
```

```
31fca7d87997c70c07b8303a1b85f1ff
76aa7790e7c341353d2b4ea5049b11a2
51346e7dd39fc1f1e53ae57c46cf60c8
24db00a871262fee78050a9df6a57322
0bb0d980b6cf1be90a2f304f99fb9cde
7cdf72d20e7dee555c7c99950aa4d8e6
86a020c3a63125fb99d56181ff4ca20c
d6711eab15a4d6faf706f2601eb6 61b7
-----END OpenVPN Static key V1-----
```

После публикации ключа здесь - это уже не секрет.

Команда `openvpn --genkey` генерирует 2048-битный ключ или 256 байт случайных данных. Эти 256 байтов перечислены в шестнадцатеричном формате в файле `secret.key`, но в настоящее время используются не все 256 байт (как мы увидим позже).

Секретный ключ используется OpenVPN для шифрования и аутентификации (подписи) каждого пакета. Алгоритм шифрования по умолчанию - Blowfish (BF-CBC), а алгоритм HMAC по умолчанию - SHA1. Алгоритм Blowfish использует 128-битное шифрование, тогда как ключ, используемый для алгоритма SHA1, составляет 160 бит.

Если OpenVPN запускается с увеличенным выходом отладки (`--verb 7` или выше), используемые ключи печатаются при запуске:

На конце прослушивания (сервере) запустите демон OpenVPN:

```
[root@server] # openvpn \
--ifconfig 10.200.0.1 10.200.0.2 \
--dev tun \
--secret secret.key \
--verb 7
```

На стороне клиента команда выглядит следующим образом:

```
[root@client] # openvpn \
--ifconfig 10.200.0.2 10.200.0.1 \
--dev tun \
--secret secret.key \
--remote openvpnserver.example.com
```

Вывод журнала на стороне сервера будет содержать строки вида:

```
Static Encrypt: Cipher 'BF-CBC' initialized with 128 bit key
Static Encrypt: CIPHER KEY: 1393ae68 7606c1f7 d465d702 27bf63e8
Static Encrypt: CIPHER block_size=8 iv_size=8
Static Encrypt: Using 160 bit message hash 'SHA1' for
HMAC authentication
Static Encrypt: HMAC KEY: 987ddfa9 d6766d3b 5e4c952d c27f518d
12ccff6b
Static Encrypt: HMAC size=20 block_size=20
Static Decrypt: Cipher 'BF-CBC' initialized with 128 bit key
Static Decrypt: CIPHER KEY: 1393ae68 7606c1f7 d465d702 27bf63e8
Static Decrypt: CIPHER block_size=8 iv_size=8
Static Decrypt: Using 160 bit message hash 'SHA1' for
HMAC authentication
Static Decrypt: HMAC KEY: 987ddfa9 d6766d3b 5e4c952d c27f518d
12ccff6b
Static Decrypt: HMAC size=20 block_size=20
```

Ключ шифра BF-CBC - это 1393 ae68 7606 c1f7 d465 D702 27bf 63e8, что точно соответствует первой строке файла секретного ключа OpenVPN.

SHA1 ключом HMAC является 987d dfa9 d676 6d3b 5e4c 952d c27f 518d 12cc ff6b,

который также можно найти в файле секретного ключа, начиная с пятой строки.

Обратите внимание, что одни и те же ключи используются для шифрования и дешифрования данных, а также аутентификации данных. В следующем разделе мы увидим, как можно использовать разные ключи для шифрования, дешифрования и аутентификации.

Использование нескольких ключей

OpenVPN поддерживает использование *направленных* ключей, то есть различные ключи используются для входящих и исходящих данных. Это еще больше повышает безопасность. Добавив флаг *направления* к параметру `--secret` мы можем указать, что будут использоваться различные ключи. Флаг *направления* должен быть установлен в 0 на одном конце и в 1 на другом:

На прослушивающем конце (сервере), запустите:

```
[root@server] # openvpn \
    --ifconfig 10.200.0.1 10.200.0.2 \
    --dev tun \
    --secret secret.key 0 \
    --verb 7
```

На стороне клиента код выглядит следующим образом:

```
[root@client] # openvpn \
    --ifconfig 10.200.0.2 10.200.0.1 \
    --dev tun \
    --secret secret.key 1 \
    --remote openvpnserver.example.com \
    --verb 7
```

Вывод журнала на стороне сервера теперь будет содержать строки вида:

```
Static Encrypt: CIPHER KEY: 1393ae68 7606c1f7 d465d702 27bf63e8
Static Encrypt: HMAC KEY: 987ddfa9 d6766d3b 5e4c952d c27f518d
                12ccff6b

Static Decrypt: CIPHER KEY: 31fca7d8 7997c70c 07b8303a 1b85f1ff
Static Decrypt: HMAC KEY: 0bb0d980 b6cf1be9 0a2f304f 99fb9cde
                7cdf72d2
```

Ключи шифрования CIPHER и HMAC теперь явно отличаются от ключей дешифрования CIPHER и HMAC. Кроме того, каждый из этих ключей можно найти в файле OpenVPN `secret.key`:

- Ключ шифрования CIPHER KEY начинается с 1 строки длиной 128 бит или 16 байт
- Ключ шифрования HMAC KEY начинается с 5 строки длиной 160 бит или 20 байт
- Ключ дешифрования CIPHER KEY начинается с 9 строки длиной 128 бит или 16 байт
- Ключ дешифрования HMAC KEY начинается с 13 строки длиной 160 бит или 20 байт

Кроме того, вывод журнала на стороне клиента показывает что ключи поменялись местами:

```
Static Encrypt: CIPHER KEY: 31fca7d8 7997c70c 07b8303a 1b85f1ff
Static Encrypt: HMAC KEY: 0bb0d980 b6cf1be9 0a2f304f 99fb9cde
                7cdf72d2
Static Decrypt: CIPHER KEY: 1393ae68 7606c1f7 d465d702 27bf63e8
Static Decrypt: HMAC KEY: 987ddfa9 d6766d3b 5e4c952d c27f518d
                12ccff6b
```

Это необходимо для функционирования VPN-туннеля, так как необходимые ключи на стороне сервера для шифрования данных, необходимы на стороне клиента для дешифрования данных и наоборот.

Использование разных алгоритмов шифрования и аутентификации

OpenVPN поддерживает несколько различных алгоритмов шифрования и аутентификации (подписи HMAC). Размер ключей, используемых в каждом алгоритме шифрования и HMAC, варьируется, причем текущий максимум составляет значение 256 бит для шифров (например, AES256) и 512 бит для ключа HMAC (например, SHA512). Статический ключ OpenVPN имеет длину 2048 бит, что достаточно для 512-битного шифра и 512-битного ключа HMAC.

Если мы укажем AES256 в качестве алгоритма шифрования и SHA512 в качестве алгоритма аутентификации, то увидим что используемые ключи увеличатся в размере:

На конце прослушивания (сервере) запустите:

```
[root@server] # openvpn \
--ifconfig 10.200.0.1 10.200.0.2 \
--dev tun \
--secret secret.key 0 \
--cipher AES256 --auth SHA512 \
--verb 7
```

На стороне клиента код выглядит следующим образом:

```
[root@client] # openvpn \
--ifconfig 10.200.0.2 10.200.0.1 \
--dev tun \
--secret secret.key 1 \
--cipher AES256 --auth SHA512 \
--remote openvpnserver.example.com \
--verb 7
```

Вывод журнала на стороне сервера теперь содержит следующие строки:

```
Static Encrypt: Cipher 'AES-256-CBC' initialized with 256 bit key
Static Encrypt: CIPHER KEY: 1393ae68 7606c1f7 d465d702 27bf63e8
                 8963e9d1 40145000 2d073d6e ab1bffde
Static Encrypt: CIPHER block_size=16 iv_size=16
Static Encrypt: Using 512 bit message hash 'SHA512' for
                 HMAC authentication
Static Encrypt: HMAC KEY: 987ddfa9 d6766d3b 5e4c952d c27f518d
                 12ccff6b 2f096628 4382ddc0 f62b824a
                 f576f098 2beec9d6 a4728d07 88499a75
                 0fd7055e f681404f d463d986 2d3a40a9
Static Encrypt: HMAC size=64 block_size=64
Static Decrypt: Cipher 'AES-256-CBC' initialized with 256 bit key
Static Decrypt: CIPHER KEY: 31fca7d8 7997c70c 07b8303a 1b85f1ff
                 76aa7790 e7c34135 3d2b4ea5 049b11a2
Static Decrypt: CIPHER block_size=16 iv_size=16
Static Decrypt: Using 512 bit message hash 'SHA512' for
                 HMAC authentication
Static Decrypt: HMAC KEY: 0bb0d980 b6cf1be9 0a2f304f 99fb9cde
                 7cdf72d2 0e7dee55 5c7c9995 0aa4d8e6
                 86a020c3 a63125fb 99d56181 ff4ca20c
                 d6711eab 15a4d6fa f706f260 1eb661b7
```

Этот журнал может быть сопоставлен с файлом `secret.key`:

- Ключ шифрования CIPHER KEY теперь соответствует первым 2 строкам файла.
- Ключ шифрования HMAC KEY теперь соответствует строкам 5-8 файла.

Аналогичным образом можно сопоставить ключи дешифрования.

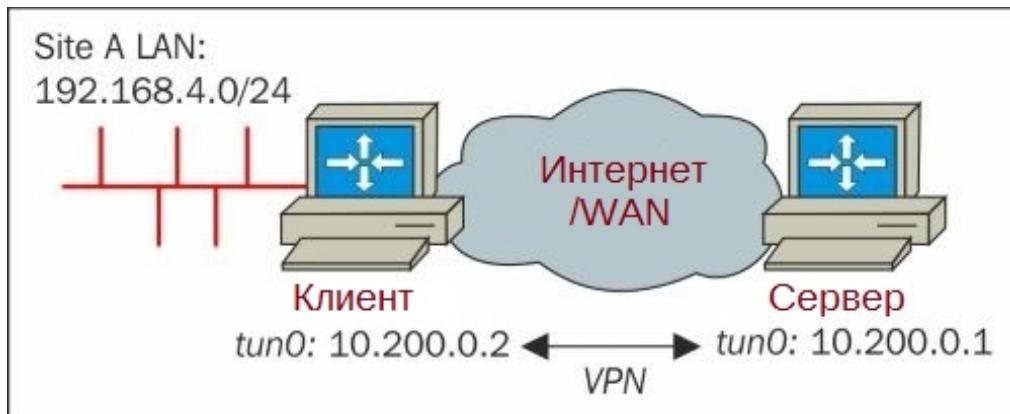
Заметка

VPN-туннель работает так же, как и раньше, но теперь с более надежным шифрованием и аутентификацией на месте. Если в будущем будут введены еще более надежные алгоритмы шифрования или HMAC - формат статического ключа OpenVPN необходимо будет обновить.

Маршрутизация

Как указывалось ранее - основной вариант использования сетей типа «точка-точка» заключается в соединении двух удаленных сетей через безопасный туннель. В предыдущем примере безопасный туннель был установлен, но не были добавлены сетевые маршруты.

В этом примере рассмотрим следующую схему сети:



Клиентскую сеть **192.168.4.0/24** (с маской сети 255.255.255.0) необходимо направить через VPN-туннель к серверу.

На сервере мы запускаем:

```
[root@server] # openvpn \
--ifconfig 10.200.0.1 10.200.0.2 \
--dev tun \
--secret secret.key 0 \
--route 192.168.4.0 255.255.255.0 \
--daemon --log /var/log/movpn-02-server.log
```

На стороне клиента код выглядит следующим образом:

```
[root@client] # openvpn \
--ifconfig 10.200.0.2 10.200.0.1 \
--dev tun \
--secret secret.key 1 \
--remote openvpnserver.example.com \
--daemon --log /var/log/movpn-02-client.log
```

На стороне сервера был добавлен оператор *route*, для сообщения OpenVPN о том, что сеть **192.168.4.0/24** находится на другом конце туннеля. OpenVPN сам по себе мало что с этим сделает, но выдаст соответствующую системную команду */sbin/route* или */sbin/ip route* для настройки таблиц системной маршрутизации. Вместо использования оператора OpenVPN *--route* мы также можем использовать следующую команду:

```
[root@server] # route add -net 192.168.4.0/24 gw 10.200.0.2
```

После того, как VPN-соединение было установлено, мы можем альтернативно использовать команду *iproute2*:

```
[root@server] # ip route add 192.168.4.0/24 via 10.200.0.2
```

Вторая строка операторов, добавленных в этом примере, инструктирует OpenVPN демонизировать себя (то есть запускать в фоновом режиме) и записывать все сообщения в файл

```
/var/log/movpn-02-server.log.
```

Аналогично, тот же оператор `daemon+log` добавляется на стороне клиента.

Заметка

Оператор `--log` перезаписывает файл журнала при каждом запуске OpenVPN. Если вы хотите дописывать в предыдущий файл журнала - используйте следующую команду:

```
--log-append /var/log/movpn-02-server.log
```

На данный момент пример еще не полностью функционален. Если мы пропингуем хост в локальной сети на стороне клиента с VPN-сервера - мы не получим никакого ответа. Это имеет мало общего с самим OpenVPN, а в основном с маршрутизацией TCP/IP. Большинство вопросов, задаваемых в списке рассылки пользователей OpenVPN и интернет-форуме OpenVPN на самом деле являются вопросами маршрутизации.

Причина, по которой от клиентской локальной сети не получен ответ, двояка:

- Переадресация IP или маршрутизация должны быть включены на клиенте OpenVPN. Для каждой операционной системы это достигается по-разному. В Linux обычно достаточно добавить или изменить строку в файле `/etc/sysctl.conf` и перезагрузиться. Должна быть изменена следующая строка:

```
net.ipv4.ip_forward = 1
```

- Можно избежать перезагрузки, выполнив следующую команду:

```
# sysctl -p
```

- Нам также необходимо убедиться в существовании обратного маршрута к серверу OpenVPN в клиентской локальной сети. Это можно сделать добавив маршрут на шлюзе локальной сети или статический маршрут на каждой машине в клиентской локальной сети. В этом примере мы добавляем маршрут на машине Linux, которая подключена к клиентской сети:

```
# route add -net 10.200.0.0/24 gw 192.168.4.100
```

Здесь 192.168.4.100 - это локальный IP-адрес клиента OpenVPN.

Заметка

Маршруты, добавленные таким образом, *не* являются статическими и исчезнут после перезагрузки. Статическое добавление маршрутов зависит от распределения.

Теперь пример работает как и ожидалось. С сервера OpenVPN мы можем пинговать IP-адреса компьютеров в локальной сети на стороне клиента и наоборот:

```
$ ping -c 2 192.168.4.100
PING 192.168.4.100 (192.168.4.100) 56(84) bytes of data.
64 bytes from 192.168.4.100: icmp_seq=1 ttl=64 time=5.97 ms
64 bytes from 192.168.4.100: icmp_seq=2 ttl=64 time=4.22 ms

$ ping -c 2 192.168.4.10
PING 192.168.4.10 (192.168.4.10) 56(84) bytes of data.
64 bytes from 192.168.4.10: icmp_seq=1 ttl=63 time=7.37 ms
64 bytes from 192.168.4.10: icmp_seq=2 ttl=63 time=6.09 ms
```

Конфигурационные файлы и командная строка

Как видно из предыдущего примера: аргументы командной строки для OpenVPN могут быстро стать длинными и сложными. Также возможно (и желательно) использовать файлы

конфигурации для хранения часто используемых опций для OpenVPN. В общем, каждый параметр может быть указан в командной строке с помощью следующей команды:

--<некоторая опция> <аргументы-опции>

Его также можно указать в файле конфигурации с помощью <некоторая опция> <аргументы-опции>, то есть удалить два дефиса перед аргументом командной строки.

Файл конфигурации указывается в командной строке с помощью параметра **--config <путь>**. Почти все параметры, указанные в файле конфигурации, обрабатываются так, как если бы они были указаны в командной строке. Как мы увидим позже в этой книге - в файле конфигурации можно хранить встроенные сертификаты и файлы приватного ключа. Нелегко сделать то же самое используя аргументы командной строки.

Также возможно смешивать файлы конфигурации и аргументы командной строки. Это позволяет легко хранить часто используемые параметры в файле конфигурации, который можно переопределить с помощью аргументов командной строки.

Заметка

Не все параметры конфигурации могут быть переопределены. Некоторые параметры могут быть указаны несколько раз (в частности, **remote <remote-host>**). В этих случаях используется *первое* вхождение.

Командная строка на стороне сервера из предыдущего примера может быть преобразована в следующий файл конфигурации:

```
ifconfig 10.200.0.1 10.200.0.2
dev tun
secret secret.key 0
route 192.168.4.0 255.255.255.0

daemon
log /var/log/movpn-02-server.log
```

Если этот файл конфигурации сохранить как **movpn-02-01-server.conf**, то команда для запуска сервера становится:

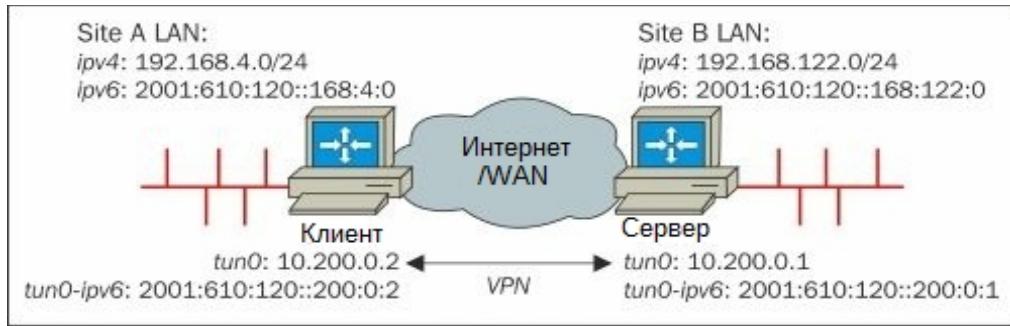
```
[root@server] # openvpn --config movpn-02-01-server.conf
```

Обратите внимание, что порядок аргументов командной строки важен. Все параметры, указанные перед параметром **--config <путь>**, переопределяются параметрами, указанными в файле конфигурации. Все параметры, указанные после параметра **--config**, отменяют параметры в файле конфигурации (с некоторыми исключениями, как отмечалось ранее).

Полная настройка

На основе предыдущих примеров мы теперь можем построить полную конфигурацию производственного уровня, используя файлы конфигурации, включая маршрутизацию, ведение журнала, поддержку IPv6, а также некоторые другие производственные функции, которые предлагает OpenVPN.

Рассмотрим следующую схему сети:



Для сервера мы создаем следующий файл конфигурации `tomvpn-02-02-server.conf`:

```

dev tun
proto udp
local openvpnserver.example.com
lport 1234
remote openvpnclient.example.com
rport 4321

secret secret.key 0
ifconfig 10.200.0.1 10.200.0.2
route 192.168.4.0 255.255.255.0

tun-ipv6
ifconfig-ipv6 2001:610:120::200:0:1 2001:610:120::200:0:2

user nobody
groupnobody # используйте 'group nogroup' для Debian/Ubuntu
persist-tun
persist-key
keepalive 10 60
ping-timer-rem

verb 3
daemon
log-append /var/log/openvpn.log

```

Для клиента мы создаем файл `tomvpn-02-02-client.conf`:

```

dev tun
proto udp
local openvpnclient.example.com
lport 4321
remote openvpnserver.example.com
rport 1234

secret secret.key 1
ifconfig 10.200.0.2 10.200.0.1
route 192.168.122.0 255.255.255.0

tun-ipv6
ifconfig-ipv6 2001:610:120::200:0:2 2001:610:120::200:0:1

user nobody
group nobodygroup nobody # используйте 'group nogroup' для Debian/Ubuntu
persist-tun
persist-key
keepalive 10 60
ping-timer-rem

```

```
verb 3
daemon
log-append /var/log/openvpn.log
```

Файлы конфигурации клиента и сервера очень похожи, за исключением зеркальных адресов и направления зеркальных ключей.

В этих файлах конфигурации появилось несколько новых опций:

- Хотя `proto udp` является протоколом по умолчанию - разумно явно указать его в файле конфигурации во избежание путаницы.
- `local <IP>` - это локальный IPv4-адрес, на котором OpenVPN будет прослушивать входящие соединения. Если этот адрес не указан - OpenVPN будет прослушивать адрес 0.0.0.0, что означает все интерфейсы.
- `lport` - это локальный порт, который будет прослушивать OpenVPN. Значение по умолчанию - 1194, но можно использовать любой допустимый и доступный номер порта.
- `remote <IP>` - это удаленный IPv4-адрес, на который процесс сервера OpenVPN будет принимать входящие соединения. Если этот адрес не указан - OpenVPN будет принимать входящие соединения со всех адресов.
- `rport` - это удаленный порт, к которому OpenVPN будет подключаться. Указывается с использованием `port`, но когда используется другой локальный порт - удобнее явно указать `rport`.
- `tun-ipv6` инструктирует OpenVPN создать туннель, способный пропускать трафик IPv6.
- `ifconfig-ipv6` настраивает локальные и удаленные конечные точки IPv6. В этом примере последние три числа адреса IPv6 соответствуют конечным точкам IPv4.
- `user nobody` и `group nobody` дают указание OpenVPN сменить UNIX-пользователя на `nobody` и группу после установки соединения. Это еще больше повышает безопасность, так как атака на туннель с меньшей вероятностью приведет к root-экспloitу. Обратите внимание, что в Debian/Ubuntu используется группа `nogroup`.
- `persist-tun` и `persist-key` инструктируют OpenVPN не поднимать устройство `tun` повторно или генерировать новый материал ключей при каждом перезапуске туннеля. Эти параметры особенно полезны в сочетании с `user nobody`, так как обычно у `nobody` нет прав доступа для запуска нового интерфейса `tun`.
- `keepalive 10 60` и `ping-timer-rem` являются полезными опциями дабы убедиться, что VPN-соединение остается работоспособным, даже если по туннелю нет трафика.

Вместо того, чтобы указывать очень длинную командную строку для запуска концов туннеля, теперь мы можем запустить оба конца используя следующие команды:

```
[root@server] # openvpn --config mvpn-02-02-server.conf
```

```
[root@client] # openvpn --config mvpn-02-02-client.conf
```

Проверьте файлы `openvpn.log` на обоих концах на наличие магического предложения:

```
Thu Sep 11 13:21:51 2014 Initialization Sequence Completed
```

Наконец, мы проверяем, что можем достичь другого конца туннеля, используя `ping` и `ping6`:

```
# remote IPv4 LAN address
$ ping -c 2 192.168.4.100
PING 192.168.4.100 (192.168.4.100) 56(84) bytes of data.
64 bytes from 192.168.4.100: icmp_seq=1 ttl=64 time=3 ms
64 bytes from 192.168.4.100: icmp_seq=2 ttl=64 time=5 ms
```

```

# remote IPv6 tunnel address
$ ping6 -c 2 2001:610:120::200:0:2
PING 2001:610:120::200:0:2(2001:610:120::200:0:2) 56 data bytes
64 bytes from 2001:610:120::200:0:2: icmp_seq=1 ttl=64 time=4 ms
64 bytes from 2001:610:120::200:0:2: icmp_seq=2 ttl=64 time=4 ms
# remote IPv6 LAN address
$ ping6 -c 2 2001:610:120::168:4:100
PING 2001:610:120::168:4:100(2001:610:120::168:4:100) 56 data byte
64 bytes from 2001:610:120::168:4:100: icmp_seq=1 ttl=64 time=6 ms
64 bytes from 2001:610:120::168:4:100: icmp_seq=2 ttl=64 time=3 ms

```

Обратите внимание - для того, чтобы заставить работать маршрутизацию, нам теперь необходима переадресация IP на обоих концах, а также обратный маршрут для компьютеров в сегменте LAN. В клиентской локальной сети нам нужны маршруты, подобные следующим:

```

# route add -net 10.200.0.0/24 gw 192.168.4.100
# route add -net 192.168.122.0/24 gw 192.168.4.100

```

Здесь 192.168.4.100 - это локальный адрес клиента OpenVPN.

В сети на стороне сервера нам необходимо следующее:

```

# route add -net 10.200.0.0/24 gw 192.168.122.1
# route add -net 192.168.4.0/24 gw 192.168.122.1

```

Здесь 192.168.122.1 - локальный адрес сервера OpenVPN.

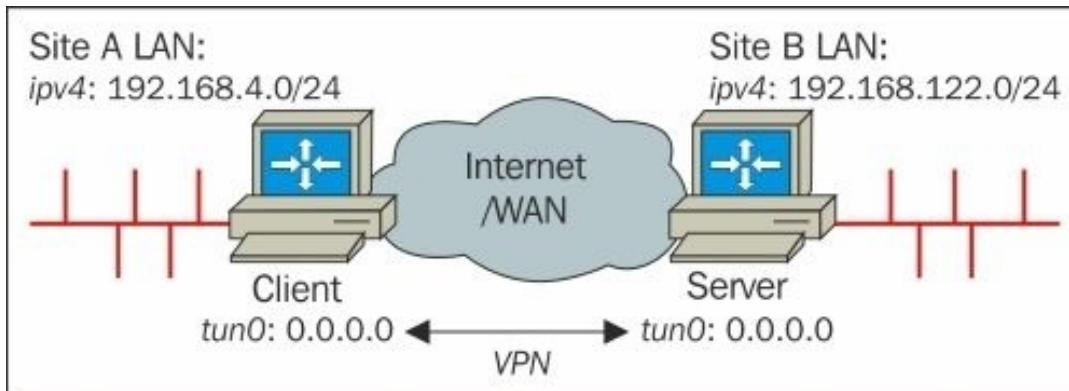
Заметка

В настоящее время требуется всегда указывать адрес IPv4 с помощью `ifconfig`, даже если туннель только для IPv6. Этот недостаток будет устранен в настройке OpenVPN 2.4+без-IP.

Расширенная настройка без-IP

Возможность OpenVPN разрешать выполнение пользовательских сценариев при запуске VPN-подключения позволяет выполнять некоторые расширенные настройки. В этом примере мы будем использовать пользовательский сценарий `up` для создания туннеля OpenVPN без назначения IP-адресов конечным точкам туннеля. При настройке маршрутизируемой сети это гарантирует что конечные точки туннеля никогда не будут доступны сами по себе, что добавляет некоторую безопасность, а также может сделать таблицы маршрутизации немного короче.

Этот сценарий был протестирован только в системах Linux, так как требует настройки сетевого интерфейса, недоступной на других платформах. Мы используем ту же схему сети, что и в предыдущем примере, но без адресации IPv6:



Для сервера мы создаем следующий файл конфигурации `movpn-02-03-server.conf`:

```
dev tun
secret secret.key
ifconfig-noexec

up /etc/openvpn/up.sh
script-security 2
verb 3

daemon
log-append /var/log/openvpn.log
```

Вот сопровождающий скрипт `up.sh`:

```
#!/bin/bash
/sbin/ifconfig $1 0.0.0.0 up
/sbin/ip route add 192.168.4.0/24 dev $1
```

Здесь сеть 192.168.4.0/24 - это клиентская локальная сеть, к которой мы хотим обратиться из серверной сети. Для клиента мы создаем файл `movpn-02-03-client.conf`:

```
dev tun
secret secret.key
ifconfig-noexec
remote openvpnserver.example.com

up /etc/openvpn/up.sh
script-security 2
verb 3
daemon
log-append /var/log/openvpn.log
```

Вот сопровождающий скрипт `up.sh`:

```
#!/bin/bash
/sbin/ifconfig $1 0.0.0.0 up
/sbin/ip route add 192.168.122.0/24 dev $1
```

Перед началом VPN-подключения убедитесь что сценарии `up.sh` являются исполняемыми (`chmod a+x up.sh`):

Запустите оба конца туннеля:

```
[root@server]# openvpn --config movpn-02-03-server.conf
```

```
[root@client]# openvpn --config movpn-02-03-client.conf
```

Проверьте файлы `openvpn.log` на обоих концах на наличие магического предложения:

```
Thu Sep 11 15:57:51 2014 Initialization Sequence Completed
```

Проверьте адреса, назначенные интерфейсу `tun0`:

```
$ ifconfig tun0
tun0      Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:504 (504.0 b) TX bytes:504 (504.0 b)
```

В качестве альтернативы вы можете использовать более современную команду `iproute2`:

```
$ ip addr show dev tun0
```

Интерфейс работает, но не имеет IP-адреса. Далее мы проверим таблицу маршрутизации:

```
$ /sbin/ip route show
[...]
192.168.4.0/24 dev tun0 scope link
[...]
```

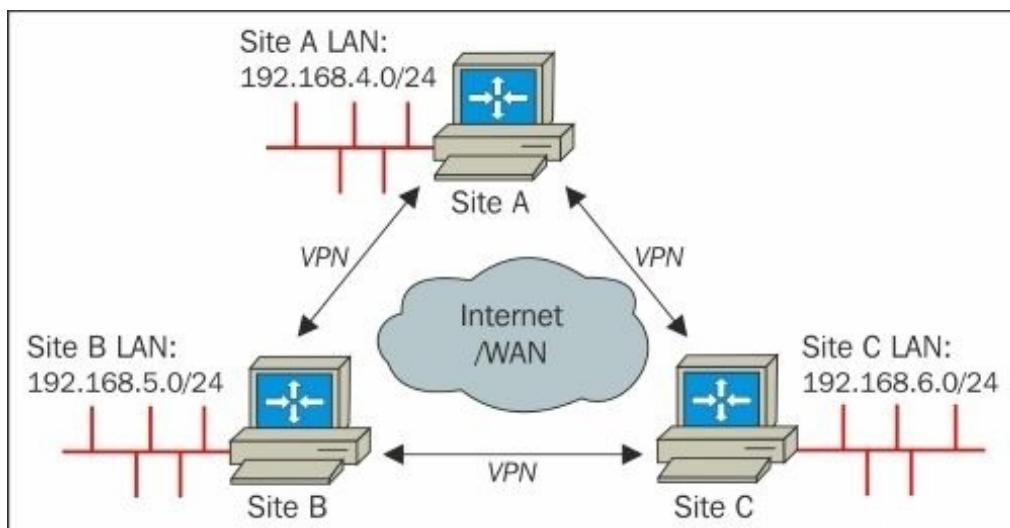
Правильный маршрут присутствует - поэтому, наконец, мы проверяем, что можем пинговать хост в локальной сети на стороне клиента:

```
[root@centos6-kvm ~]# ping -c 2 192.168.4.10
PING 192.168.4.10 (192.168.4.10) 56(84) bytes of data.
64 bytes from 192.168.4.10: icmp_seq=1 ttl=62 time=5 ms
64 bytes from 192.168.4.10: icmp_seq=2 ttl=62 time=5 ms
```

Трехсторонняя маршрутизация

Как указано во введении, двухточечные сети являются отличным выбором при подключении небольшого количества конечных точек. В этом примере мы покажем как соединить три места вместе, используя двухточечные туннели. Мы также покажем, как быстро простая конфигурация такой установки может стать очень сложной.

Рассмотрим следующую схему сети:



Мы создадим три туннеля между точками и избыточные маршруты. Таким образом, если один из туннелей выйдет из строя - все они останутся видимыми друг для друга. Тем не менее, это достигается за счет снижения производительности. Предположим, что связь между точками А и В не работает. Резервный маршрут идет от А к С и к В, поэтому теперь трафик с точки А на точку В должен сделать дополнительный скачок.

Сначала мы создаем три секретных ключа:

```
$ openvpn --genkey --secret AtoB.key
$ openvpn --genkey --secret AtoC.key
$ openvpn --genkey --secret BtoC.key
```

Затем мы передаем эти ключи всем конечным точкам по защищенному каналу (например, используя scp).

Теперь создаем шесть файлов конфигурации: три конфигурации сервера и три конфигурации клиента.

Сначала создайте файл конфигурации сервера BtoA.conf:

```
dev tun
proto udp
port 1194
remote siteA
secret AtoB.key 0
ifconfig 10.200.0.1 10.200.0.2
route 192.168.4.0 255.255.255.0 vpn_gateway 5
route 192.168.6.0 255.255.255.0 vpn_gateway 10
route-delay
keepalive 10 60
verb 3
daemon
log-append /var/log/openvpn-BtoA.log
```

Затем создайте CtoA.conf со следующим содержимым:

```
dev tun
proto udp
port 1195
remote siteA secret AtoC.key 0
ifconfig 10.200.0.5 10.200.0.6
route 192.168.4.0 255.255.255.0 vpn_gateway 5
route 192.168.5.0 255.255.255.0 vpn_gateway 10
route-delay
keepalive 10 60
verb 3
daemon
log-append /var/log/openvpn-CtoA.log
```

Далее создайте последний файл конфигурации сервера BtoC.conf:

```
dev tun
proto udp
port 1196
remote siteC
secret BtoC.key 0
ifconfig 10.200.0.9 10.200.0.10
route 192.168.4.0 255.255.255.0 vpn_gateway 10
route 192.168.6.0 255.255.255.0 vpn_gateway 5
route-delay
keepalive 10 60
verb 3
daemon
log-append /var/log/openvpn-BtoC.log
```

Теперь создадим файл конфигурации клиента (коннектора) AtoB.conf:

```
dev tun
proto udp
port 1194
remote siteB
secret AtoB.key 1
ifconfig 10.200.0.2 10.200.0.1
route 192.168.5.0 255.255.255.0 vpn_gateway 5
route 192.168.6.0 255.255.255.0 vpn_gateway 10
route-delay
keepalive 10 60
verb 3
daemon
log-append /var/log/openvpn-AtoB.log
```

Далее мы создаем файл конфигурации клиента AtoC.conf:

```
dev tun
proto udp
```

```

port 1195
remote siteC
secret AtoC.key 1
ifconfig 10.200.0.6 10.200.0.5
route 192.168.5.0 255.255.255.0 vpn_gateway 10
route 192.168.6.0 255.255.255.0 vpn_gateway 5
route-delay
keepalive 10 60
verb 3
daemon
log-append /var/log/openvpn-AtoC.log

```

Наконец, мы создаем файл конфигурации клиента CtoB.conf:

```

dev tun
proto udp
port 1196
remote siteB
secret BtoC.key 1
ifconfig 10.200.0.10 10.200.0.9
route 192.168.4.0 255.255.255.0 vpn_gateway 10
route 192.168.5.0 255.255.255.0 vpn_gateway 5
route-delay
keepalive 10 60
verb 3
daemon
log-append /var/log/openvpn-CtoB.log

```

Запустите каждый сервер и подключите соответствующего клиента:

```

[siteB]# openvpn --config BtoA.conf
[siteA]# openvpn --config AtoB.conf

```

Проверьте файлы журналов по обе стороны туннеля и убедитесь что маршрутизация (частично) работает, прежде чем переходить к следующему узлу:

```

[siteB]$ openvpn --config BtoC.conf
[siteC]$ openvpn --config CtoB.conf

```

и наконец:

```

[siteC]$ openvpn --config CtoA.conf
[siteA]$ openvpn --config AtoC.conf

```

На этом этапе должны присутствовать все маршруты, включая избыточные. Например, узел А имеет два маршрута к узлу В (LAN 192.168.5.0/24), как видно из таблицы маршрутизации:

```

[siteA]$ ip route show
[...]
192.168.5.0/24 via 10.200.0.1 dev tun0 metric 5
192.168.5.0/24 via 10.200.0.5 dev tun1 metric 10
[...]

```

Мы можем наблюдать следующее из этой таблицы:

- Один *прямой* туннель до узла В. Этот маршрут имеет самую низкую метрику.
- Один *непрямой* туннель: сначала на узел С, а затем на В. Этот маршрут имеет более высокую метрику и не выбирается до тех пор, пока не "упадет" первый маршрут.

Преимущество этой настройки заключается в том, что в случае сбоя одного туннеля через 60 секунд соединение и соответствующие ему маршруты сбрасываются. Затем резервный маршрут к другой сети автоматически поднимается и все три узла могут снова связаться друг с другом.

После возвращения исходного туннеля маршруты с более высокой метрикой снова получают приоритет и исходная ситуация восстанавливается.

Недостатком этой конфигурации является то, что в течение этих 60 секунд весь трафик теряется. Протоколы маршрутизации, такие как RIPv2 или OSPF, могут помочь обнаружить сбой маршрутов намного быстрее, что приведёт к меньшему времени простоя сети.

Маршрут, net_gateway, vpn_gateway и метрики

Следующие операторы конфигурации имеют жизненно важное значение в этой настройке. Слово `vpn_gateway` - это специальное ключевое слово OpenVPN, указывающее адрес удаленной конечной точки VPN. Обычно его указывать не нужно, кроме случаев, когда необходимо указать метрику для этого маршрута.

Синтаксис и параметры для директивы `route`:

```
route <network> <netmask> vpn_gateway <metric>
```

Здесь `gateway` можно либо указать явно в качестве адреса IPv4, либо использовать специальные ключевые слова `vpn_gateway` или `net_gateway`. Если шлюз и метрика не указаны - используется `vpn_gateway`.

Ключевое слово `net_gateway` полезно для указания подсети, которая не должна явно маршрутизоваться через VPN. В [Главе 4, Режим клиент/сервер с tun устройствами](#), будет дано более подробное объяснение вариантов `route`.

Метрика имеет значение по-умолчанию, которое можно установить с помощью следующей команды:

```
route-metric m
```

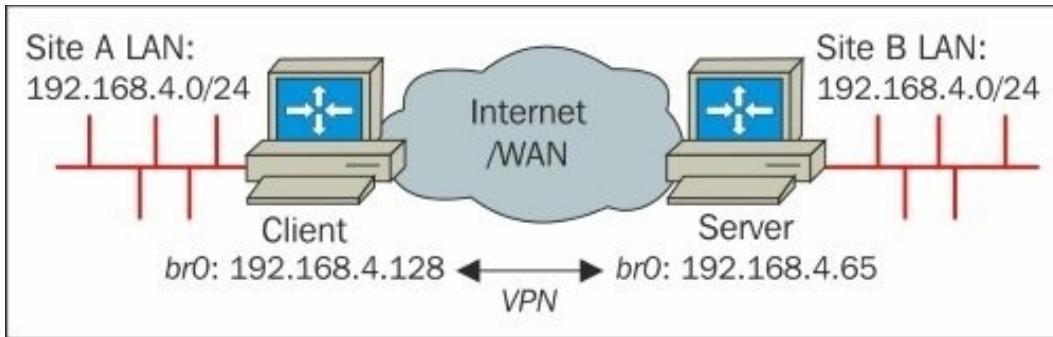
Это относится ко всем маршрутам. Если кто-то хочет отменить метрику для определенного маршрута (как мы это сделали в данном примере), то необходимо указать шлюз (в нашем случае `vpn_gateway`), за которым следует метрика для этого конкретного маршрута.

Инструкция `route-delay` необходима здесь, чтобы гарантировать добавление маршрутов после того, как все соединения доступны. Без этого маршруты могут быть добавлены слишком рано, что приведет к невозможности добавления маршрута в одну из удаленных подсетей.

Мостовой tap-переходник на обоих концах

Еще один сложный пример использования выделенной двухточечной VPN заключается в соединении двух удаленных сегментов сети. OpenVPN позволяет соединить два сегмента сети с одним и тем же диапазоном IP-адресов чтобы сформировать один прозрачный сегмент сети. Как правило, делать этого не рекомендуется, поскольку производительность такой объединенной сети не будет оптимальной. В некоторых случаях это неизбежно. На самом деле было бы лучше назначить разные подсети обоим концам, но иногда специфичное программное обеспечение привязано к определенному IP-адресу и нет альтернативы, кроме как иметь одну подсеть на обоих концах.

Рассмотрим следующую схему сети:



На стороне клиента используется сеть **192.168.4.0/24**, а клиент OpenVPN находится по адресу **192.168.4.128**. На стороне сервера используется та же подсеть - сервер OpenVPN находится по адресу **192.168.4.65**. Цель состоит в том, чтобы соединить две сети вместе для прозрачного видения всех машин на обоих концах.

В режиме `dev tap` OpenVPN создаст новый или откроет существующий tap-адаптер. В большинстве современных операционных систем tap-адаптер ведет себя так же, как обычный сетевой адаптер и если операционная система поддерживает мостовое подключение адаптера, то адаптер tap можно соединить с другим сетевым адаптером в системе. Известно, что это работает в Linux, Free/Open/NetBSD и Microsoft Windows. В Linux для работы этого примера необходимо установить пакет `bridge-utils`.

В приведенном примере tap-адаптер соединяется с интерфейсом локальной сети как клиента, так и сервера OpenVPN. Чтобы иметь возможность сделать это - tap-адаптер создается в постоянной работоспособности до инициализации VPN-подключения:

```
# openvpn --mktun --dev tap0
Thu Sep 11 16:57:30 2014 TUN/TAP device tap0 opened
Thu Sep 11 16:57:30 2014 Persist state set to: ON
```

Затем создается мост и инициализируется. На стороне клиента выполните следующие команды:

```
# brctl addbr br0
# brctl addif br0 eth0
# brctl addif br0 tap0
# ifconfig eth0 0.0.0.0 up
# ifconfig tap0 0.0.0.0 up
# ifconfig br0 192.168.4.128 netmask 255.255.255.0 up
```

Проверьте состояние моста и связанных с ним адаптеров, прежде чем продолжить.

Также убедитесь, что доступ к локальной сети все еще возможен:

```
# brctl show
bridge name      bridge id      STP enabled      interfaces
br0              8000.5c260a307224    no            eth0
                           tap0

# ifconfig -a
br0      Link encap:Ethernet HWaddr 5C:26:0A:30:72:24
        inet addr:192.168.4.128 Bcast:192.168.4.255 Mask:255.255.255.0
                  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                  RX packets:4 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:0
                  RX bytes:244 (244.0 b) TX bytes:732 (732.0 b)

eth0      Link encap:Ethernet HWaddr 5C:26:0A:30:72:24
                  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                  RX packets:2087 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:2427 errors:0 dropped:0 overruns:0 carrier:0
```

```

    collisions:0 txqueuelen:1000
    RX bytes:203516 (198.7 KiB) TX bytes:231571 (226.1 KiB)
    Interrupt:20 Memory:f5400000-f5420000
tap0      Link encap:Ethernet HWaddr CA:85:1E:AE:AF:59
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:11 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
# ping -c 2 192.168.4.10

```

Вывод должен быть одинаковым для серверной стороны - с IP-адресом моста **192.168.4.65**.

Далее мы создаем файл конфигурации **movpn-02-05.conf**, который может быть одинаковым с обеих сторон:

```

dev tap0
secret secret.key
verb 3
daemon
log-append /var/log/openvpn.log

```

Заметка

В файле конфигурации используется полное имя устройства **tap0**. Если значение **0** опущено, то OpenVPN откроет новый tap-адаптер и мост не будет работать.

Далее запускаем клиент и сервер:

```
[root@server] # openvpn --config movpn-02-05.conf \
--remote openvpnclient.example.com
```

```
[root@client] # openvpn --config movpn-02-05.conf \
--remote openvpnserver.example.com
```

После завершения последовательности инициализации - сегменты сети соединяются мостом. Проверьте это пропингуя хост на удаленном конце.

Также полезно наблюдать за трафиком, проходящим через туннель. Недостатком мостовой сети является распространение всего (широковещательного) трафика, генерируемого на одном конце. Это может привести к снижению производительности сети. Если в сети много *фонового шума* - это отобразится в **tcpdump** на интерфейсе **tap0**:

```
17:19:57.280459 72:24:b4:f0:16:81 > 01:80:c2:00:00:00, 802.3,
length 52: LLC, dsap STP (0x42) Individual, ssap STP (0x42)
Command, ctrl 0x03: STP 802.1d, Config, Flags [none], bridge-id
8000.52:54:00:6e:cd:0b.8003, length 35
```

```
17:19:59.280486 72:24:b4:f0:16:81 > 01:80:c2:00:00:00, 802.3,
length 52: LLC, dsap STP (0x42) Individual, ssap STP (0x42)
Command, ctrl 0x03: STP 802.1d, Config, Flags [none], bridge-id
8000.52:54:00:6e:cd:0b.8003, length 35
```

```
17:20:00.112516 52:54:00:6e:cd:0b > 98:4f:ee:00:7d:e1, ethertype
ARP (0x0806), length 42: Request who-has 192.168.122.23 tell
192.168.122.1, length 28
```

```
17:20:01.112534 52:54:00:6e:cd:0b > 98:4f:ee:00:7d:e1, ethertype
ARP (0x0806), length 42: Request who-has 192.168.122.23 tell
192.168.122.1, length 28
```

```
17:20:01.280468 72:24:b4:f0:16:81 > 01:80:c2:00:00:00, 802.3,
length 52: LLC, dsap STP (0x42) Individual, ssap STP (0x42)
Command, ctrl 0x03: STP 802.1d, Config, Flags [none], bridge-id
```

```

8000.52:54:00:6e:cd:0b.8003, length 35
17:20:02.112524 52:54:00:6e:cd:0b > 98:4f:ee:00:7d:e1, ethertype
ARP (0x0806), length 42: Request who-has 192.168.122.23 tell
192.168.122.1, length 28

17:20:04.161591 98:4f:ee:00:7d:e1 > ff:ff:ff:ff:ff:ff, ethertype
ARP (0x0806), length 60: Request who-has 192.168.4.100 tell
192.168.4.10, length 46

17:20:05.153670 98:4f:ee:00:7d:e1 > ff:ff:ff:ff:ff:ff, ethertype
ARP (0x0806), length 60: Request who-has 192.168.4.100 tell
192.168.4.10, length 46

```

Вывод данных этого захвата tcpdump показал, что протокол Spanning Tree Protocol был включен на мосту на стороне сервера, выполнив следующую команду:

```
# brctl stp br0 off
```

Этот трафик был остановлен, что привело к гораздо меньшему шуму на мосту.

Заметка

Отключайте STP на сетевом мосту только если знаете что делаете. В таком случае риск возникновения петель отсутствует, поскольку имеется только один мост и подключено только два устройства.

Запросы ARP, которые видны в захвате tcpdump не так просто подавить. Однако эти запросы очень малы и не должны приводить к значительному снижению производительности. Однако на линии с высокой задержкой - эти запросы станут узким местом.

Удаление мостов

Если мост больше не нужен - лучше удалить его и постоянные устройства tap0:

```

# ifconfig br0 down
# brctl delif br0 tap0
# brctl delif br0 eth0
# brctl delbr br0
# openvpn --rm tun --dev tap0
Thu Sep 11 18:55:22 2014 TUN/TAP device tap0 opened
Thu Sep 11 18:55:22 2014 Persist state set to: OFF

```

Не забудьте снова подключить сетевой интерфейс eth0.

Совмещение режима точка-точка с сертификатами

В следующем примере мы заимствуем некоторые части из [Главы 3](#), *PKI и сертификаты*. В режиме клиент/сервер OpenVPN настраивается с помощью **инфраструктуры публичных ключей (PKI)** с сертификатами X.509 и приватными ключами. Также можно использовать сертификаты X.509 и приватные ключи для настройки туннеля точка-точка. Преимущество использования сертификатов X.509 перед предустановленными ключами состоит в том, что он обеспечивает **Perfect Forwarding Secrecy (PFS)**, что значительно повышает безопасность ваших данных VPN. Без PFS, если злоумышленнику удастся в какой-то момент нарушить шифрование, то весь ранее записанный трафик VPN может быть расшифрован. С PFS невозможно расшифровать *старые* данные.

Чтобы настроить туннель точка-точка с использованием сертификатов - мы должны сначала скопировать сертификат CA и пару сертификат/приватный ключ для обеих конечных точек:

```
[root@server] # mkdir -p /etc/openvpn/movpn
[root@server] # chmod 700 /etc/openvpn/movpn
[root@server] # cd /etc/openvpn/movpn
[root@server] # PKI=<PKI_DIR>/ssladm/active
[root@server] # cp -a $PKI/ca.crt movpn-ca.crt
[root@server] # cp -a $PKI/Mastering_OpenVPN_Server.crt server.crt
[root@server] # cp -a $PKI/Mastering_OpenVPN_Server.key server.key
```

и

```
[root@client] # mkdir -p /etc/openvpn/movpn
[root@client] # chmod 700 /etc/openvpn/movpn
[root@client] # cd /etc/openvpn/movpn
[root@client] # PKI=<PKI_DIR>/ssladm/active
[root@client] # cp -a $PKI/ca.crt movpn-ca.crt
[root@client] # cp -a $PKI/client1.crt client1.crt
[root@client] # cp -a $PKI/client1.key client1.key
```

На стороне сервера нам также необходимо сгенерировать файл параметров Диффи-Хеллмана, который необходим для ключей сеанса VPN. Ключи сеанса являются кратковременными или временными ключами и генерируются когда устанавливается соединение между клиентом и сервером.

Чтобы сгенерировать файл параметров Диффи-Хеллмана, выполните следующие команды:

```
[root@server] # cd /etc/openvpn/movpn
[root@server] # openssl dhparam -out dh2048.pem 2048
```

Теперь мы готовы настроить файлы конфигурации OpenVPN. На стороне сервера создайте следующий файл конфигурации и сохраните его как movpn-02-06-server.conf:

```
proto udp
port 1194
dev tun
tls-server
ifconfig 10.200.0.1 10.200.0.2
tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key
persist-key
persist-tun
keepalive 10 60
user nobody
group nobody
# use 'group nogroup' on Debian/Ubuntu
verb 3
daemon
log-append /var/log/openvpn.log
```

На стороне клиента создайте файл конфигурации movpn-02-06-client.conf:

```
port 1194
dev tun
tls-client
ifconfig 10.200.0.2 10.200.0.1
remote openvpnserver.example.com
remote-cert-tls server
tls-auth /etc/openvpn/movpn/ta.key 1
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/client1.crt
key /etc/openvpn/movpn/client1.key
persist-key
```

```
persist-tun
keepalive 10 60
user nobody
group nobody
# use 'group nogroup' on Debian/Ubuntu
verb 3
daemon
log-append /var/log/openvpn.log
```

Далее запускаем клиент и сервер:

```
[root@server] # openvpn --config movpn-02-06-server.conf
```

```
[root@client] # openvpn --config movpn-02-06-client.conf
```

После завершения последовательности инициализации мы увидим, что созданный туннель имеет те же свойства, что и туннель, созданный с помощью предустановленных ключей.

Резюме

Точка-точка была единственной поддерживаемой конфигурацией в начальных версиях OpenVPN. В этой главе мы начали с очень простого примера «точка-точка». Мы представили больше возможностей OpenVPN и увидели, что есть веские причины использовать этот режим в производственной среде. В последнем случае мостовые tap-адAPTERы используются как на клиентской стороне, так и на серверной.

Это единственная глава, в которой объясняется режим «точка-точка». В следующей главе мы правильно настроим сертификаты, необходимые для использования другого режима OpenVPN - модели клиент/сервер.

Глава 3. PKI и сертификаты

В первую очередь OpenVPN использует сертификаты X.509 для аутентификации клиента и шифрования трафика VPN, хотя эту поддержку можно отключить. Просмотр списка рассылки и истории IRC-каналов, настройка и обслуживание **инфраструктуры приватного ключа (PKI)** для сертификатов X.509 является сложной концепцией и может быть трудоемкой задачей.

Двоичный файл OpenSSL содержит все инструменты, необходимые для ручного управления PKI, но параметры команды сложны и, если не автоматизированы, могут быть подвержены ошибкам. Организациям или частным лицам рекомендуется использовать скрипт или другой пакет для управления своей PKI. Это не только ограничивает количество ошибок, но также позволяет лучше соблюдать правила и другие общие критерии.

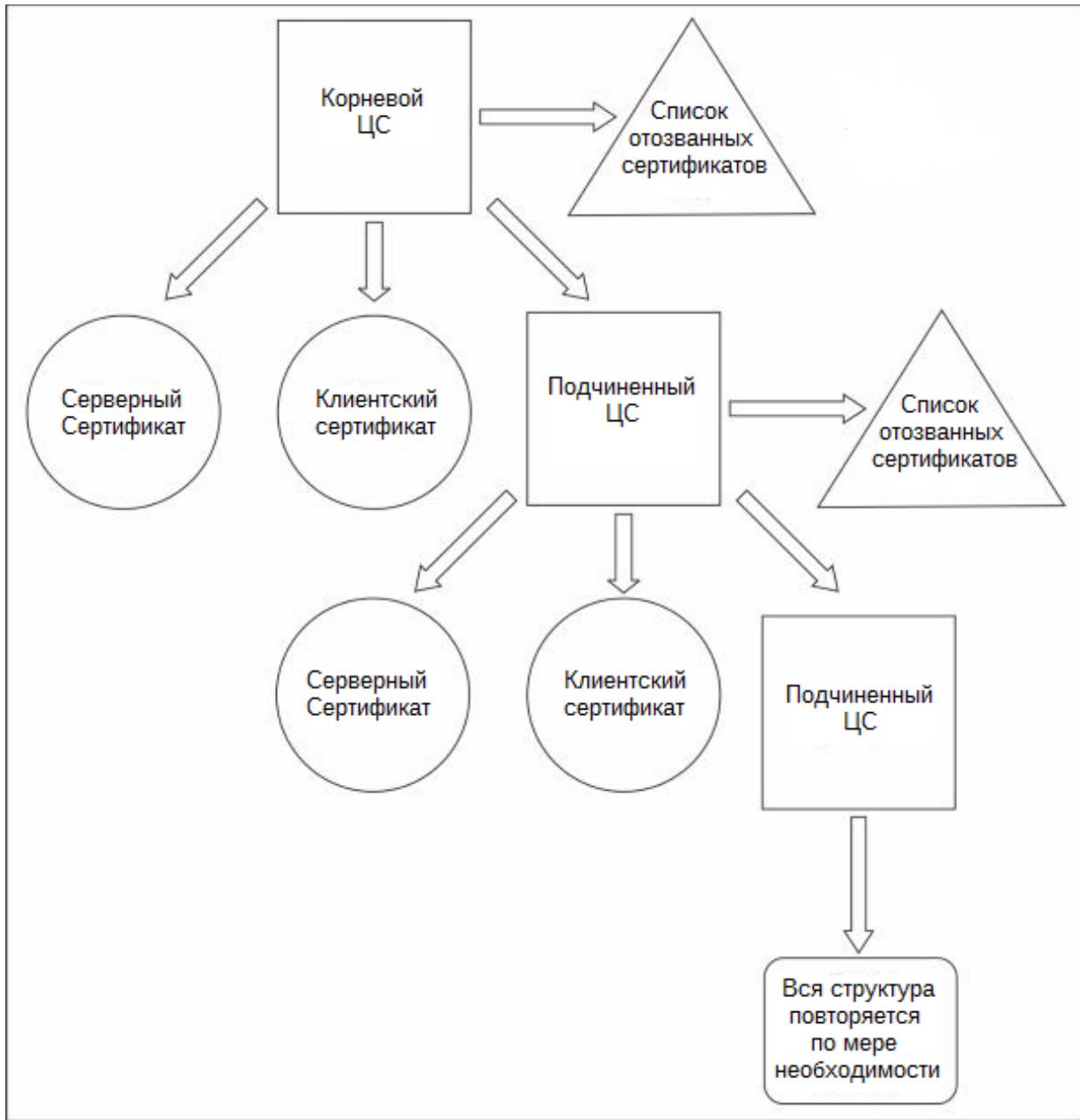
Существуют два проекта с открытым исходным кодом, которые специально написаны для хорошей работы с реализациями OpenVPN. EasyRSA - это давний проект, который всегда был тесно связан с проектом OpenVPN. Первоначально написанный вместе с OpenVPN - его первоначальная цель заключалась в создании **центра сертификации (Certificate Authority - CA)** и необходимых компонентов. Сегодня этот проект все еще поддерживается вместе с проектом OpenVPN, хотя они технически разделены.

Другой проект - ssl-admin, представляет собой Perl-скрипт, написанный для заполнения видимых пробелов в коде EasyRSA. Оба проекта по-разному подходят к задачам управления PKI и оба имеют уникальное решение. Проект ssl-admin - это интерактивный скрипт, предоставляющий меню и отзывы пользователей, в то время как EasyRSA - это, прежде всего, пакетная утилита.

На сегодняшний день Эрик Крист поддерживает проекты EasyRSA и ssl-admin. Джош Сепек присоединился и написал большую часть исходного кода EasyRSA v3.0. Цель состоит в том, чтобы в конечном итоге объединить два проекта и сохранить все функциональные возможности обоих.

Обзор PKI

PKI - это, как правило, иерархическая организация сертификатов шифрования и пар ключей. Как правило, как и в большинстве веб-сайтов, вершиной иерархии является CA. Это корень всего дерева, и доверие основано на этом уровне. Если корень является доверенным - все пары ключей, лежащие в основе, также будут доверенными. Из CA корневого уровня могут быть клиентские сертификаты, серверные сертификаты, подчиненные CA и **справки отзыва сертификатов (certificate revocation lists - CRL)**. Под каждым подчиненным CA этот список возможностей повторяется.



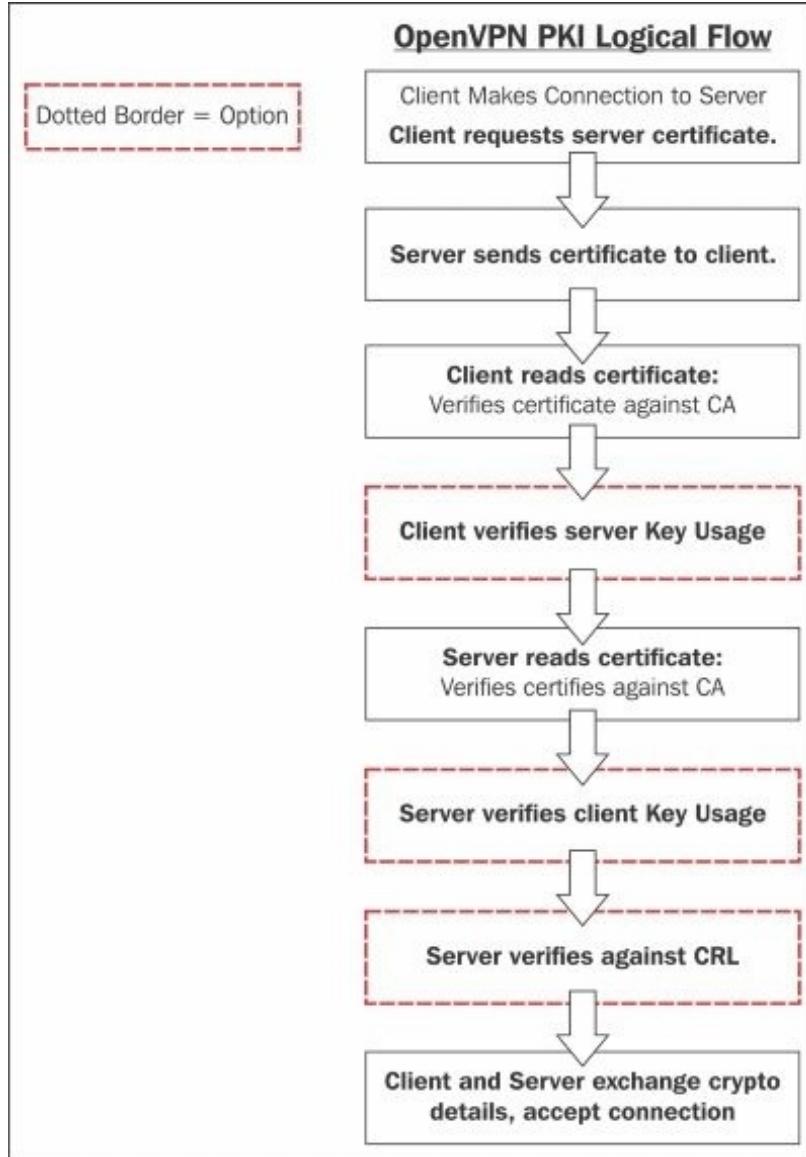
Чтобы использовать PKI в полной мере - пользователи и системы должны доверять корневому ЦС и любым промежуточным ЦС в цепочке. В большинстве современных веб-браузеров авторы или поставщики браузера по умолчанию проверяют и утверждают большой список центров сертификации корневого уровня, которым можно доверять. Этими органами обычно являются коммерческие поставщики, такие как VeriSign, Go Daddy, Comodo, Trend Micro, различные государственные структуры и многие другие.

Из-за этого предварительно одобренного списка для браузеров, подавляющее большинство пользователей Интернета совершенно не знают о том, как работает PKI. В результате, новые пользователи OpenVPN и любого программного обеспечения, требующего PKI, находят настройку и управление препятствием - как концептуально, так и технически. В случае общедоступных веб-сайтов требуется проверка стороннего сайта доверенным органом. Однако в контексте OpenVPN, как правило, в организации существует единый объект, которому неявно доверяют - отдел IT. OpenVPN обычно используется в одной организации, и доверие к нему является неотъемлемым. EasyRSA и ssl-admin были написаны в помощь как начинающим, так и опытным пользователям для управления своей PKI.

При использовании единственной двухточечной связи часто не имеет смысла включать сложность PKI для защиты туннеля - предустановленных общих ключей достаточно. Однако, когда в дело вовлечено много пользователей - существует гораздо больший потенциал для потерянных и украденных ключей и текучести кадров. При правильно настроенной PKI отозвать утерянный сертификат или увольняющегося сотрудника относительно просто. Новый также легко создается и перераспределяется.

Используя EasyRSA и ssl-admin - мы создадим простую PKI с ЦС, сертификатом сервера, некоторыми сертификатами клиента и списком отзыва сертификатов. Кроме того, мы будем использовать эти утилиты для генерации параметров **Диффи-Хеллмана (DH)**, которые будут использоваться и обсуждаться в последующих главах.

Поток OpenVPN PKI выглядит следующим образом:



PKI с использованием EasyRSA

На момент написания этой главы EasyRSA 2.2.2 была официально последней версией наряду с v3.0.0-rc2. Поскольку серия v3.0 близится к выпуску - этот раздел будет посвящен ей. Релизы для EasyRSA можно найти по адресу <https://github.com/OpenVPN/easyrsa/releases>.

Это упражнение продемонстрирует как построить ЦС с нуля. Обновление с EasyRSA 2.2 здесь не рассматривается. После загрузки пакета EasyRSA распакуйте файлы и вы должны будете найти каталог с извлеченными файлами (в нашем случае EasyRSA-3.0.0-rc2):

```
ecrist@computer:~/Downloads-> tar -xzvf EasyRSA-3.0.0-rc2.tgz
x EasyRSA-3.0.0-rc2/
x EasyRSA-3.0.0-rc2/x509-types/
x EasyRSA-3.0.0-rc2/x509-types/server
x EasyRSA-3.0.0-rc2/x509-types/ca
x EasyRSA-3.0.0-rc2/x509-types/COMMON
x EasyRSA-3.0.0-rc2/x509-types/client
x EasyRSA-3.0.0-rc2/openssl-1.0.cnf
x EasyRSA-3.0.0-rc2/ChangeLog
x EasyRSA-3.0.0-rc2/Licensing/
x EasyRSA-3.0.0-rc2/Licensing/gpl-2.0.txt
x EasyRSA-3.0.0-rc2/COPYING
x EasyRSA-3.0.0-rc2/KNOWN_ISSUES
x EasyRSA-3.0.0-rc2/doc/
x EasyRSA-3.0.0-rc2/doc/Hacking.md
x EasyRSA-3.0.0-rc2/doc/EasyRSA-Upgrade-Notes.md
x EasyRSA-3.0.0-rc2/doc/EasyRSA-Readme.md
x EasyRSA-3.0.0-rc2/doc/EasyRSA-Advanced.md
x EasyRSA-3.0.0-rc2/doc/Intro-To-PKI.md
x EasyRSA-3.0.0-rc2/README.quickstart.md
x EasyRSA-3.0.0-rc2/vars.example
x EasyRSA-3.0.0-rc2/easyrsa
```

После извлечения - скопируйте файл vars.example как vars. Рекомендуется установить рабочий каталог EasyRSA. Стока 45 из vars определяет EASYRSA в \$PWD (текущий рабочий каталог) по умолчанию. Это может оказаться проблемой, особенно если вы используете EasyRSA для управления несколькими центрами сертификации. Раскомментируйте эту строку и измените ее на что-то подходящее для вашей среды, например, /usr/local/etc/easyrsa.

Подсказка

При инициализации EasyRSA все содержимое каталога EASYRSA будет удалено. Будьте осторожны с тем, что вы определяете здесь. В каталоге EASYRSA находится хранилище сертификатов. Это *не* местонахождение исполняемых файлов и переменных.

Вы, вероятно, захотите установить организационные поля, являющиеся следующими переменными:

- EASYRSA_REQ_COUNTRY
- EASYRSA_REQ_PROVICE
- EASYRSA_REQ_CITY
- EASYRSA_REQ_ORG
- EASYRSA_REQ_EMAIL
- EASYRSA_REQ_OU

Значения используются в качестве значений по умолчанию для всех сгенерированных запросов на сертификат, включая корневой CA. Убедитесь что эти строки не закомментированы в файле. Никаких других изменений в vars не требуется.

Если определенный вами каталог EASYRSA не существует - создайте его сейчас и скопируйте файл openssl-1.0.cnf из пакета дистрибутива в ваш новый каталог. Для наших примеров мы поместили хранилище сертификатов EASYRSA в /usr/local/etc/easyrsa:

```
ecrist@computer:~/Downloads/EasyRSA-3.0.0-rc2-> mkdir -p
/usr/local/etc/easyrsa
ecrist@computer:~/Downloads/EasyRSA-3.0.0-rc2-> cp openssl-1.0.cnf
/usr/local/etc/easyrsa
ecrist@computer:~/Downloads/EasyRSA-3.0.0-rc2-> cp -R x509-types
```

/usr/local/etc/easyrsa/

Далее мы готовы инициализировать EasyRSA PKI:

ecrist@computer:~/Downloads/EasyRSA-3.0.0-rc2-> ./easyrsa init-pki

Обратите внимание - мы используем конфигурацию EasyRSA из ./vars.

**init-pki complete; you may now create a CA or requests.
Your newly created PKI dir is: /usr/local/etc/easyrsa/pki**

В этом случае процесс инициализации очищает содержимое - в нашем случае - каталога pki и создает подкаталоги private и reqs.

Вы можете иметь несколько файлов vars для управления несколькими ЦС и все они могут быть вложены в один корневой каталог EASYRSA. Чтобы сделать это - вы должны изменить переменную EASYRSA_PKI для каждого ЦС.

Создание СА

Подкоманда build-са сначала генерирует **запрос на подпись сертификата (Certificate Signing Request - CSR)**, а затем самостоятельно подписывает этот запрос.

Чтобы создать пару сертификат/ключ корневого центра сертификации, выполните команду build-са:

```
ecrist@computer:~/Downloads/EasyRSA-3.0.0-rc2-> ./easyrsa build-ca
Note: using EasyRSA configuration from: ./vars
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to '/usr/local/etc/easyrsa/pki/private/ca.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
```

Вас попросят ввести информацию, которая будет включена в ваш запрос на сертификат. То, что вы собираетесь ввести - называется уникальным именем или DN. Есть довольно много полей, но некоторые вы можете оставить пустыми; в некоторых полях будет определено значение по умолчанию; если вы введете '!' - поле останется пустым.

```
-----  
Common Name (eg: your user, host, or server name) [EasyRSA CA]:Mastering  
OpenVPN
```

Подсказка

Пробелы не должны использоваться в поле Common Name (Общее имя). Это вызовет проблемы в EasyRSA, а также, возможно, с CCD, общим именем в качестве имени пользователя и другими случаями. Убедитесь, что после пути есть косая черта (/) или некоторые подкоманды (gen-crl и другие) не будут работать.

CA creation complete and you may now import and sign cert requests.

Your new CA certificate file for publishing is at:

/usr/local/etc/easyrsa/pki/ca.crt

Создание СА завершено, и теперь вы можете импортировать и подписывать сертификаты.

**Ваш новый файл сертификата ЦС для публикации находится по адресу:
/usr/local/etc/easyrsa/pki/ca.crt**

При самоподписании для ограничения СА устанавливается значение true и определяются параметры использования ключа, что позволяет этому новому сертификату подписывать другие

сертификаты, включая список отзыва сертификатов (CRL). Эта информация может быть проверена с помощью утилиты командной строки openssl:

```
ecrist@computer:~/Downloads/EasyRSA-3.0.0-rc2-> openssl x509 -in
/usr/local/etc/easyrsa/pki/ca.crt -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      89:39:42:bb:f3:6b:a9:f6
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: CN=Mastering OpenVPN
    Validity
      Not Before: Oct 1 15:02:44 2014 GMT
      Not After : Sep 28 15:02:44 2024 GMT
    Subject: CN=Mastering OpenVPN
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (2048 bit)
        Modulus (2048 bit):
          ...
          Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Subject Key Identifier:
        69:6C:A3:85:63:61:09:DE:8F:7D:38:F7:A2:CB:1C:31:75:90:34:93
      X509v3 Authority Key Identifier:
        keyid:69:6C:A3:85:63:61:09:DE:8F:7D:38:F7:A2:CB:1C:31:75:90:34:93
        DirName:/CN=Mastering OpenVPN
        serial:89:39:42:BB:F3:6B:A9:F6
      X509v3 Basic Constraints:
        CA:TRUE
      X509v3 Key Usage:
        Certificate Sign, CRL Sign
      Signature Algorithm: sha256WithRSAEncryption
      ...
```

В выводе openssl мы видим x509v3 Basic Constraints, а также параметры x509v3 Key Usage. Теперь ваш ЦС готов начать подписывать сертификаты клиента и сервера.

Список отзыва сертификатов

Подкоманда gen-crl генерирует список отзыва сертификатов. Хотя на данный момент у нас есть только сертификат ЦС - рекомендуется создать пустой CRL. Это позволит вам создавать конфигурацию OpenVPN с файлом и не требовать перезагрузки позже. OpenVPN будет регистрировать ошибки если в его конфигурации указан несуществующий CRL, но файл можно заменить на лету, так как он перечитывается при каждом клиентском подключении.

```
ecrist@computer:~/Downloads/EasyRSA-3.0.0-rc2-> ./easyrsa gen-crl
Note: using EasyRSA configuration from: ./vars
Using configuration from /usr/local/etc/easyrsa/openssl-1.0.cnf
Enter pass phrase for /usr/local/etc/easyrsa/pki/private/ca.key:

An updated CRL has been created.
CRL file: /usr/local/etc/easyrsa/pki/crl.pem
```

Мы можем проверить CRL, используя команду openssl crl:

```
ecrist@computer:~/Downloads/EasyRSA-3.0.0-rc2-> openssl crl -noout -text -in
/usr/local/etc/easyrsa/pki/crl.pem
Certificate Revocation List (CRL):
    Version 2 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: /CN=Mastering OpenVPN
    Last Update: Oct 1 15:50:46 2014 GMT
    Next Update: Mar 30 15:50:46 2015 GMT
    CRL extensions:
        X509v3 Authority Key Identifier:
            X509v3 Authority Key Identifier:
keyid:69:6C:A3:85:63:61:09:DE:8F:7D:38:F7:A2:CB:1C:31:75:90:34:93
    DirName:/CN=Mastering OpenVPN
    serial:89:39:42:BB:F3:6B:A9:F6

No Revoked Certificates.
Signature Algorithm: sha256WithRSAEncryption
...
```

Серверные сертификаты

OpenVPN может использовать параметры использования ключа x509, гарантируя, что клиенты соединяются с действительными клиентскими сертификатами, и клиенты могут гарантировать, что сервер авторизован как сервер. Это предотвращает использование одного из ваших клиентских сертификатов в качестве сервера при атаке **Man-In-The-Middle (MITM)**. Без этого ограничения любой сертификат, подписанный ЦС VPN, может использоваться для подмены клиента или сервера. Поскольку поддельный сертификат находится в той же PKI - сам сертификат все еще действителен и будет проходить общие проверки PKI.

EasyRSA поддерживает подписывание сертификатов с помощью параметра использования ключа сервера с использованием встроенной под команды `build-server-full`.

```
ecrist@computer:~/EasyRSA-3.0.0-rc2-> ./easyrsa build-server-full movpn-
server

Note: using EasyRSA configuration from: ./vars
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to
'/usr/local/etc/easyrsa/pki/private/movpnserver.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
Using configuration from /usr/local/etc/easyrsa/openssl-1.0.cnf
Enter pass phrase for /usr/local/etc/easyrsa/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName      :PRINTABLE:'movpn-server'
Certificate is to be certified until Oct 18 19:15:31 2024 GMT (3650
days)

Write out database with 1 new entries
Data Base Updated
```

Мы можем проверить сертификат сервера с помощью команды `openssl`:

```
ecrist@computer:~/EasyRSA-3.0.0-rc2-> openssl x509 -noout -text -in
/usr/local/etc/easyrsa/pki/issued/movpn-server.crt
Certificate:
```

```

Data:
  Version: 3 (0x2)
  Serial Number: 1 (0x1)
Signature Algorithm: sha256WithRSAEncryption
  Issuer: CN=Mastering OpenVPN
Validity
  Not Before: Oct 21 19:15:31 2014 GMT
  Not After : Oct 18 19:15:31 2024 GMT
Subject: CN=movpn-server
Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
    Public-Key: (2048 bit)
      Modulus:
        ...
        Exponent: 65537 (0x10001)
X509v3 extensions:
  X509v3 Basic Constraints:
    CA:FALSE
  X509v3 Subject Key Identifier:
    11:AD:47:E2:1C:87:91:3B:C0:A1:53:F5:77:A7:2F:9F:0B:0F:5D:9E
  X509v3 Authority Key Identifier:
    keyid:F0:AF:20:ED:6F:A7:47:0F:C7:F2:B0:EC:CF:8B:30:09:02:E4:81:A0
      DirName:/CN=Mastering OpenVPN
      serial:84:9E:B9:14:2B:62:B4:50

  X509v3 Extended Key Usage:
    TLS Web Server Authentication
  X509v3 Key Usage:
    Digital Signature, Key Encipherment
Signature Algorithm: sha256WithRSAEncryption
  ...

```

Обратите внимание на раздел X509v3 Key Usage и его идентификацию TLS Web Server Authentication. Это то, что ищут текущие версии OpenVPN, когда указаны типы удаленных сертификатов.

Клиентские сертификаты

Как и сертификаты сервера - клиенты могут проходить аутентификацию с использованием клиентских сертификатов. При использовании этого метода каждому клиенту потребуется уникальный сертификат. **Общее имя** сертификата (**Common Name - CN**) можно использовать для определения других параметров, которые должны передаваться в данном соединении с помощью сценариев `client-connect` или опции `client-config-dir`. Начиная с OpenVPN 2.3.7, все еще поддерживается `-client-cert-not-required`. Говорили об удалении этой поддержки из будущего выпуска. `client-cert-not-required` позволяет клиенту подключаться без уникального (или какого-либо) предварительно определенного сертификата, так же как веб-браузер будет подключаться к веб-серверу.

Команда `easyrsa` используется для создания сертификата клиента почти так же, как мы создали сертификат сервера:

```

ecrist@computer:~/EasyRSA-3.0.0-rc2-> ./easyrsa build-client-full
client1
Note: using EasyRSA configuration from: ./vars
Generating a 2048 bit RSA private key
.....+++
.....+++
.....+++

```

```
writing new private key to
'homeecrist/EasyRSA-3.0.0-rc2/pki/private/client1.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
Using configuration from homeecrist/EasyRSA-3.0.0-rc2/openssl-1.0.cnf
Enter pass phrase for homeecrist/EasyRSA-3.0.0-rc2/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName :PRINTABLE:'client1'
Certificate is to be certified until Oct 18 19:37:40 2024 GMT (3650 days)
```

```
Write out database with 1 new entries
Data Base Updated
```

Мы можем использовать команду openssl для проверки параметров использования ключа:

```
ecrist@computer:~/EasyRSA-3.0.0-rc2-> openssl x509 -noout -text -in
/usr/local/etc/easyrsa/pki/issued/client1.crt
```

Certificate:

```
    Data:
        Version: 3 (0x2)
        Serial Number: 2 (0x2)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=Mastering OpenVPN
        Validity
            Not Before: Oct 21 19:37:40 2014 GMT
            Not After : Oct 18 19:37:40 2024 GMT
        Subject: CN=client1
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
            Modulus:
                ...
            Exponent: 65537 (0x10001)
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA:FALSE
        X509v3 Subject Key Identifier:
```

```
47:80:59:8D:5F:63:4B:1C:21:C6:DE:C6:C0:7B:DE:6A:5D:53:F9:37
    X509v3 Authority Key Identifier:
```

```
keyid:F0:AF:20:ED:6F:A7:47:0F:C7:F2:B0:EC:CF:8B:30:09:02:E4:81:A0
    DirName:/CN=Mastering OpenVPN
    serial:84:9E:B9:14:2B:62:B4:50
    X509v3 Extended Key Usage:
        TLS Web Client Authentication
    X509v3 Key Usage:
        Digital Signature
    Signature Algorithm: sha256WithRSAEncryption
    ...
```

Как отмечено в разделе о сервере, мы рассмотрим x509v3 Extended Key Usage. В случае клиента мы ищем TLS Web Client Authentication. Опять же - он используется при проверке типа сертификата удаленного клиента.

PKI с использованием ssl-admin

Проект ssl-admin был запущен в то время, когда EasyRSA считался заброшенным и сломанным. Это интерактивная утилита на основе меню, написанная на Perl. Как и EasyRSA, ssl-admin - это

оболочка для утилит командной строки OpenSSL.

Чтобы установить ssl-admin во FreeBSD, установите порт `security/ssl-admin`. Для всех других Unix-систем, включая OS X, самым простым методом является экспорт svn. Следующие примеры выполняются на Mac OS X, но будут аналогичны для других операционных систем.

Чтобы получить ssl-admin в ОС, отличной от FreeBSD, используйте утилиту командной строки SVN для экспорта текущей версии:

```
ftp://ftp.secure-computing.net/pub/ssl-admin/  
  
ecrist@computer:~-> curl -o ssa.tgz ftp://ftp.secure-computing.net//pub/ssl-  
admin/ssl-admin-1.2.1.tar.gz  
% Total % Received % Xferd Average Speed Time Time Time  
Current Dload Upload Total Spent Left  
Speed  
100 11196 100 11196 0 0 17568 0 --:--:-- --:--:-- --:--:--  
17548
```

Подсказка

Сервер `ftp2.secure-computing.net` может быть использован в качестве альтернативы, если основной `ftp.secure-computing.net` недоступен.

```
ecrist@computer:~-> tar -xzvf ssa.tgz  
x ssl-admin-1.2.1/  
x ssl-admin-1.2.1/man5/  
x ssl-admin-1.2.1/man1/  
x ssl-admin-1.2.1/ssl-admin  
x ssl-admin-1.2.1/ssl-admin.conf  
x ssl-admin-1.2.1/Makefile  
x ssl-admin-1.2.1/configure  
x ssl-admin-1.2.1/openssl.conf  
x ssl-admin-1.2.1/ssl-admin-e  
x ssl-admin-1.2.1/man1/ssl-admin.1  
x ssl-admin-1.2.1/man1/ssl-admin.1-e  
x ssl-admin-1.2.1/man5/ssl-admin.conf.5  
x ssl-admin-1.2.1/man5/ssl-admin.conf.5-e
```

Чтобы установить после экспорта - измените директорию на ваше экспортируемое дерево и запустите `./configure`, а затем `make install`:

```
ecrist@computer:~/ssl-admin-1.2.1-> ./configure
```

Подсказка

Bourne Shell является единственным требованием для настройки. Это не типичный набор правил настройки - он просто имитирует его поведение.

```
ecrist@computer:~/ssl-admin-1.2.1-> sudo make install
```

После установки при запуске команды `ssl-admin` сначала отобразится ошибка:

```
ecrist@computer:~/ssl-admin-> ssl-admin  
Libraryssl-admin/ssl-admin.conf doesn't exist. Did you copy the  
sample from Libraryssl-admin/ssl-admin.conf.sample? at  
/usr/local/bin/ssl-admin line 40.
```

Подсказка

Некоторые версии `ssl-admin` ссылаются на следующий файл: `ssl-admin.conf.default`.

Чтобы начать использовать программное обеспечение - необходимо скопировать файл по

умолчанию в место, указанное в сообщении об ошибке. Это зависит от операционной системы в зависимости от стандартной иерархии файловой системы. Здесь мы скопируем `ssl-admin.conf.sample` в `ssl-admin.conf` и исправим разрешения для нового файла.

Заметка

`ssl-admin` требует чтобы все операции выполнялись от имени пользователя `root`. Это известная плохая практика и она будет исправлена в следующем выпуске.

Команды для этого следующие:

```
ecrist@computer:~/ssl-admin-> sudo csh  
Password:
```

Нет особой причины использовать `csh` - это просто предпочтение автора (лучшее или худшее).

```
root@computer:~/ssl-admin-> cd Libraryssl-admin/  
root@computer:Libraryssl-admin-> cp ssl-admin.conf.sample ssl-admin.conf  
root@computer:Libraryssl-admin-> chmod ug+rw ssl-admin.conf  
root@computer:Libraryssl-admin-> ls -l  
total 12  
-r--r--r-- 1 root wheel 2511 Oct 1 12:02 openssl.conf.sample  
-rw-rw-r-- 1 root wheel 531 Oct 1 12:11 ssl-admin.conf  
-r--r--r-- 1 root wheel 531 Oct 1 12:02 ssl-admin.conf.sample
```

Отсюда мы можем редактировать файл конфигурации. Как правило, необходимо изменить только нижние переменные:

- `$ENV{'KEY_COUNTRY'}`
- `$ENV{'KEY_PROVINCE'}`
- `$ENV{'KEY_CITY'}`
- `$ENV{'KEY_ORG'}`
- `$ENV{'KEY_EMAIL'}`
- `$ENV{'KEY_COUNTRY'}`

Переменная `KEY_COUNTRY` должна состоять из двух букв. Это ограничение/запрет стандарта, а не `ssl-admin`. Эти значения совпадают с одноименными переменными в `EasyRSA`. Они не должны быть изменены после создания ЦС, так как `openssl` будет выдавать ошибки для несовпадающих значений.

Подсказка

`$ENV{'KEY_CRL_LOC'}` следует оставить в покое, если в вашей организации нет действующего URL-адреса для распространения CRL. Пустое значение приведет к ошибкам с `openssl`.

Как только файл конфигурации был отредактирован - мы можем запустить программу. При первом запуске `ssl-admin` проверит хранилище сертификатов. Если в нем нет действующего ЦС и структуры - пользователь может импортировать существующую PKI или создать новую. Как и в случае с `EasyRSA` мы создаем новый центр сертификации. В некоторых операционных системах файл `/etc/openssl.cnf` необходимо скопировать вручную в корень хранилища сертификатов. Эта ошибка не встречалась в `Mac OS X`. В системах `Linux` просто скопируйте файл `/etc/openssl.cnf` в файл `/etc/ssl-admin/openssl.cnf`. Это будет исправлено в более позднем выпуске `ssl-admin`.

```
root@computer:/Library/ssl-admin-> ssl-admin  
This program will walk you through requesting, signing,  
organizing and revoking SSL certificates.
```

```
Looks like this is a new install, installing...
You will first need to edit the /Library/ssl-admin/ssl-admin.conf
default variables. Have you done this? (y/n): y
I need the CA credentials. Would you like to create a new CA key
and certificate now? (y/n): y
Please enter certificate owner's name or ID.
Usual format is first initial-last name (jdoe) or
hostname of server which will use this certificate.
All lower case, numbers OK.
Owner []: Mastering OpenVPN
```

File names will use Mastering_OpenVPN.

```
====> Creating private key with 2048 bits and generating request.
Do you want to password protect your CA private key? (y/n): y
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
Enter pass phrase for Mastering_OpenVPN.key:
Verifying - Enter pass phrase for Mastering_OpenVPN.key:
====> Self-Signing request.
Enter pass phrase for /Library/ssl-admin/Mastering_OpenVPN.key:
====> Moving certificate and key to appropriate directory.
====> Creating initial CRL.Using configuration from
/Library/ssl-admin/openssl.conf
Enter pass phrase for /Library/ssl-admin/active/ca.key:
ssl-admin installed Wed Oct 1 12:28:23 CDT 2014
OPTIONAL: I can't find your OpenVPN client config. Please copy
your config to
Libraryssl-admin/packages/client.ovpn
```

```
=====
# SSL-ADMIN v1.2.1 #
=====
Please enter the menu option from the following list:
1) Update run-time options:
   Key Duration (days): 3650
   Current Serial #: 01
   Key Size (bits): 2048
   Intermediate CA Signing: NO
2) Create new Certificate Request
3) Sign a Certificate Request
4) Perform a one-step request/sign
5) Revoke a Certificate
6) Renew/Re-sign a past Certificate Request
7) View current Certificate Revocation List
8) View index information for certificate.
i) Generate a user config with in-line certificates and keys.
z) Zip files for end user.
dh) Generate Diffie Hellman parameters.
CA) Create new Self-Signed CA certificate.
S) Create new Signed Server certificate.
q) Quit ssl-admin
```

Menu Item:

Как видите, ssl-admin значительно более многословен и интерактивен, чем EasyRSA. Кроме того, ssl-admin автоматически генерирует начальный CRL для вас.

Перед отображением меню было предупреждение OPTIONAL о конфигурации OpenVPN. Если вы предоставляете свою конфигурацию `client.ovpn` - `ssl-admin` может автоматически упаковывать конфигурационные файлы со встроенными сертификатами в ZIP-файл. Строки сертификата файла конфигурации должны быть универсальными:

```
ca ca.crt
cert client.crt
key client.key
```

Эти значения будут автоматически заменены встроенными ключами или файлы будут переименованы в соответствии с тем, как распространяются сертификаты, ключи и конфигурации OpenVPN. Теперь, когда мы инициализировали PKI путем создания пары ключей корневого центра сертификации - мы можем приступить к созданию сертификатов нашего сервера и клиента.

Серверные сертификаты OpenVPN

Сначала мы создадим сертификат для использования на сервере OpenVPN. Опция меню S генерирует CSR, ключ и предложит подписать сертификат центром сертификации:

```
Menu Item: S
Please enter certificate owner's name or ID.
Usual format is first initial-last name (jdoe) or
hostname of server which will use this certificate.
All lower case, numbers OK.
Owner []: Mastering OpenVPN Server
```

```
File names will use Mastering_OpenVPN_Server.
Please enter certificate owner's name or ID.
Usual format is first initial-last name (jdoe) or
hostname of server which will use this certificate.
All lower case, numbers OK.
Owner [Mastering_OpenVPN_Server]:
Would you like to password protect the private key (y/n): y
Generating a 2048 bit RSA private key
.....+++
.....++
writing new private key to 'Mastering_OpenVPN_Server.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
==> Serial Number = 01
Using configuration from /Library/ssl-admin/openssl.conf
Enter pass phrase for /Library/ssl-admin/active/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName          :PRINTABLE:'ZA'
stateOrProvinceName :PRINTABLE:'Enlightenment'
localityName         :PRINTABLE:'Overall'
organizationName    :PRINTABLE:'Mastering OpenVPN'
commonName           :PRINTABLE:'Mastering OpenVPN Server'
emailAddress         :IA5STRING:'root@example.org'
Certificate is to be certified until Sep 28 17:48:20 2024 GMT (3650
days)

Write out database with 1 new entries
Data Base Updated
=====> Moving certificates and keys to /Library/ssl-admin/active
```

```
for production.  
Can I move signing request (Mastering_OpenVPN_Server.csr) to the  
csr directory for archiving? (y/n): y  
==> Mastering_OpenVPN_Server.csr moved.  
MENU
```

Подсказка

Чтобы сэкономить место - меню, напечатанное `ssl-admin`, будет опущено, вместо этого оно будет заменено словом MENU.

В предыдущем коде мы использовали сертификат CN с пробелами для демонстрации поведения `ssl-admin`. Здесь он предупредил что пробелы будут заменены символом подчеркивания и дал пользователю возможность изменить CN при необходимости. Далее мы решили защитить приватный ключ парольной фразой. Наконец, пользователя спросили - можно ли заархивировать CSR.

Чтобы показать добавленные токены сервера - мы снова запускаем команду `openssl` для вывода сведений о сертификате. Следующий вывод опускает некоторые детали ключей для краткости:

```
root@computer:/Library/ssl-admin-> openssl x509 -noout -text -in  
active/Mastering_OpenVPN_Server.crt  
Certificate:  
    Data:  
        Version: 3 (0x2)  
        Serial Number: 1 (0x1)  
        Signature Algorithm: sha1WithRSAEncryption  
        Issuer: C=ZA, ST=Enlightenment, L=Overall, O=Mastering  
OpenVPN, CN=Mastering OpenVPN/emailAddress=root@example.org  
        Validity  
            Not Before: Oct 1 17:48:20 2014 GMT  
            Not After : Sep 28 17:48:20 2024 GMT  
        Subject: C=ZA, ST=Enlightenment, O=Mastering OpenVPN,  
CN=Mastering OpenVPN Server/emailAddress=root@example.org  
        Subject Public Key Info:  
            Public Key Algorithm: rsaEncryption  
            RSA Public Key: (2048 bit)  
                Modulus (2048 bit):  
                    ...  
                Exponent: 65537 (0x10001)  
X509v3 extensions:  
    X509v3 Basic Constraints:  
        CA:FALSE  
    Netscape Cert Type:  
        SSL Server  
    Netscape Comment:  
        ssl-admin (OpenSSL) Generated Server Certificate  
X509v3 Subject Key Identifier:  
  
FB:A8:91:01:E3:51:5D:A7:29:8C:54:63:9F:22:7F:F8:DE:AB:5A:39  
X509v3 Authority Key Identifier:  
  
keyid:1F:85:DF:90:5C:3F:73:A9:03:B9:F4:E6:C2:2C:A3:27:CF:5B:44:95  
  
DirName:/C=ZA/ST=Enlightenment/L=Overall/O=Mastering  
OpenVPN/CN=Mastering OpenVPN/emailAddress=root@example.org  
    serial:D2:93:32:F0:8E:BC:58:EE  
  
X509v3 Extended Key Usage:  
    TLS Web Server Authentication
```

X509v3 Key Usage:
Digital Signature, Key Encipherment
Signature Algorithm: sha1WithRSAEncryption

Обратите внимание, что x509v3 Extended Key Usage включает TLS Web Server Authentication. Более старый стандарт nsCertType, известный как Netscape Cert Type, также включен для обратной совместимости. Это не столько уместно для OpenVPN, сколько ssl-admin был написан как общая утилита управления CA x509.

Клиентские сертификаты OpenVPN

Клиентские сертификаты генерируются так же, как сертификат сервера. Опция 4 в меню создаст запрос на подпись сертификата (CSR) и впоследствии подпишет CSR:

```
Menu Item: 4
Please enter certificate owner's name or ID.
Usual format is first initial-last name (jdoe) or
hostname of server which will use this certificate.
All lower case, numbers OK.
Owner []: client1

File names will use client1.
Please enter certificate owner's name or ID.
Usual format is first initial-last name (jdoe) or
hostname of server which will use this certificate.
All lower case, numbers OK.
Owner [client1]:
Would you like to password protect the private key (y/n): n
Generating a 2048 bit RSA private key
.....
.....
+++
.....+++
writing new private key to 'client1.key'
-----
==> Serial Number = 02
=====> Signing request for client1
Using configuration from /Library/ssl-admin/openssl.conf
Enter pass phrase for /Library/ssl-admin/active/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName          :PRINTABLE:'ZA'
stateOrProvinceName :PRINTABLE:'Enlightenment'
localityName         :PRINTABLE:'Overall'
organizationName    :PRINTABLE:'Mastering OpenVPN'
commonName           :PRINTABLE:'client1'
emailAddress         :IA5STRING:'root@example.org'
Certificate is to be certified until Sep 28 18:05:14 2024 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
=====> Moving certificates and keys to Libraryssl-admin/active
for production.
Can I move signing request (client1.csr) to the csr directory for
archiving? (y/n): ==> client1.csr moved.
MENU
```

Подсказка

В последующих упражнениях будет использоваться до трех клиентских сертификатов - поэтому рекомендуется повторить предыдущие шаги для client2 и client3.

Используя двоичный файл `openssl` для проверки сертификата мы видим, что в сертификате `client1` отсутствуют расширения использования ключей сервера, которые присутствовали в сертификате сервера, который мы создали ранее.

```
root@computer:Libraryssl-admin-> openssl x509 -noout -text -in
active/client1.crt
Certificate:
Data:
Version: 1 (0x0)
Serial Number: 2 (0x2)
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=ZA, ST=Enlightenment, L=Overall, O=Mastering
OpenVPN, CN=Mastering OpenVPN/emailAddress=root@example.org
Validity
    Not Before: Oct 1 18:05:14 2014 GMT
    Not After : Sep 28 18:05:14 2024 GMT
Subject: C=ZA, ST=Enlightenment, O=Mastering OpenVPN,
CN=client1/emailAddress=root@example.org
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (2048 bit)
        Modulus (2048 bit):
        ...
        Exponent: 65537 (0x10001)
Signature Algorithm: sha1WithRSAEncryption
...
```

Этот сертификат, очевидно, имеет более простую структуру, чем сертификат сервера, и параметры использования ключа сервера отсутствуют.

После создания нашего ЦС, серверного и трех клиентских сертификатов у нас остается следующая структура каталогов:

```
root@computer:Libraryssl-admin-> ls -lrth
total 16
-rw-rw-r-- 1 root wheel 541B Oct 1 12:22 ssl-admin.conf
drwxr-x--- 2 root wheel 68B Oct 1 12:24 revoked
-rw-rw---- 1 root wheel 2.5K Oct 1 12:27 openssl.conf
drwxr-x--- 2 root wheel 102B Oct 1 12:28 packages
-r--r--r-- 1 root wheel 531B Oct 1 12:43 ssl-admin.conf.sample
-r--r--r-- 1 root wheel 2.5K Oct 1 12:43 openssl.conf.sample
drwxr-x--- 2 root wheel 340B Oct 1 13:05 prog
drwxr-x--- 2 root wheel 340B Oct 1 13:05 csr
drwxr-x--- 2 root wheel 544B Oct 1 13:05 active
```

Активный (`active`) каталог содержит все сертификаты и ключи, которые не были отозваны, включая сертификат и ключ ЦС. Когда сертификаты отозваны - они перемещаются из `active` в `revoked` (аннулированные). Чтобы использовать утилиты OpenSSL для отзыва сертификата - он должен присутствовать. Без сертификата возможно проблематичное редактирование индексов вручную в файле `index.txt`. Как следует из его названия, каталог `csr` содержит все CSR. Как правило, их можно безопасно удалить и они хранятся только для устранения неполадок или для восстановления сертификата.

Администратору предлагается оставить управление содержимым хранилища сертификатов утилите. Это относится как к `ssl-admin`, так и к Easy-RSA.

Каталог `prog` содержит рабочие файлы OpenSSL и последний CRL. Не рекомендуется, чтобы эти файлы были повреждены, так как есть вероятность сделать вашу PKI непригодной для использования в случае допуска ошибок.

Наконец, каталог `packages` будет содержать все файлы, которые вы можете распространять среди своих конечных пользователей: не только клиентов OpenVPN, но и администраторов веб-серверов и так далее. Упаковка сертификатов и ключей гарантирует, что конечный пользователь получит все необходимые файлы и они будут в правильном формате.

Другие преимущества

Утилита `ssl-admin` имеет некоторые другие функции, которые могут заинтересовать кого-то, кто управляет PKI. Индекс доступный для поиска (опция 8), который показывает статус данного сертификата. Также возможно отображение текущего CRL (опция 7). `ssl-admin` способен упаковывать файлы конфигурации OpenVPN с сертификатами для пользователей как во встроенным формате (опция `i`), так и в отдельные файлы - все они содержатся в `zip`-файле (опция `z`). Эти два последних варианта будут обсуждаться далее в этой книге, поскольку мы генерируем конфигурации сервера и клиента.

Несколько CA и CRL

Easy-RSA 3.0 довольно легко поддерживает несколько корневых ЦС. Создав отдельный каталог ЦС в корневом каталоге `EASYRSA` и имея разные файлы `vars` для каждого - можно управлять каждым отдельным ЦС с помощью Easy-RSA.

В настоящее время `ssl-admin` не поддерживает несколько корневых ЦС, но поддерживается создание промежуточных CA.

В OpenVPN один экземпляр сервера может поддерживать несколько корневых ЦС, при этом принимаются клиентские подключения, подписанные любым ЦС. Чтобы включить такую поддержку - сертификат ЦС для каждого авторизованного ЦС должен быть объединен в один файл, который можно вызвать с помощью опции OpenVPN `--ca`. То же самое можно сделать со списком отзыва сертификатов.

Как правило, не рекомендуется использовать несколько сертификатов ЦС для одного экземпляра OpenVPN; исключениями могут оказаться миграция сервера или центра сертификации, приобретение компании или организации и т.д.

Ни при каких обстоятельствах не стоит использовать корневой центр сертификации веб-браузера для цепочки сертификатов OpenVPN. Невозможно определить у кого есть сертификат, и любой, кто попадает в эту иерархию ЦС, сможет подключиться к вашему экземпляру VPN.

В дальнейшем планируется объединить проекты `ssl-admin` и Easy-RSA в единый полностью функциональный пакет управления PKI. Для Easy-RSA 4.0 необходимо реализовать эти миграции, используя лучшие функции обеих утилит.

Дополнительная безопасность - аппаратные токены, смарт-карты и PKCS #11

В этом разделе мы предоставим некоторую справочную информацию о криптографических аппаратных устройствах. Вы узнаете как сгенерировать приватный ключ на аппаратном токене и как скопировать связанный сертификат X.509 в токен.

После этого мы обсудим, как OpenVPN может найти и использовать эту пару сертификат/приватный ключ для установления VPN-соединения.

Исходная информация

Начиная с версии 2.1, OpenVPN поддерживает двухфакторную аутентификацию, предоставляя поддержку **PKCS#11**. Двухфакторная аутентификация основана на идее, что для использования

системы (например, VPN) необходимо предоставить две вещи:

- Что-то, что Вы **знаете**, например, пароль
- Что-то, чем Вы **владеете**, например, смарт-карта или аппаратный токен

PKCS#11 является отраслевым стандартом для связи со смарт-картами или аппаратными токенами, и здесь доступны как открытые, так и коммерческие драйверы. Стандарт PKCS#11 был первоначально опубликован RSA Laboratories и иногда также упоминается как стандарт **cryptoki**, что означает **CRYPtographic TOKen Interface**.

Помимо терминов аппаратный токен и смарт-карта, термин **аппаратный модуль безопасности (Hardware Security Module - HSM)** также часто используется для двухфакторной аутентификации. В этом разделе мы в основном будем использовать термин аппаратный токен. Основное различие между аппаратными токенами и смарт-картами заключается в форм-факторе: аппаратный токен обычно поставляется как устройство USB, тогда как смарт-карта выглядит как карта банкомата или кредитная карта. Для использования смарт-карты требуется специальный картридер, который иногда интегрируется в ноутбуки и даже некоторые настольные компьютеры. В некоторых странах выпускаются национальные электронные удостоверения личности, которые обычно классифицируются как смарт-карты.

Чаще всего HSM - это устройство, способное безопасно хранить криптографические ключи и управлять ими, часто обеспечивающее аппаратное ускорение, а также ускоряющее шифрование и дешифрование.

Аппаратный токен, смарт-карта или HSM, обычно представляют собой небольшое устройство со встроенным чипом. Этот встроенный чип работает под управлением миниатюрной операционной системы (часто называемой Card OS), отвечающей за безопасное создание, хранение и управление закрытыми ключами SSL. Большинство аппаратных токенов также способны хранить другую информацию, такую как сертификаты SSL, так что действительная пара сертификат/приватный ключ может надежно храниться на одном устройстве.

Поддерживаемые платформы

Основная сложность при использовании двухфакторной аутентификации заключается в поддержке программного обеспечения на разных платформах. Хотя большинство поставщиков аппаратных токенов и смарт-карт предоставляют драйверы операционной системы для Microsoft Windows, Linux или даже Mac OS X поддерживается гораздо меньше плат и токенов. Обратите внимание, что это не связано с самим OpenVPN: если определенный аппаратный токен поддерживается используемой операционной системой и драйвером PKCS#11, то, как правило, OpenVPN может использовать этот аппаратный токен или смарт-карту.

Для этой книги мы использовали аппаратный токен Aladdin eToken Pro 72K USB. (Aladdin Systems была куплена SafeNet (<http://www.safenet.com>). Однако, только недавно SafeNet объединилась с Gemalto.) Этот аппаратный токен поддерживается только с использованием клиента аутентификации SafeNet с закрытым исходным кодом, доступным для Microsoft Windows, Mac OS X и Linux.

Более ранние версии этих токенов имели преимущество в том, что могли использовать либо платный драйвер с закрытым исходным кодом от SafeNet, либо бесплатный драйвер с открытым исходным кодом из проекта OpenSC (в настоящее время можно найти по адресу <https://github.com/OpenSC/OpenSC/wiki>). К сожалению, эти старые токены больше не могут быть приобретены и текущие токены от SafeNet используют другую Card OS, которая не поддерживается OpenSC.

Общий процесс и концепция применимы к большинству аппаратных токенов, которые вы можете использовать. Эти старые устройства использовались просто для тестирования и

демонстрации. Многие поставщики используют мобильное приложение на смартфоне пользователя в качестве аппаратного токена. SafeNet также имеет такой продукт: MobilePASS.

Кроме того, драйверы и инструменты из проекта OpenSC не так зрелы, как программное обеспечение от коммерческих поставщиков программного обеспечения.

Другие производители смарт-карт и аппаратных токенов - Aktiv Co и Feitian (с поддержкой открытого исходного кода).

Обратите внимание, что OpenVPN зависит исключительно от работающего драйвера PKCS#11. При выборе аппаратного токена будет важно проверить - поддерживается ли устройство на необходимых платформах, а не будет ли оно работать с самим OpenVPN. Кроме того - обратите внимание что не требуется (или даже не рекомендуется!) использовать аппаратный токен на сервере OpenVPN.

Инициализация аппаратного токена

Предполагается, что пакет Aladdin pkiclient или SafeNet AuthenticationClient установлен и аппаратный токен распознается программным обеспечением драйвера. Если eToken уже был инициализирован - пропустите этот шаг.

Сначала откройте окно свойств клиента eToken и нажмите **Initialize eToken**. Это вызовет следующее диалоговое окно:



Введите пароль токена и пароль администратора, снимите флажок **Token Password must be changed on first logon** и нажмите **Start**.

Все содержимое токена теперь будет уничтожено, а eToken будет инициализирован с использованием нового токена и пароля администратора.

Генерация пары сертификат/приватный ключ

При использовании аппаратного токена процесс генерации пары сертификата и приватного ключа немного отличается от использования инструментов ssl-admin или Easy-RSA. С помощью

ssl-admin или Easy-tools приватный ключ, запрос сертификата и сертификат X.509 генерируются за один шаг. С аппаратным токеном нам сначала нужно сгенерировать приватный ключ на токене.

Используя этот приватный ключ, нам нужно создать CSR. Этот CSR затем подписывается ЦС что приводит к сертификату X.509. Этот сертификат затем записывается обратно в токен. Инструменты ssl-admin и Easy-RSA фактически следуют одному и тому же процессу, но они скрывают файл CSR от пользователя.

Генерация приватного ключа на токене

Чтобы сгенерировать приватный ключ на eToken - нам нужна команда `pkcs11-tool`, которая является частью пакета OpenSC. Пакет OpenSC доступен для Microsoft Windows, Mac OS X и Linux. Команда `pkcs11-tool` предоставляет интерфейс для аппаратных токенов с использованием драйвера PKCS#11. Драйвером PKCS#11, включенным в программное обеспечение драйвера Aladdin/SafeNet, является `libeTPkcs11.so` (Linux и Mac OS X) или `eTPkcs11.dll` (Windows). Следующая команда была применена на 64-битной Linux-машине и генерирует 2048-битный ключ RSA, идентифицированный с помощью метки `movpn` и ID `20141001`. Поскольку мы генерируем приватный ключ - необходимо войти в систему токена:

```
# pkcs11-tool --module libeTPkcs11.so \
  --keypairgen --key-type rsa:2048 \
  --label "movpn" --id 20141001 --login
Using slot 0 with a present token (0x0)
Logging in to "Mastering OpenVPN".
Please enter User PIN: [enter Token password]
Key pair generated:
Private Key Object; RSA
  label:      movpn
  ID:        20141001
  Usage:     decrypt, sign, unwrap
Public Key Object; RSA 2048 bits
  label:      movpn
  ID:        20141001
  Usage:     encrypt, verify, wrap
```

Для генерации 2048-битного ключа на аппаратном токене потребуется некоторое время, в течение которого красный индикатор на аппаратном токене выключен. После этого индикатор снова включится.

Выходные данные вышеприведенной команды сообщают нам что сгенерирован приватный ключ RSA вместе с публичным 2048-битным открытым ключом RSA. Это не то же самое, что сертификат SSL. Чтобы сгенерировать сертификат SSL или X.509, нам сначала нужно сгенерировать запрос сертификата.

Генерация запроса на сертификат

Нам необходимо использовать движок OpenSSL `engine_pkcs11` для генерации запроса сертификата с использованием приватного ключа из аппаратного токена. Механизм `engine_pkcs11` лучше всего использовать с пользовательским файлом `openssl.cnf`. Сначала мы создаем этот файл:

```
openssl_conf = openssl_def

[ openssl_def ]
engines = engine_section

[ engine_section ]
pkcs11 = pkcs11_section
```

```

[ pkcs11_section ]
engine_id = pkcs11
dynamic_path = /usr/lib64/openssl/engines/engine_pkcs11.so
MODULE_PATH = /usr/lib64/libeTPkcs11.so
init = 0

[ req ]
distinguished_name = req_distinguished_name

[ req_distinguished_name ]

```

Этот файл был сгенерирован для 64-битной системы CentOS Linux. В других системах пути и имена драйверов будут другими. Обратите внимание - операторы в файле openssl.cnf чувствительны к регистру!

Теперь мы генерируем запрос сертификата с субъектом /CN=movpn, используя следующую команду openssl:

```

$ openssl req -engine pkcs11 -keyform engine -key 20141001 \
-new -text -out movpn.csr -config openssl.cnf \
-subj "/CN=movpn"
engine "pkcs11" set.
PKCS#11 token PIN: [enter Token password]

```

Команда не производит дальнейших выходных данных в случае успеха. Теперь должен появиться файл movpn.csr, который должен быть подписан центром сертификации, созданным ранее в этой главе. Предполагается, что подписанный сертификат будет называться movpn.crt.

Запись сертификата X.509 в токен

OpenVPN ожидает что сертификат X.509 будет присутствовать на аппаратном токене. Поэтому сперва мы должны записать сертификат X.509 из предыдущего шага в eToken.

Сначала преобразуйте подписанный сертификат в формат DER:

```
$ openssl x509 -in movpn.crt -outform der -out movpn.der
```

Далее мы записываем файл DER в токен:

```

$ pkcs11-tool --module libeTPkcs11.so \
--write-object movpn.der --type cert \
--label movpn --id 20141001 --login
Using slot 0 with a present token (0x0)
Logging in to "Mastering OpenVPN".
Please enter User PIN: [enter Token password]
Created certificate:
Certificate Object, type = X.509 cert
label:      movpn
ID:        20141001

```

Нам необходимо убедиться что идентификаторы приватного ключа и сертификата совпадают:

```

$ pkcs11-tool --module libeTPkcs11.so --login -0
Using slot 0 with a present token (0x0)
Logging in to "Mastering OpenVPN".
Please enter User PIN: [enter Token password]
Private Key Object; RSA
label:      movpn
ID:        20141001
Usage:      decrypt, sign, unwrap
Public Key Object; RSA 2048 bits
label:      movpn
ID:        20141001

```

```
Usage:      encrypt, verify, wrap
Certificate Object, type = X.509 cert
label:      movpn
ID:        20141001
```

Теперь токен готов к использованию с OpenVPN.

Получение идентификатора аппаратного токена

Чтобы использовать сертификат и приватный ключ от аппаратного токена - вы должны сначала узнать идентификатор аппаратного токена, ожидаемый OpenVPN. Это делается с помощью опции --show-pkcs11-id:

```
$ openvpn --show-pkcs11-ids /usr/lib64/libeTPkcs11.so
The following objects are available for use.
Each object shown below may be used as parameter to
--pkcs11-id option please remember to use single quote mark.
```

```
Certificate
    DN:           CN=movpn
    Serial:       01
    Serialized id:
SafeNet\x20Inc\x2E/eToken/00a3659e/Mastering\x20openVPN/20141001
```

Серийный идентификатор состоит из следующего:

- Имя драйвера PKCS#11 (SafeNet Inc)
- Название продукта (eToken)
- Серийный номер токена (00a3659e)
- Имя токена (Mastering OpenVPN)
- Идентификатор сертификата и приватный ключ на токене (20141001)

Метка сертификата или приватного ключа не используется, но рекомендуется также синхронизировать их друг с другом.

Использование аппаратного токена с OpenVPN

Теперь мы наконец готовы использовать аппаратный токен в OpenVPN. Чтобы использовать его - мы заменим строки в файле конфигурации OpenVPN:

```
cert myclient.crt
key myclient.key
```

Параметрами pkcs11-provider и pkcs11-id:

```
pkcs11-providers /usr/lib64/libeTPkcs11.so
pkcs11-id
'SafeNet\x20Inc\x2E/eToken/00a3659e/Mastering\x20openVPN/20141001'
```

Резюме

В этой главе мы обсудили инструменты и методы для создания корневого центра сертификации, а также базовых сертификатов сервера и клиента. Кроме того, была рассмотрена концепция PKCS#11, хотя базовая технология постоянно развивается. Теперь у вас должна быть полная PKI и инструменты для ее расширения.

В следующей главе будет представлена настройка маршрутизируемого VPN. Также будет обсуждаться использование сетевого устройства tun и требования уровня 3 для работающего соединения.

Глава 4. Режим клиент-сервер с устройствами tun

Наиболее часто используемая модель развертывания для OpenVPN - это один сервер с несколькими удаленными клиентами, способными маршрутизировать IP-трафик. Мы называем эту модель развертывания *режимом клиент-сервер с устройствами tun*.

В этой главе мы начнем с базовой настройки клиент-сервер. По мере продвижения мы добавим больше возможностей и в конце этой главы приведены некоторые продвинутые примеры того, как настроить OpenVPN в режиме клиент-сервер. В следующей главе мы объясним, как интегрировать настройку клиент-серверного режима на основе tun в существующую настройку сети, включая такие темы, как общий доступ к файлам Windows и маршрутизацию на основе политик.

В этой главе будут рассмотрены следующие темы:

- Настройка инфраструктуры публичных ключей
- Начальная настройка режима клиент-сервер
- Добавление усиленной безопасности с помощью файлов конфигурации продакшен-уровня
- Маршрутизация и маршрутизация на стороне сервера
- Специфичная для клиента конфигурация с использованием файлов CCD
- Клиентская маршрутизация
- Перенаправление шлюза по умолчанию
- Файл статуса OpenVPN
- Интерфейс управления OpenVPN
- Пересогласование ключей сеанса
- Использование IPv6
- Прокси ARP
- Раздача публичных IP-адресов

Понимание режима клиент-сервер

Клиент-серверный режим был впервые представлен в OpenVPN 2.0. В этом режиме сервер представляет собой один процесс OpenVPN, к которому могут подключаться несколько клиентов. Каждому аутентифицированному и авторизованному клиенту назначается IP-адрес из пула адресов, управляемого сервером OpenVPN. Клиенты не могут общаться напрямую друг с другом. Весь трафик проходит через сервер - что имеет как преимущества, так и недостатки.

Преимущества заключаются в следующем:

- Контроль. Администратор VPN-сервера может контролировать, какой трафик может передаваться между клиентами. По умолчанию трафик между клиентами не разрешен. Однако, используя либо опцию `client-to-client` OpenVPN, либо используя умный брандмауэр и правила маршрутизации, можно разрешить клиентам обмениваться данными друг с другом.
- Простота развертывания: гораздо проще настроить один сервер, к которому могут

обращаться несколько разных клиентов, чем обеспечить связь между множеством клиентов, каждый из которых имеет собственную сеть и конфигурации брандмауэра.

Недостатки заключаются в следующем:

- **Масштабируемость.** Поскольку весь трафик передается от клиента к серверу (и наоборот) - сервер может быстро стать узким местом в крупномасштабных установках VPN.
- **Производительность:** поскольку весь трафик между двумя клиентами (клиентами А и В) должен проходить от клиента А к серверу, а затем от сервера к клиенту В - производительность этого типа VPN всегда будет ниже по сравнению с прямым соединением клиент-клиент.

Наиболее распространенным сценарием развертывания для этого режима является корпоративный сервер OpenVPN, к которому подключаются различные VPN-клиенты. Клиентами могут быть филиалы, работники на выезде, работающие из дома, а также пользователи смартфонов и планшетов.

Эта модель развертывания покрывает 95 процентов типичных требований для VPN и предпочтительнее сложных установок с использованием расширенных функций, таких как мостовое соединение. Только если существуют конкретные требования для маршрутизации трафика, не относящегося к IP (например, устаревший трафик IPX), или если необходимо сформировать единый сетевой широковещательный домен, такой модели развертывания будет недостаточно.

Настройка инфраструктуры открытых ключей

В режиме клиент-сервер OpenVPN настраивается с помощью инфраструктуры открытых ключей (PKI) с сертификатами X.509 и приватными ключами. Прежде чем мы сможем настроить клиент-сервер VPN - нам необходимо сначала настроить эту PKI. PKI состоит из CA, приватных ключей и сертификатов (публичных ключей) как для клиента, так и для сервера. В [Главе 3](#), *PKI и сертификаты* мы подробно обсудили как настроить такую PKI. Эта глава основана на сертификатах и ключах, сгенерированных в той главе.

Сначала мы копируем сертификат и ключи в отдельное место. Как правило, рекомендуется хранить файлы PKI в отдельном месте, а если возможно даже на отдельном компьютере. Особое внимание следует уделить защите файла ca.key, поскольку вся безопасность вашей PKI зависит от этого файла. Если файл ca.key скомпрометирован каким-либо образом - вся PKI становится небезопасной и должна быть удалена. В следующих командах предполагается, что файлы PKI генерируются с использованием ssldadmin и хранятся в каталоге <PKI_DIR>, где <PKI_DIR> представляет собой реальный каталог в системе. Выполните следующие команды, чтобы скопировать необходимые файлы PKI для сервера:

```
[root@server] # mkdir -p /etc/openvpn/movpn
[root@server] # chmod 700 /etc/openvpn/movpn
[root@server] # cd /etc/openvpn/movpn
[root@server] # PKI=<PKI_DIR>/ssladm/active
[root@server] # cp -a $PKI/ca.crt movpn-ca.crt
[root@server] # cp -a $PKI/Mastering_OpenVPN_Server.crt server.crt
[root@server] # cp -a $PKI/Mastering_OpenVPN_Server.key server.key
```

Нам также необходимо создать файл параметров **Диффи-Хеллмана (DH)**, который необходим для ключей сеанса VPN. Ключи сеанса являются эфемерными или временными и генерируются при первой настройке соединения между клиентом и сервером. Для обеспечения оптимальной безопасности эфемерные ключи регенерируются во время сеанса через фиксированные интервалы. Интервал регенерации ключей по умолчанию для OpenVPN составляет один час, но его можно настроить с помощью различных опций OpenVPN. Это будет объяснено позже в этой

главе в разделе [Пересогласование ключей сеанса](#).

Чтобы создать файл параметров DH, выполните следующие команды:

```
[root@server] # cd /etc/openvpn/movpn  
[root@server] # openssl dhparam -out dh2048.pem 2048  
Generating DH parameters, 2048 bit long safe prime, generator 2  
This is going to take a long time  
.....+.....  
.....  
.....+.....  
.....
```

В этом примере мы выбираем размер ключа DH в 2048 бит, что является рекомендуемым. Вы также можете использовать ключи DH большего размера, но это замедлит начальный процесс подключения для каждого клиента OpenVPN. Теперь мы готовы установить и запустить сервер OpenVPN.

Начальная настройка режима клиент-сервер

Чтобы настроить базовый сервер OpenVPN, мы сначала создаем файл конфигурации сервера, используя следующие шаги:

1. Создайте следующий файл

```
proto udp  
port 1194  
dev tun  
server 10.200.0.0 255.255.255.0  
topology subnet  
persist-key  
persist-tun  
keepalive 10 60  
  
dh    /etc/openvpn/movpn/dh2048.pem  
ca    /etc/openvpn/movpn/movpn-ca.crt  
cert  /etc/openvpn/movpn/server.crt  
key   /etc/openvpn/movpn/server.key  
  
user nobody  
group nobody # используйте 'group nogroup' для Debian/Ubuntu  
verb 3  
daemon  
log-append /var/log/openvpn.log
```

2. Затем сохраните его как `movpn-04-01-server.conf`. Подробное объяснение каждой из строк конфигурации будет дано позже.
3. Запустите сервер OpenVPN:

```
[root@server]# openvpn --config movpn-04-01-server.conf
```

4. Команда не дает никакого вывода в командной строке, так как весь вывод перенаправляется в файл журнала `/var/log/openvpn.log`. Проверьте этот файл для просмотра журнала запуска OpenVPN:

```
OpenVPN 2.3.2 x86_64-redhat-linux-gnu [SSL (OpenSSL)] [LZO]  
[EPOLL] [PKCS11] [eurephia] [MH] [IPv6] built on Sep 12 2013  
Enter Private Key Password:  
WARNING: this configuration may cache passwords in memory --  
use the auth-nocache option to prevent this  
TUN/TAP device tun0 opened
```

```

do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
/sbin/ip link set dev tun0 up mtu 1500
/sbin/ip addr add dev tun0 10.200.0.1/24 broadcast 10.200.0.255
GID set to nobody
UID set to nobody
UDPV4 link local (bound): [undef]
UDPV4 link remote: [undef]
Initialization Sequence Completed

```

5. Обратите внимание, что обычно каждая запись в файле журнала начинается с отметки времени. Для ясности эта временная метка была удалена.
6. Затем создайте файл конфигурации клиента:

```

client
proto udp
remote openvpnserver.example.com
port 1194
dev tun
nobind
ca    /etc/openvpn/movpn/movpn-ca.crt
cert  /etc/openvpn/movpn/client1.crt
key   /etc/openvpn/movpn/client1.key

```

Сохраните его как movpn-04-01-client.conf.

7. Передайте файлы PKI клиенту, используя безопасный канал, например с помощью команды scp:

```

[root@client]# mkdir -p /etc/openvpn/movpn
[root@client]# chmod 700 /etc/openvpn/movpn
[root@client]# cd /etc/openvpn/movpn
[root@client]# PKI_HOST=openvpnserver.example.com
[root@client]# PKI=<PKI_DIR>/ssladmin/active
[root@client]# scp root@$PKI_HOST:$PKI/ca.crt movpn-ca.crt
[root@client]# scp root@$PKI_HOST:$PKI/client1.crt client1.crt
[root@client]# scp root@$PKI_HOST:$PKI/client1.key client1.key

```

8. Запустите клиент OpenVPN:

```

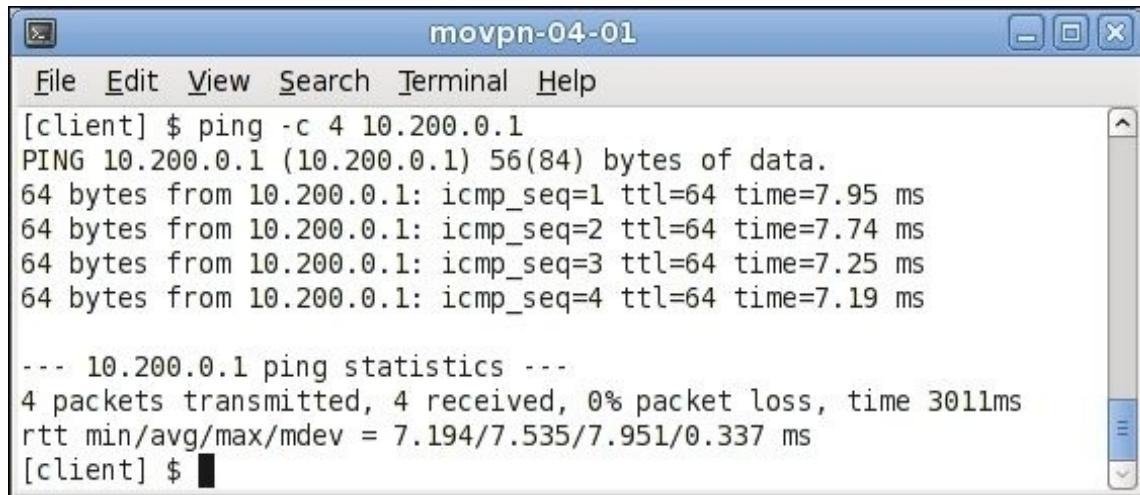
[root@client]# openvpn --config movpn-04-01-client.conf --
suppress-timestamps
OpenVPN 2.3.2 x86_64-redhat-linux-gnu [SSL (OpenSSL)] [LZO]
[EPOLL] [PKCS11] [eurephia] [MH] [IPv6] built on Sep 12 2013
WARNING: No server certificate verification method has been
enabled. See http://openvpn.net/howto.html#mitm for more info.
UDPV4 link local: [undef]
UDPV4 link remote: [AF_INET]openvpnserver:1194
[Mastering OpenVPN Server] Peer Connection Initiated with
[AF_INET]openvpnserver:1194
TUN/TAP device tun0 opened
do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
/sbin/ip link set dev tun0 up mtu 1500
/sbin/ip addr add dev tun0 10.200.0.2/24 broadcast 10.200.0.255
Initialization Sequence Completed

```

9. Временные метки снова отсутствуют, но на этот раз они подавляются с помощью опции OpenVPN suppress-timestamps, как указано в командной строке.
10. После установления соединения проверьте следующее сообщение:

Initialization Sequence Completed

11. Вы можете убедиться, что соединение работает правильно, проверив VPN-адрес сервера:



```
[client] $ ping -c 4 10.200.0.1
PING 10.200.0.1 (10.200.0.1) 56(84) bytes of data.
64 bytes from 10.200.0.1: icmp_seq=1 ttl=64 time=7.95 ms
64 bytes from 10.200.0.1: icmp_seq=2 ttl=64 time=7.74 ms
64 bytes from 10.200.0.1: icmp_seq=3 ttl=64 time=7.25 ms
64 bytes from 10.200.0.1: icmp_seq=4 ttl=64 time=7.19 ms

--- 10.200.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3011ms
rtt min/avg/max/mdev = 7.194/7.535/7.951/0.337 ms
[client] $
```

Подробное объяснение файлов конфигурации

Так как это первый пример клиент-серверной конфигурации - приведено подробное объяснение файлов конфигурации как сервера, так и клиента. Файл конфигурации сервера содержит следующие строки:

- **proto udp**: хотя это протокол по умолчанию, разумно явно указать его в файле конфигурации во избежание путаницы.
- **port 1194**: локальный порт, который будет прослушивать OpenVPN. Значение по умолчанию - 1194, но можно использовать любой допустимый и доступный номер порта.
- **dev tun**: указывает имя устройства tun, которое будет использоваться для сервера. Не добавляя номер после tun, мы инструктируем OpenVPN открыть новое устройство tun. Этому новому устройству будет присвоен первый доступный номер в ядре системы, начиная с 0 (tun0, tun1, tun2 и т.д.). Для серверов Windows желательно сохранить эту строку как есть. Если необходимо использовать определенное устройство Windows, то потребуется опция **dev-node**.
- **server 10.200.0.0 255.255.255.0**: оператор **server** переводит OpenVPN в режим сервера. Адрес подсети и маска определяют подсеть и маску, которые будут использоваться для VPN-сервера и клиентов. Серверу VPN назначается первый адрес, который в данном случае равен 10.200.0.1. Первому клиенту будет назначен адрес 10.200.0.2 (потому что мы используем **topology subnet**). Оператор **server** для этой конфигурации внутренне расширяется следующим образом:

```
mode server
tls-server
push "topology subnet"

ifconfig 10.200.0.1 255.255.255.0
ifconfig-pool 10.200.0.2 10.200.0.254 255.255.255.0
push "route-gateway 10.200.0.1"
```

Это взято со страницы руководства OpenVPN по адресу <https://community.openvpn.net/openvpn/wiki/Openvpn23ManPage>. Если эти строки конфигурации используются вместо макроса сервера - используется та же конфигурация.

Заметка

Расширение включает push "topology subnet", потому что мы также указали topology subnet в файле конфигурации. Без этой строки расширение не произошло бы.

- **topology subnet**: определяет топологию для VPN. Текущей топологией по умолчанию является net30, в которой серверу и каждому клиенту назначено отдельное миниатюрное пространство подсети /30. Более подробная информация об использовании *topology subnet* против *topology net30* приведена в следующем разделе.
- **persist-tun** и **persist-key**: дает указание OpenVPN не открывать повторно устройство tun и не генерировать новый ключ при каждом перезапуске туннеля. Эти параметры особенно полезны в сочетании с user nobody, так как обычно у nobody нет прав доступа для открытия нового интерфейса tun.
- **keepalive 10 60**: используется для проверки работоспособности VPN-соединения, даже если по туннелю нет трафика. Оператор **keepalive** - это макрос для команд **ping** и **ping-restart**. Оператор **keepalive 10 60** в конфигурации на стороне сервера расширяется до:

```
ping 10
ping-restart 120
push "ping 10"
push "ping-restart 60"
```

Предыдущий код означает:

- Отправка пинг-сообщений каждому клиенту каждые 10 секунд.
- Перезапуск соединения, если клиент не отвечает в течение 120 секунд ($2 * 60 = 120$)
- Передача инструкций **ping 10** и **ping-restart 60** каждому клиенту
- **dh <путь к файлу Диффи-Хеллмана>**: указывает путь к файлу DH, который требуется для сервера OpenVPN. Без этого файла сервер не может установить безопасное соединение TLS с клиентами. Рекомендуется использовать абсолютный путь для этого файла (а также к другим сертификатам и секретным ключам).
- **ca <путь к файлу CA>**: указывает путь к файлу CA. Файл CA должен содержать сертификат CA (или даже набор сертификатов), который использовался для подписи **клиентских сертификатов**. Это необязательно должен быть тот же CA, который использовался для подписи сертификата сервера, хотя в нашей настройке PKI мы использовали тот же CA. Рекомендуется использовать абсолютный путь для этого файла (а также к другим сертификатам и секретным ключам).
- **cert <путь к файлу сертификата X.509>**: указывает путь к файлу публичного сертификата X.509 сервера. Этот сертификат необходим серверу OpenVPN даже если клиенты подключаются без использования сертификатов. Рекомендуется использовать абсолютный путь для этого файла (а также к другим сертификатам и секретным ключам).
- **key <путь к файлу приватного ключа>**: указывает путь к файлу приватного ключа сервера. Этот файл приватного ключа необходим серверу OpenVPN, даже если клиенты подключаются без использования сертификатов или приватных ключей. Этот файл должен быть доступен для чтения только пользователю root (или администратору), так как любой пользователь, имеющий доступ на чтение приватных ключей, может расшифровать трафик OpenVPN. Обратите внимание - OpenVPN прочтет этот файл перед удалением пользовательских привилегий. Рекомендуется использовать абсолютный путь для этого файла (а также пути к другим сертификатам и приватным ключам).

- `user nobody` и `group nobody`: дает указание OpenVPN перейти на пользователя Unix `nobody` и группу `nobody` после установления соединения. Это дополнительно повышает безопасность, так как атака на туннель с меньшей вероятностью приведет к эксплуату `root`. Обратите внимание, что в Debian/Ubuntu используется группа `nogroup`.
- `verb 3` : устанавливает уровень детализации в значение по умолчанию 3. Увеличьте это число для просмотра более подробного вывода процесса OpenVPN. Если детализация установлена в 0, то вряд ли будет выдаваться какой-либо результат регистрации. Потому это не рекомендуется.
- `daemon`: указывает OpenVPN *демонизировать* себя - что означает продолжение работы процесса OpenVPN даже после закрытия окна терминала, в котором был запущен OpenVPN.
- `log-append <путь к файлу журнала>`: указывает путь к файлу журнала сервера. Используя `log-append` вместо `log <путь к файлу>`, мы запрещаем OpenVPN обрезать файл журнала при каждом запуске. Для этого файла также рекомендуется использовать абсолютный путь.

Файл конфигурации клиента содержит:

- `client`: переводит OpenVPN в режим клиента. Он инструктирует OpenVPN подключаться к удаленному серверу и получать и обрабатывать параметры конфигурации с сервера после успешного установления соединения. Оператор `client` внутренне расширен следующим образом:

```
tls-client
pull
```

- `proto udp`: указывает протокол для использования. Хотя это протокол по умолчанию, разумно явно указать его в файле конфигурации для избежания путаницы.
- `remote openvpnserver.example.com`: здесь указывается имя VPN-сервера для подключения. Имя может быть либо **полным именем домена (FQDN)**, либо адресом IPv4. Далее в этой главе мы увидим, как подключиться к VPN-серверу на основе IPv6.
- `port 1194`: порт, который клиент OpenVPN будет использовать для подключения к серверу. Значение по умолчанию - 1194, но можно использовать любой допустимый и доступный номер порта.

Заметка

Существует несколько способов указать удаленный адрес и порт для VPN-сервера. Например, также можно использовать `remote openvpnserver.example.com:1194`.

- `dev tun`: указывает имя устройства `tun`, которое будет использоваться для сервера. Не добавляя номер позади `tun` - мы инструктируем OpenVPN открыть новое устройство. Этому новому устройству будет присвоен первый доступный номер в ядре системы, начиная с 0 (`tun0`, `tun1`, `tun2` и т.д.). Для Windows-клиентов желательно сохранить эту строку как есть. Если необходимо использовать определенное устройство Windows, то потребуется опция `dev-node`.
- `nobind`: указывает клиенту OpenVPN не связывать (и не прослушивать) порт, указанный с использованием `port`. Вместо этого клиент OpenVPN будет использовать порт в диапазоне анонимных портов, который обычно составляет 1024-65335.
- `ca <путь к файлу CA>`: указывает путь к файлу CA. Этот файл CA должен содержать сертификат CA (или даже набор сертификатов), который использовался для подписи

сертификата сервера. Это необязательно должен быть тот же СА, который использовался для подписи сертификата клиента, хотя в нашей настройке PKI мы использовали тот же СА. В Linux/Unix рекомендуется использовать абсолютный путь для этого файла (а также пути к другим сертификатам и секретным ключам).

- `cert <путь к файлу сертификата X.509>`: указывает путь к файлу публичного сертификата клиента X.509. Можно настроить OpenVPN для использования аутентификации по имени пользователя/паролю вместо сертификатов, но это считается менее безопасным. В Linux/Unix рекомендуется использовать абсолютный путь для этого файла (а также пути к другим сертификатам и приватным ключам).
- `key <путь к файлу закрытого ключа>`: указывает путь к файлу приватного ключа клиента. Этот файл должен быть доступен для чтения только пользователю root (или администратору), так как OpenVPN прочтёт этот файл перед удалением пользовательских привилегий. В Linux рекомендуется использовать абсолютный путь для этого файла (а также пути к другим сертификатам и секретным ключам).

Обратите внимание, что мы не указали `daemon` или `log-append` для конфигурации клиента, так как в большинстве случаев приложение-оболочка запускает процесс `openvpn`. Это приложение будет затем управлять журналом OpenVPN. Наиболее часто используемые приложения-оболочки:

Операционная система	Приложения-оболочки
Windows	OpenVPN-GUI.exe (часть пакета установки OpenVPN)
Mac OS X	Tunnelblick или Viscosity
Linux	NetworkManager (с плагином OpenVPN)

Topology subnets против topology net30

OpenVPN поддерживает несколько топологий в режиме tun:

- `net30` (по умолчанию, может измениться с v2.4)
- `subnet` (подсеть)
- `p2p`

Начнем с последней - топология `p2p` почти никогда не использовалась и была первым методом назначения одного IP-адреса VPN-клиенту. Однако, она работает только на производных от Linux и Unix и, следовательно, никогда не использовалась очень широко.

Топология `net30` является текущим значением по умолчанию. В этом режиме OpenVPN устанавливает сетевой интерфейс «точка-точка» для каждого клиента (и для сервера) и назначает подсеть `/30` каждому. Это означает, что серверу и каждому клиенту назначен блок из четырех IP-адресов. В файле конфигурации сервера будет указано `server 10.200.0.0 255.255.255.0`. С топологией `net30` это заставляет OpenVPN назначать следующие IP-блоки:

- Серверу OpenVPN назначается с 10.200.0.0 по 10.200.0.3.
- Первому клиенту назначаются с 10.200.0.4 по 10.200.0.7.
- Второму клиенту назначаются с 10.200.0.8 по 10.200.0.11 и так далее.

Каждая подсеть `/30` состоит из четырех адресов; для первого клиента эти адреса следующие:

- 10.200.0.4: Это адрес подсети /30. Каждая подсеть должна иметь такой адрес, связанный с ней.
- 10.200.0.5: Это адрес виртуальной конечной точки. Он необходим для функционирования OpenVPN, но на самом деле его нельзя использовать, и он даже не может быть проверен.
- 10.200.0.6: IP-адрес VPN клиента.
- 10.200.0.7: широковещательный адрес подсети /30. Каждая подсеть должна иметь такой широковещательный адрес, связанный с ней.

Как видите, это не очень эффективный метод назначения IP-адресов - для каждого VPN-клиента выделяются четыре IP-адреса. Для небольших установок VPN этот метод работает нормально, но он не масштабируется для сервера с более чем 100 подключенными клиентами.

Чтобы преодолеть эту проблему - был введен режим `topology subnet`. Он позволяет OpenVPN назначать один IP-адрес всем клиентам, что значительно упрощает управление крупномасштабной VPN. Существуют некоторые проблемы с маршрутизацией на стороне сервера (подробнее см. раздел «*Маршрутизация и маршрутизация на стороне сервера*» далее в этой главе), из-за которых этот режим топологии не стал стандартным, но ожидается, что начиная с версии 2.4 он может стать режимом топологии по умолчанию.

Добавление повышенной безопасности

Начальный набор файлов конфигурации является хорошей отправной точкой для развертывания клиент-серверной конфигурации. Однако для системы производственного уровня мы должны добавить больше безопасности. Безопасность можно повысить двумя способами:

- Добавляя ключи `tls-auth`
- Путем проверки расширенных атрибутов использования ключей используемых сертификатов

Использование ключей `tls-auth`

В режиме клиент-сервер OpenVPN будет пытаться установить канал управления TLS для каждого подключаемого клиента. Настройка канала управления TLS требует много ресурсов, что делает OpenVPN подверженным атакам типа «отказ в обслуживании»: злоумышленник может запустить множество неправильно настроенных клиентов, которые будут одновременно пытаться подключиться к серверу OpenVPN. Для каждого из них сервер OpenVPN будет пытаться установить соединение TLS, что фактически приведет к отказу в обслуживании для правильно настроенных клиентов. Это особенно верно, когда OpenVPN работает с использованием `proto udp` (рекомендуется по умолчанию). UDP - это трафик без установления соединения, что означает проверку сервером действительности пакета OpenVPN для каждого нового UDP-пакета.

Для устранения этой возможной уязвимости OpenVPN ввел дополнительный уровень аутентификации в канал управления TLS, используя опцию `tls-auth`. Эта TLS-аутентификация должна выполняться с использованием предустановленного общего ключа, поскольку сервер еще не знает - пытается подключиться действительный клиент или нет. Предустановленные ключи, используемые здесь, являются теми же самыми ключами, используемыми в режиме точка-точка, как описано в [Главе 2, Режим точка-точка](#).

Генерация ключа tls-auth

Чтобы сгенерировать ключ `tls-auth` - мы используем ту же команду, что описана в [Главе 2](#), Режим точки-точки:

```
[root@server]# openvpn --genkey --secret /etc/openvpn/movpn/ta.key
```

Точно так же, как файл приватного ключа клиента, этот файл должен быть скопирован каждому клиенту с использованием безопасного канала или должен быть включен в пакет конфигурации безопасного клиента:

```
[root@client]# cd /etc/openvpn/movpn  
[root@client]# scp root@openvpnserver:/etc/openvpn/movpn/ta.key
```

Проверка атрибутов использования ключа сертификата

Когда сертификаты X.509 сгенерированы - к сертификату могут быть добавлены специальные атрибуты **расширенного использования ключа** (**Extended Key Usage** - EKU). Это позволяет нам указать назначение сертификата, например, сертификат *только для сервера* или сертификат *только для клиента*. Сертификаты, используемые на защищенных веб-сайтах, используют те же атрибуты EKU.

Скрипт `easy-rsa` и инструмент `ssladm` устанавливают атрибуты EKU по умолчанию при создании сертификата сервера или несерверного (клиентского) сертификата. Чтобы проверить атрибуты EKU сертификата, используйте следующие команды:

```
$ openssl x509 -text -noout -in server.crt | \  
grep -C 1 "Key Usage"  
  
X509v3 Extended Key Usage:  
    TLS Web Server Authentication  
X509v3 Key Usage:  
    Digital Signature, Key Encipherment
```

Это говорит нам о том, что сертификат `server.crt` может использоваться только для аутентификации сервера.

В старых сертификатах эти атрибуты EKU могут быть не заданы, а вместо этого используется (не рекомендуется) атрибут `Netscape Cert Type`. Скрипт `easy-rsa` и инструмент `ssladm` также устанавливают этот атрибут:

```
$ openssl x509 -text -noout -in server.crt | \  
grep -C 1 "Netscape Cert"  
  
Netscape Cert Type:  
    SSL Server
```

Однако этот сертификат может быть установлен только для серверных сертификатов.

Безопасность OpenVPN может быть повышена путем проверки этих атрибутов. Для этого мы используем опцию `remote-cert-tls`.

Опция `remote-cert-tls client` предписывает серверу OpenVPN разрешать подключения только от VPN-клиентов, у которых есть сертификат с атрибутом EKU X.509, установленным как `TLS Web Client Authentication`.

Это не позволяет хакеру настроить мошеннический сервер OpenVPN с помощью клиентского сертификата.

Аналогично, для клиента опция `remote-cert-tls server` дает указание клиенту OpenVPN разрешать подключения только к VPN-серверу, у которого есть сертификат с атрибутом EKU X.509, установленным в значение `TLS Web Server Authentication`.

Это не позволяет злонамеренному клиенту настроить мошеннический сервер OpenVPN для привлечения подключений от других пользователей VPN.

Также можно проверить наличие атрибута `Netscape Cert Type`. Поскольку это атрибут сертификата сервера - клиент OpenVPN должен проверить этот атрибут при подключении. Для этого можно использовать опцию `ns-cert-type server`. Предпочтительно использовать параметр `remote-cert-tls`.

Основные конфигурационные файлы производственного уровня

Мы расширим предыдущие файлы конфигурации клиента и сервера, чтобы использовать только что созданный ключ `tls-auth`. Мы сделаем это добавив строку в файл конфигурации `movpn-04-01-server.conf`, а также второй параметр повышения безопасности:

```
proto udp
port 1194
dev tun
server 10.200.0.0 255.255.255.0
topology subnet
persist-key
persist-tun
keepalive 10 60

remote-cert-tls client
tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

user nobody
group nobody

verb 3
daemon
log-append /var/log/openvpn.log
```

Заметка

Обратите внимание, что порядок операторов в этом файле является случайным. Строки `remote-cert-tls` и `tls-auth` могут быть добавлены в любом месте файла.

Этот файл конфигурации является основным файлом конфигурации сервера, который мы будем использовать в этой и других главах. Сохраните его как `basic-udp-server.conf`, чтобы мы могли использовать его позже.

Мы добавляем две одинаковые строки в файл конфигурации клиента `movpn-04-01-client.conf`:

```
client

proto udp
remote openvpnserver.example.com
port 1194
dev tun
nobind

remote-cert-tls server
tls-auth /etc/openvpn/movpn/ta.key 1
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/client1.crt
```

```
key      /etc/openvpn/movpn/client1.key
```

Сохраните его как basic-udp-client.conf.

Второй параметр опции `tls-auth` - это так называемое направление ключа. OpenVPN поддерживает использование *направления* ключа, то есть разные ключи используются для входящих и исходящих данных. Это еще больше повышает безопасность. Флаг `direction` должен быть установлен в 0 на одном конце и 1 - на другом. В режиме клиент-сервер это означает, что сервер имеет параметр 0 для направления, а для всех клиентов параметр направления установлен в 1.

Когда мы запускаем сервер OpenVPN, то видим что канал управления TLS теперь защищен статическим ключом:

```
[root@server]# openvpn --config basic-udp-server.conf --suppresstimestamps
OpenVPN 2.3.2 x86_64-redhat-linux-gnu [SSL (OpenSSL)] [LZO] [EPOLL]
[PKCS11] [eurephia] [MH] [IPv6] built on Sep 12 2013
Enter Private Key Password:
WARNING: this configuration may cache passwords in memory -- use
the auth-nocache option to prevent this
Control Channel Authentication: using '/etc/openvpn/movpn/ta.key' as
a OpenVPN static key file
TUN/TAP device tun0 opened
do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
/sbin/ip link set dev tun0 up mtu 1500
/sbin/ip addr add dev tun0 10.200.0.1/24 broadcast 10.200.0.255
GID set to nobody
UID set to nobody
UDPV4 link local (bound): [undef]
UDPV4 link remote: [undef]
Initialization Sequence Completed
```

Аналогично, когда мы запускаем клиент OpenVPN - мы видим:

```
[root@client]# openvpn --config basic-udp-client.conf --suppresstimestamps
OpenVPN 2.3.2 x86_64-redhat-linux-gnu [SSL (OpenSSL)] [LZO] [EPOLL]
[PKCS11] [eurephia] [MH] [IPv6] built on Sep 12 2013
Control Channel Authentication: using '/etc/openvpn/movpn/ta.key' as
a OpenVPN static key file
UDPV4 link local: [undef]
UDPV4 link remote: [AF_INET]openvpnserver:1194
[Mastering OpenVPN Server] Peer Connection Initiated with
[AF_INET]openvpnserver:1194
TUN/TAP device tun0 opened
do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
/sbin/ip link set dev tun0 up mtu 1500
/sbin/ip addr add dev tun0 10.200.0.2/24 broadcast 10.200.0.255
Initialization Sequence Completed
```

Конфигурация на основе TCP

Протоколом, используемым в OpenVPN по умолчанию, является протокол UDP. Создание версий с TCP на основе созданных ранее файлов конфигурации очень простое. В файлах конфигурации клиента и сервера измените строку `proto udp` на `proto tcp`. Весь файл конфигурации сервера для TCP указан здесь:

```
proto tcp
port 1194
dev tun
server 10.200.0.0 255.255.255.0
topology subnet
```

```
persist-key
persist-tun
keepalive 10 60

remote-cert-tls client
tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

user nobody
group nobody

verb 3
daemon
log-append /var/log/openvpn.log
```

Сохраните этот файл конфигурации как basic-tcp-server.conf.

Аналогично, для файла конфигурации клиента:

```
client
proto tcp
remote openvpnserver.example.com
port 1194
dev tun
nobind

remote-cert-tls server
tls-auth /etc/openvpn/movpn/ta.key 1
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/client1.crt
key /etc/openvpn/movpn/client1.key
```

Сохраните его как basic-tcp-client.conf.

Конфигурационные файлы для Windows

Базовые файлы конфигурации для платформы Windows немного отличаются от файлов для платформ Linux/Unix или Mac OS. На платформе Windows используется оболочка OpenVPN-GUI.exe, которая предполагает, что все файлы конфигурации хранятся в каталоге C:\Program Files\OpenVPN\config или его подкаталогах. Название каталога Program Files может отличаться для других языков. На всех языках переменная среды Windows %PROGRAMFILES% будет указывать на правильное расположение.

Таким образом, базовые файлы конфигурации UDP и TCP на самом деле немного короче. Создайте файл конфигурации клиента UDP:

```
client
proto udp
remote openvpnserver.example.com
port 1194
dev tun
nobind

remote-cert-tls server
tls-auth ta.key 1
ca movpn-ca.crt
cert client1.crt
key client1.key
```

Сохраните его как `basic-udp-client.ovpn`, чтобы мы могли использовать его позже в этой книге.

Аналогичным образом создайте конфигурацию клиента:

```
client
proto tcp
remote openvpnserver.example.com
port 1194
dev tun
nobind
remote-cert-tls server
tls-auth ta.key 1
ca      movpn-ca.crt
cert    client1.crt
key     client1.key
```

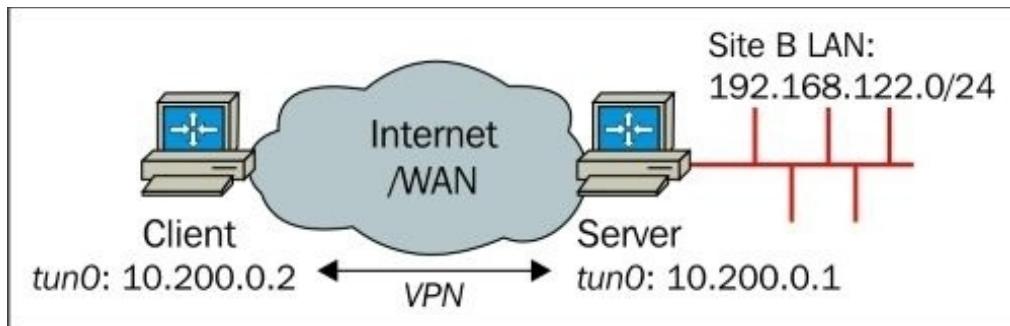
Сохраните её как `basic-tcp-client.ovpn`.

Маршрутизация и маршрутизация на стороне сервера

VPN действительно полезен только тогда, когда клиенты VPN имеют доступ к ресурсам на стороне сервера. Для доступа к этим ресурсам в большинстве случаев необходима маршрутизация. OpenVPN имеет много опций для автоматической настройки и удаления дополнительных маршрутов при подключении или отключении клиента.

Следует отметить, что большинство проблем при устранении неполадок OpenVPN связано с маршрутизацией. Настройка VPN-соединения - это одно, а правильная передача сетевого трафика - это другое. Она имеет мало общего с самим OpenVPN и больше связана с таблицами маршрутизации и правилами брандмауэра на стороне клиента и сервера.

Наиболее распространенная схема доступа к ресурсам в серверной сети изображена здесь:



Локальная сеть на стороне сервера: **192.168.122.0/24**. Ресурсы, к которым VPN-клиенты должны получить доступ, находятся в этой подсети. Таким образом - сервер должен проинструктировать VPN-клиентов что необходимо установить дополнительный маршрут. Это делается с помощью опции `push`, которая передает маршрут клиенту. Этого также можно достичь добавив маршрут в сам файл конфигурации клиента, но это плохо масштабируется. Связано это с тем, что для каждого нового сетевого маршрута на стороне сервера необходимо обновлять все файлы конфигурации клиента.

Мы начнем с файла `basic-udp-server.conf` и добавим одну строку:

```
proto udp
port 1194
dev tun
server 10.200.0.0 255.255.255.0
topology subnet
persist-key
```

```

persist-tun
keepalive 10 60

remote-cert-tls client
tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

user nobody
group nobody

verb 3
daemon
log-append /var/log/openvpn.log

push "route 192.168.122.0 255.255.255.0"

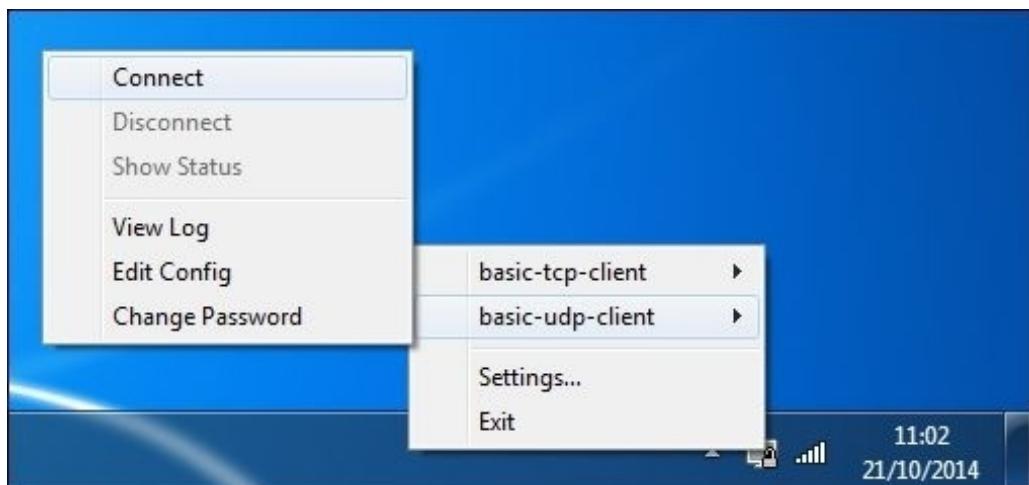
```

Мы сохраним его как `movpn-04-03-server.conf` и запустим сервер OpenVPN, используя этот файл конфигурации. На этот раз мы используем Windows 7 64-разрядную версию Professional в качестве клиента OpenVPN, на которой установлена версия OpenVPN 2.3.4-I004 для X86_64. Скопируйте следующие файлы на компьютер с Windows:

- `basic-udp-client.ovpn`
- `movpn-ca.crt`
- `client1.crt`
- `client1.key`

Возьмите и поместите их в `C:\Program Files\OpenVPN\config` (или `%PROGRAMFILES%\OpenVPN\config`).

Запустите приложение OpenVPN GUI, выберите конфигурацию `basic-udp-client` и нажмите **Connect**:



Как только соединение будет успешно установлено - значок OpenVPN GUI станет зеленым и информация о соединении будет отображаться при наведении курсора на значок:



Теперь мы можем проверить, работает ли VPN-соединение с сервером, открыв командную строку и выполнив команду ping в сторону сервера:

```
Console
c:\Users\janjust\movpn>ping 10.200.0.1

Pinging 10.200.0.1 with 32 bytes of data:
Reply from 10.200.0.1: bytes=32 time=3ms TTL=64
Reply from 10.200.0.1: bytes=32 time=3ms TTL=64
Reply from 10.200.0.1: bytes=32 time=6ms TTL=64
Reply from 10.200.0.1: bytes=32 time=4ms TTL=64

Ping statistics for 10.200.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 <0% loss>,
    Approximate round trip times in milli-seconds:
        Minimum = 3ms, Maximum = 6ms, Average = 4ms

c:\Users\janjust\movpn>
```

После того, как мы проверили что сервер OpenVPN доступен - мы должны убедиться, что сервер OpenVPN пересыпает IP-трафик и нам необходимо добавить дополнительный маршрут на шлюзе со стороны сервера чтобы гарантировать правильное перенаправление VPN-трафика обратно через сервер VPN. Без этого маршрута машины в серверной сети не будут знать откуда поступает трафик VPN с IP-адресами 10.200.0.0/24 и, скорее всего, будут неправильно маршрутизировать или отбрасывать пакеты:

```
[root@server]# sysctl -w net.ipv4.ip_forward=1
[router]# ip route add 10.200.0.0/24 via 192.168.122.1
```

Теперь мы проверим таблицу маршрутизации на стороне клиента и проверим - сможем ли мы подключиться к компьютеру в локальной сети на стороне сервера:

```
Console
c:\Users\janjust\movpn>route print | find "10.200.0"
 10.200.0.0  255.255.255.0      On-link          10.200.0.2    286
   10.200.0.2  255.255.255.255  On-link          10.200.0.2    286
 10.200.0.255 255.255.255.255  On-link          10.200.0.2    286
 192.168.122.0  255.255.255.0      10.200.0.1    10.200.0.2    30
   224.0.0.0    240.0.0.0      On-link          10.200.0.2    286
 255.255.255.255 255.255.255.255  On-link          10.200.0.2    286

c:\Users\janjust\movpn>ping 192.168.122.184

Pinging 192.168.122.184 with 32 bytes of data:
Reply from 192.168.122.184: bytes=32 time=4ms TTL=63
Reply from 192.168.122.184: bytes=32 time=7ms TTL=63
Reply from 192.168.122.184: bytes=32 time=3ms TTL=63
Reply from 192.168.122.184: bytes=32 time=4ms TTL=63

Ping statistics for 192.168.122.184:
    Packets: Sent = 4, Received = 4, Lost = 0 <0% loss>,
    Approximate round trip times in milli-seconds:
        Minimum = 3ms, Maximum = 7ms, Average = 4ms
```

Первая часть выходных данных показывает что несколько маршрутов для подсети VPN 10.200.0.0/24 были добавлены к таблицам маршрутизации, включая маршрут для pushed подсети 192.168.122.0/24. Обратите внимание на последний столбец выходных данных,

который показывает метрику маршрута. Windows вычисляет метрику (в данном случае 286), но она может быть отменена с помощью правильных операторов маршрута. Маршрут, добавленный с помощью `push route 192.168.122.0 255.255.255.0`, имеет более низкую метрику, поскольку была указана метрика OpenVPN по умолчанию - 30.

Специальные параметры для вариантов маршрута

Аналогично тому, что описано в [Главе 2](#), Режим точка-точка инструкция `push route <network> <netmask> [vpn_gateway] [metric]` жизненно важна в этой настройке. Параметр `route` принимает до четырех параметров: два обязательных и два дополнительных. Третий параметр играет важную роль в этой настройке. Слово `vpn_gateway` - это специальное ключевое слово OpenVPN, указывающее адрес удаленной конечной точки VPN. Обычно это ключевое слово указывать не нужно, кроме случаев необходимости указания метрики для этого маршрута.

Полный синтаксис для инструкции маршрута - `route <network> <netmask> [gateway] [metric]`, где `gateway` может быть явно задан как адрес IPv4 или могут использоваться специальные ключевые слова `vpn_gateway` или `net_gateway`. Если шлюз и метрика не указаны - используется `vpn_gateway`.

Ключевым словом `net_gateway` полезно указывать подсеть, которая *не* должна быть маршрутизируема через VPN. Для `net_gateway` заменяется шлюз по умолчанию до установления VPN-соединения.

Метрика имеет метрику по умолчанию, которую можно установить с помощью `route-metric m`, которая затем применяется ко всем маршрутам. Если вы хотите отменить метрику для определенного маршрута (как мы это сделали в данном примере), то необходимо указать шлюз (в нашем случае `vpn_gateway`), за которым следует метрика для этого конкретного маршрута.

Маскарадинг

Иногда невозможно добавить маршрут на стороне сервера для перенаправления всего VPN-трафика обратно на сервер OpenVPN. В этом случае быстрый и грязный подход заключается в использовании маскарадинга. В Linux вы можете использовать команду `iptables` для настройки маскарадинга на сервере:

```
[root@server]# iptables -t nat -I POSTROUTING -o eth0 \
    -s 10.200.0.0/24 -j MASQUERADE
```

Этот оператор `iptables` указывает ядру Linux перезаписывать весь трафик, поступающий из подсети VPN `10.200.0.0/24` и покидающий интерфейс Ethernet `eth0`. У трафика, покидающего интерфейс `eth0` адрес источника переписывается так, что он выглядит будто исходит от самого сервера OpenVPN, а не от клиента OpenVPN. Это простой способ заставить маршрутизацию работать, но недостаток заключается в том, что больше невозможно различить, поступает ли данный трафик с самого сервера OpenVPN или с одного из подключенных клиентов.

Перенаправление шлюза по умолчанию

Очень распространенное использование VPN - это маршрутизация всего трафика через безопасный туннель. Это позволяет безопасно получить доступ к сети или даже самому Интернету изнутри *враждебной* среды (например, небезопасный wi-fi в интернет-кафе).

Перенаправление шлюза по умолчанию достигается добавлением строки `push "redirect-gateway [def1 local bypass-dhcp bypass-dns]"` в файл конфигурации сервера.

Параметры для `redirect-gateway`, перечисленные ранее, являются необязательными, но они

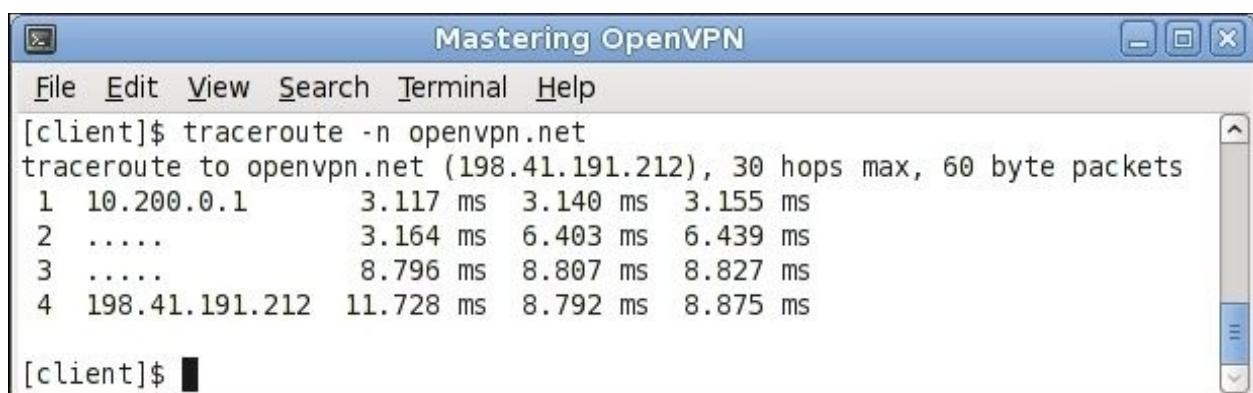
могут играть очень важную роль:

- Параметры не добавлены: В этом случае OpenVPN заменит существующий шлюз по умолчанию (0.0.0.0/0) на адрес самого сервера OpenVPN. Также добавляется дополнительный маршрут к самому серверу OpenVPN, чтобы сам трафик OpenVPN отправлялся непосредственно на сервер, а не через туннель. Недостаток заключается в том, что если соединение OpenVPN остановлено или прервано - исходный шлюз по умолчанию не восстанавливается. Это обычно приводит к полной потере сетевого подключения.
- Параметр `def1`: вместо замены существующего шлюза по умолчанию OpenVPN добавит два новых маршрута, 0.0.0.0/1 и 128.0.0.0/1. Эти маршруты вместе также охватывают все пространство IPv4 и являются более конкретными (/1), чем обычный шлюз (/0). Маршрутизация всегда происходит по более конкретным маршрутам, и, таким образом, весь трафик передается через VPN. Преимущество этого трюка в том, что шлюз по умолчанию остается без изменений. Если VPN-соединение останавливается - исходный шлюз можно восстановить. Обратите внимание, что в этом случае OpenVPN добавит явный маршрут к самому серверу OpenVPN - поэтому сам зашифрованный трафик не будет отправляться через туннель.
- Параметр `bypass-dhcp`: иногда полезно добавить явный маршрут к локальному DHCP-серверу в локальной сети на стороне клиента. Это позволяет избежать туннелирования обновлений DHCP через VPN, хотя в большинстве сетевых настроек этого не происходит, поскольку как правило существует более конкретный маршрут к сегменту сети, в котором расположен локальный сервер DHCP.
- Параметр `bypass-dns`: иногда может потребоваться добавить явный маршрут к локальному DNS-серверу, так как в противном случае разрешение доменных имен нарушается. Это происходит только в том случае - если вы хотите использовать клиентский DNS-сервер, но хотите направить весь трафик через туннель.

Помимо опции `redirect-gateway`, мы также можем указать опцию `push "redirect-private [def1 local bypass-dhcp bypass-dns]"` в файл конфигурации сервера. Этот параметр принимает те же параметры, что и `redirect-gateway`, но не изменяет существующий шлюз по умолчанию. Это может быть полезно для передачи маршрутов к частным подсетям.

Сейчас мы добавляем `push "redirect-gateway def1"` в файл конфигурации `basic-udp-server.conf`. Сохраните его как `mvpn-04-06-server.conf`, запустите сервер OpenVPN и повторно подключите клиента, используя файл конфигурации по умолчанию.

После того, как соединение установлено, мы проверяем что весь трафик теперь проходит через VPN, используя команду `traceroute` (используйте `tracert -d` в командной оболочке Windows):



The screenshot shows a terminal window titled "Mastering OpenVPN". The window has a menu bar with File, Edit, View, Search, Terminal, and Help. The main area displays the output of the command `[client]$ traceroute -n openvpn.net`. The output shows a traceroute to the server at 198.41.191.212, with four hops listed:

Hop	Address	RTT 1	RTT 2	RTT 3
1	10.200.0.1	3.117 ms	3.140 ms	3.155 ms
2	3.164 ms	6.403 ms	6.439 ms
3	8.796 ms	8.807 ms	8.827 ms
4	198.41.191.212	11.728 ms	8.792 ms	8.875 ms

Первый hop в выводе отслеживаемого маршрута `10.200.0.1`, что является IP-адресом сервера OpenVPN. Это доказывает, что трафик проходит через VPN по умолчанию.

Параметр конфигурации `redirect-gateway def1` указывает клиенту OpenVPN добавить три маршрута в клиентскую операционную систему:

```
10.198.1.1 via 192.168.4.254 dev eth0
0.0.0.0/1 via 10.200.0.1 dev tun0
128.0.0.0/1 via 10.200.0.1 dev tun0
```

Первый маршрут - это явный маршрут от клиента к серверу OpenVPN через интерфейс LAN. Этот маршрут необходим, так как в противном случае весь трафик для самого сервера OpenVPN будет проходить через туннель.

Два других маршрута - это хитрая уловка для отмены маршрута по умолчанию, чтобы весь трафик передавался через туннель, а не шлюз по умолчанию для локальной сети.

Преимущество этого метода заключается в том, что исходный шлюз по умолчанию остается без изменений. Когда VPN отключена - первоначальный адрес шлюза автоматически вступает во владение снова. Если бы мы просто использовали `redirect-gateway` - есть вероятность что шлюз по умолчанию будет потерян при отключении VPN, что приведет к полной потере сетевого подключения.

Недостатком этого метода являются клиенты Windows 7 и выше: Windows иногда отказывается доверять адаптеру TAP-Win без маршрута по умолчанию и поэтому помечает его как общедоступный адаптер. В Windows 7 невозможно использовать общедоступный адаптер для общего доступа к файлам или принтерам. В следующей главе мы рассмотрим как обойти эту особенность.

Специфичная для клиента конфигурация - файлы CCD

В инсталляциях, где один сервер может обрабатывать подключения от множества клиентов, иногда необходимо установить параметры для каждого клиента, которые отменяют глобальные параметры или добавить дополнительные параметры для конкретного клиента. Опция `client-config-dir` очень полезна для этого. Она позволяет администратору VPN назначать конкретный IP-адрес клиенту для передачи определенных параметров, как например DNS-сервер - конкретному клиенту или для временного отключения клиента. Эта опция также необходима, если вы хотите маршрутизировать клиентскую подсеть на сторону сервера, как мы увидим позже.

`client-config-dir` или CCD-файл может содержать следующие параметры:

- `push`: полезно для отправки DNS и WINS-серверов, маршрутов и т.д.
- `push-reset`: полезно для отмены глобальных параметров `push`
- `iroute`: полезно для маршрутизации клиентских подсетей IPv4 на сервер
- `iroute-ipv6`: полезно для маршрутизации клиентских подсетей IPv6 на сервер
- `ifconfig-push`: полезно для назначения определенного IPv4- адреса клиенту
- `ifconfig-ipv6-push`: полезно для назначения конкретного IPv6- адреса клиенту
- `disable`: полезно для временного полного отключения клиента
- `config`: полезно для включения другого файла конфигурации CCD

Чтобы использовать файлы CCD мы добавляем в файл конфигурации `basic-udp-server.conf` строку:

```
client-config-dir /etc/openvpn/movpn/clients
```

Сохраните его как `movpn-04-04-server.conf`. Затем создайте каталог для CCD и создайте в нем файл CCD для клиента с сертификатом `client1.crt`:

```
[root@server]# mkdir -p /etc/openvpn/movpn/clients
[root@server]# echo "ifconfig-push 10.200.0.99 255.255.255.0" \
> /etc/openvpn/movpn/clients/client1
[root@server]# chmod 755 /etc/openvpn/movpn/clients
[root@server]# chmod 644 /etc/openvpn/movpn/clients/client1
```

Имя файла CCD основано на общем имени субъекта сертификата (часть “/CN=”), как указано в файле `client1.crt`:

```
$ openssl x509 -subject -noout -in client1.crt
subject= /C=ZA/ST=Enlightenment/O=Mastering
OpenVPN/CN=client1/emailAddress=root@example.org
```

В нашем случае имя файла должно быть просто `client1` без расширения, даже в Windows! Если в common name есть пробелы - их необходимо преобразовать в подчеркивания (_). Если проводник Windows настроен на скрытие расширений для распространенных типов файлов - проще будет открыть окно командной оболочки (cmd.exe) и удалить расширение с помощью следующих команд:

```
C:\> cd %PROGRAMFILES%\openvpn\config\clients
C:\> rename client1.txt client1,145.102.134.201:35519
```

Затем мы запускаем сервер OpenVPN, используя этот файл конфигурации и подключаем VPN-клиента. Журнал подключений показывает что клиенту назначен адрес 10.200.0.99:

```
[root@client]# openvpn --config basic-udp-client.conf
OpenVPN 2.3.2 x86_64-redhat-linux-gnu [SSL (OpenSSL)] [LZO] [EPOLL]
[PKCS11] [eurephia] [MH] [IPv6] built on Sep 12 2013
Control Channel Authentication: using '/etc/openvpn/movpn/ta.key' as
a OpenVPN static key file
UDPV4 link local: [undef]
UDPV4 link remote: [AF_INET]openvpnserver:1194
[Mastering OpenVPN Server] Peer Connection Initiated with
[AF_INET]openvpnserver:1194
TUN/TAP device tun0 opened
do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
/sbin/ip link set dev tun0 up mtu 1500
/sbin/ip addr add dev tun0 10.200.0.99/24 broadcast 10.200.0.255
Initialization Sequence Completed
```

При настройке детальности вывода по-умолчанию журнал на стороне сервера не показывает никаких сообщений о получении файла CCD. Увеличьте настройку детальности до 5 или выше, чтобы увидеть обрабатывается ли файл CCD:

```
<client-ip>:49299 [client1] Peer Connection Initiated with
[AF_INET]<client-ip>:49299
client1<client-ip>:49299 OPTIONS IMPORT: reading client specific
options from: /etc/openvpn/movpn/clients/client1client1<client-ip>:49299
MULTI: Learn: 10.200.0.99 -> client1<client-ip>:49299
```

Как определить, правильно ли обрабатывается файл CCD

Поиск и исправление правильности обработки CCD-файла может быть немного сложнее. Следующие рекомендации помогут при отладке проблем с файлами CCD:

- Всегда указывайте полный путь для опции `client-config-dir`.
- Убедитесь что каталог доступен, а файл CCD доступен для чтения пользователю, который используется для запуска OpenVPN (в большинстве случаев `nobody` или

openvpn; в конфигурациях, перечисленных в этой книге, используется user nobody).

- Убедитесь что для файла CCD используется правильное имя файла без каких-либо расширений.
- Если возможно - добавьте опцию ccd-exclusive в файл конфигурации сервера. Это дает указание OpenVPN разрешать клиентские подключения только при наличии определенного CCD-файла для этого клиента. Если при чтении CCD-файла для определенного клиента возникла проблема - клиенту также будет отказано в доступе. Таким образом, вы будете знать, что ваши настройки client-config-dir выполнены неправильно.

CCD файлы и топология net30

Если вы используете (по умолчанию) настройку топологии (topology net30), то оператор ifconfig-push немного отличается. Поскольку каждому клиенту теперь назначена подсеть /30, в операторе ifconfig-push должна быть указана действительная подсеть VPN /30.

Применяются следующие правила:

- Каждая подсеть /30 должна начинаться с адреса, кратного 4 (4, 8, 12 и т.д.)
- Локальный IP-адрес VPN является третьим адресом в этой подсети.
- IP-адрес виртуальной удаленной конечной точки является вторым адресом.

Например, действительный IP-адрес для VPN-клиента - 10.200.0.50:

- Подсеть 10.200.0.48/32, где 48 кратно 4
- IP-адрес VPN-клиента будет 50 ($48 + 2 = 50$)
- Виртуальная удаленная конечная точка - 49 ($48 + 1 = 49$)

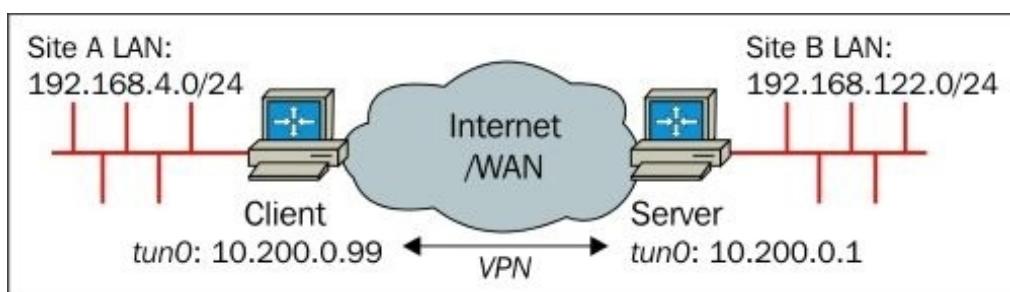
Файл CCD теперь должен содержать следующее:

```
ifconfig-push 10.200.0.50 10.200.0.49
```

Клиентская маршрутизация

Иногда полезно разрешить серверу VPN (или другим клиентам VPN) доступ к ресурсам, подключенными к конкретному клиенту. Это известно как маршрутизация на стороне клиента. Для клиентской маршрутизации в OpenVPN для этого клиента требуется файл CCD, содержащий оператор iroute. Это также требует соответствующей инструкции route в файле конфигурации сервера OpenVPN.

Рассмотрим следующую схему сети:



Подсеть **192.168.4.0/24** должна быть доступна из локальной сети на стороне сервера, а подсеть **192.168.122.0/24** - на стороне сервера из локальной сети клиента. Это может быть достигнуто следующим образом:

1. Добавьте две строки в файл конфигурации basic-udp-server.conf:

```
client-config-dir /etc/openvpn/movpn/clients
route 192.168.4.0 255.255.255.0 10.200.0.1
```

Сохраните его как movpn-04-05-server.conf.

2. Создайте CCD-файл client1 в каталоге /etc/openvpn/movpn/clients с содержимым:

```
ifconfig-push 10.200.0.99 255.255.255.0
iroute 192.168.4.0 255.255.255.0
push "route 192.168.122.0 255.255.255.0"
```

3. Убедитесь, что пересылка (форвардинг) IP-трафика включена и разрешена как на клиенте, так и на сервере:

```
[root@client]# sysctl -w net.ipv4.ip_forward=1
[root@server]# sysctl -w net.ipv4.ip_forward=1
```

4. Запустите сервер OpenVPN, используя файл конфигурации movpn-04-05-server.conf.
5. Подключите клиента, используя файл конфигурации по умолчанию basic-udp-client.conf.
6. После того как соединение установлено - мы проверяем, что обе подсети могут связаться друг с другом используя ping:

```
[root@client]# ping -c 3 192.168.122.184
PING 192.168.122.184 (192.168.122.184) 56(84) bytes of data.
64 bytes from 192.168.122.184: icmp_seq=1 ttl=63 time=3.29 ms
64 bytes from 192.168.122.184: icmp_seq=2 ttl=63 time=3.27 ms
64 bytes from 192.168.122.184: icmp_seq=3 ttl=63 time=3.31 ms
--- 192.168.122.184 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 3.277/3.296/3.317/0.016 ms
```

```
[root@server]# ping -c 3 192.168.4.10
PING 192.168.4.10 (192.168.4.10) 56(84) bytes of data.
64 bytes from 192.168.4.10: icmp_seq=1 ttl=63 time=6.31 ms
64 bytes from 192.168.4.10: icmp_seq=2 ttl=63 time=5.07 ms
64 bytes from 192.168.4.10: icmp_seq=3 ttl=63 time=5.14 ms
--- 192.168.4.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2007ms
rtt min/avg/max/mdev = 5.073/5.512/6.317/0.575 ms
```

Подробное объяснение конфигурации client-config-dir

На первом этапе мы добавляем две новые строки в файл конфигурации сервера. Эти строки устанавливают client-config-dir и инструктируют OpenVPN добавить системный сетевой маршрут для подсети 192.168.4.0/24. Здесь необходимо явно указать адрес шлюза 10.200.0.1 из-за незначительной ошибки в OpenVPN. Ожидается, что этот недостаток будет исправлен в версии 2.4, после чего вы можете снова указать route 192.168.4.0 255.255.255.0.

Содержимое файла CCD указывает OpenVPN, что при подключении клиента с Common Name client1 IP-адрес для этого клиента должен быть установлен на 10.200.0.99. Кроме того, OpenVPN должен установить внутренний маршрут (iroute) для этого клиента, чтобы сам OpenVPN знал, что подсеть 192.168.4.0/24 расположена за этим конкретным клиентом.

Наконец, оператор push route инструктирует OpenVPN "продвинуть" маршрут для этой конкретной подсети клиенту client1. Таким образом, сервер OpenVPN может прозрачно передавать разные маршруты разным клиентам.

Заметка

Выборочное продвижение маршрута к конкретному клиенту может быть удобным, но оно не защищено от несанкционированного доступа. Мошеннический VPN-клиент, который сам добавит маршрут к этой подсети, также будет иметь к ней доступ. Если вам нужно контролировать доступ к определенной подсети - используйте решение в виде межсетевого экрана, такого как `iptables` или `ipfw`.

В этом примере показана гибкость параметров конфигурации OpenVPN. Без необходимости изменять одну строку в файле конфигурации клиента можно назначить другой IP-адрес, направить конкретную подсеть клиенту или маршрутизировать конкретную подсеть из сети клиента в локальную сеть на стороне сервера.

Трафик клиент-клиент

OpenVPN также позволяет вам настраивать трафик клиент-клиент. По умолчанию VPN-клиенты не могут напрямую общаться друг с другом. Это хорошая мера безопасности, но иногда необходимо разрешить межклиентский трафик. Помните, что весь VPN-трафик клиент-клиент будет проходить через сервер OpenVPN: от `client1` к VPN-серверу, а затем от VPN-сервера к `client2` и наоборот. Это может легко привести к проблемам с производительностью.

В режиме `tun` соединение клиент-клиент может быть достигнуто либо с помощью `iptables`, либо с помощью параметра OpenVPN `client-to-client`. Параметр `client-to-client` имеет преимущество в том, что он быстрее: трафик от одного клиента, поступающий на сервер, автоматически перенаправляется второму клиенту, не проходя через таблицы системной маршрутизации или правила брандмауэра. Недостатком является сложность контроля трафика и невозможность применения контроля доступа.

Без параметра `client-to-client` трафик от одного клиента принимается сервером OpenVPN, перенаправляется в таблицы системной маршрутизации и межсетевого экрана и (при правильной настройке) снова возвращается на сервер OpenVPN. Затем сервер пересыпает его второму клиенту.

Если другим VPN-клиентам необходим доступ к подсети 192.168.4.0/24, как указано в предыдущем примере, то конфигурацию сервера необходимо расширить строкой:

```
push "192.168.4.0 255.255.255.0"
```

Это дает команду серверу OpenVPN продвинуть маршрут всем клиентам, которые находятся в подсети 192.168.4.0/24, доступной через VPN-туннель, за исключением клиента `client1`. Сам клиент `client1` исключен из-за соответствующей записи `iroute`.

Файл состояния OpenVPN

OpenVPN предлагает несколько вариантов для мониторинга клиентов, подключенных к серверу. Наиболее часто используемый метод - использование файла состояния. Файл состояния OpenVPN постоянно обновляется процессом OpenVPN и содержит следующую информацию:

- Какие клиенты подключены
- С какого IP-адреса подключаются клиенты
- Количество байт, которые каждый клиент получил и передал
- Время подключения клиента
- Кроме того, таблица маршрутизации также показывает, какие сети маршрутизируются каждому клиенту.

Мы поправим конфиг сервера `tnovpn-04-05-server.conf` в плане клиентской маршрутизации,

добавив строку:

```
proto udp
port 1194
dev tun
server 10.200.0.0 255.255.255.0
topology subnet
persist-key
persist-tun
keepalive 10 60

remote-cert-tls client
tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn-movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

user nobody
group nobody

verb 3
daemon
log-append /var/log/openvpn.log

client-config-dir /etc/openvpn/movpn/clients
route 192.168.4.0 255.255.255.0 10.200.0.1
status /var/run/openvpn.status 3
```

Сохраните его как `movpn-04-07-server.conf`. После установления VPN-соединения мы видим следующее содержимое в файле состояния после того, как VPN-клиент `client1` подключился и передал некоторые данные:

```
OpenVPN CLIENT LIST
Updated, Tue Oct 21 15:45:27 2014
Common Name, Real Address, Bytes Received, Bytes Sent, Connected Since
client1,<client-IP>:35519, 7730, 9342, Tue Oct 21 15:44:35 2014
ROUTING TABLE
Virtual Address, Common Name, Real Address, Last Ref
192.168.4.0/24, client1, 145.102.134.201:35519, Tue Oct 21 15:44:35
2014
10.200.0.99, client1, 145.102.134.201:35519, Tue Oct 21 15:44:35 2014
GLOBAL STATS
Max bcast/mcast queue length, 0
END
```

`CLIENT LIST` показывает список подключенных клиентов, включая информацию о количестве полученных и отправленных байт.

`ROUTING TABLE` показывает список внутренних маршрутов OpenVPN:

- Подсеть 192.168.4.0/24 маршрутизируется на `client1` из-за оператора `iroute` в конфигурации сервера
- IP-адрес 10.200.0.99 - это адрес `client1`, который мы явно установили в файле CCD с именем `client1`

При отключении клиента файл состояния обновляется через 3 секунды и подключенный клиент больше не отображается в списке.

Заметка

Когда клиент отключается - вся информация удаляется из файла состояния и вся статистика

сбрасывается. Если клиент снова подключается позже - количество принятых и отправленных байт начинается с нуля. Скрипт отключения клиента получает всю информацию о состоянии, когда клиент был отключен.

Вторым параметром опции `status` является интервал обновления (перезаписи) файла состояния. Значение по умолчанию составляет 60 секунд.

Надежное отслеживание соединения в режиме UDP

Сервер OpenVPN при использовании протокола UDP не может сразу обнаружить что клиент отключен либо преднамеренно, либо из-за плохого интернет-соединения. Это позволяет клиенту, в случае плохого соединения, переподключиться к серверу VPN без потери всех туннельных соединений. Недостатком является то, что серверу OpenVPN требуется некоторое время для понимания что клиент отключился.

Клиент OpenVPN может явно уведомить сервер о своем отключении, используя опцию `explicit-exit-notify`. Эта опция принимает один параметр, который указывает количество явных сообщений, которые клиент пытается отправить на сервер. Значение по умолчанию равно 1, что не работает если само основное сетевое соединение нестабильно. В этом случае рекомендуется увеличить данное значение до 3.

При использовании `explicit-exit-notify` сервер OpenVPN сразу же получает сообщение об удаленном выходе при отключении клиента, что можно увидеть в файле журнала сервера:

```
SIGTERM[soft,remote-exit] received, client-instance exiting
```

Обратите внимание - эта проблема не возникает при использовании `proto tcp` поскольку сервер немедленно замечает разрыв TCP-соединения.

Интерфейс управления OpenVPN

Одним из наиболее мощных, но менее известных вариантов OpenVPN является интерфейс управления. Интерфейс управления доступен как на стороне сервера, так и на стороне клиента. На стороне сервера его можно использовать для сбора статистики, мониторинга и контроля подключенных клиентов, а также для выполнения других задач, связанных с управлением. На стороне клиента его можно использовать для запроса паролей, ввода информации прокси-сервера для установления соединения с сервером VPN, взаимодействия с устройством PKCS#11 и сбора статистики на стороне клиента.

Плагин OpenVPN для Linux NetworkManager широко использует интерфейс управления для управления запуском и отключением VPN-подключения.

Чтобы использовать интерфейс управления - добавьте строку `management 127.0.0.1 23000 stdin` в файл конфигурации клиента или сервера. Эта опция указывает OpenVPN настроить интерфейс управления на IP-адресе 127.0.0.1, порту 23000 и использовать `stdin` для указания пароля управления.

Если мы добавим это в файл конфигурации `basic-udp-server.conf` и запустим сервер OpenVPN, то OpenVPN сначала запросит у нас пароль для управления:

```
[root@server]# ssh nikhef cat status.txt
[root@server]# openvpn --config movpn-04-08-server.conf
OpenVPN 2.3.2 x86_64-redhat-linux-gnu [SSL (OpenSSL)]
[LZO] [EPOLL] [PKCS11] [eurephia] [MH] [IPv6] built on Sep 12 2013
Enter Management Password:
MANAGEMENT: TCP Socket listening on [AF_INET]127.0.0.1:23000

[root@server]#
```

Затем мы можем использовать telnet для входа в интерфейс управления (ввод пользователя выделен жирным шрифтом):

```
[root#server]# telnet 127.0.0.1 23000
Trying 127.0.0.1...
Connected to 127.0.0.1
Escape character is '^]'.
ENTER PASSWORD:[password]
SUCCESS: password is correct
>INFO:OpenVPN Management Interface Version 1 -- type 'help' for more info
help
Management Interface for OpenVPN 2.3.2 x86_64-redhat-linux-gnu [SSL
(OpenSSL)]
[LZO] [EPOLL] [PKCS11] [eurephia] [MH] [IPv6] built on Sep 12 2013
Commands:
auth-retry t      : Auth failure retry mode
                    (none, interact, nointeract).
bytecount n       : Show bytes in/out, update every n secs
                    (0=off).
echo [on|off] [N|all] : Like log, but only show messages in echo
                      buffer.
exit|quit         : Close management session.
forget-passwords : Forget passwords entered so far.
help              : Print this message.
[...]
END
```

Этот необработанный интерфейс telnet можно использовать для просмотра состояния сервера - он предоставляет тот же вывод, что и опция `status` из предыдущего примера. Его также можно использовать для немедленного завершения клиентского соединения с помощью следующей команды:

```
kill client1
```

Это приведет к отключению клиента `client1`. Обратите внимание, что в большинстве случаев клиент автоматически пытается восстановить соединение. Теперь введите `exit`, чтобы завершить сеанс telnet.

Интерфейс управления может использоваться для управления OpenVPN различными способами (адаптировано со страницы руководства OpenVPN <https://community.openvpn.net/openvpn/wiki/Openvpn23ManPage>):

- `management IP port [pw-file]`: включить TCP-сервер на IP:порт для обработки функций управления демоном. `pw-file`, если он указан, является файлом паролей (пароль на первой строке) или `stdin` для стандартного ввода. Предоставленный пароль будет устанавливать пароль, который клиенты TCP должны будут указать для доступа к функциям управления.

Интерфейс управления также может прослушивать сокет домена Unix, если он поддерживается. Для использования доменного сокета укажите путь к сокету Unix вместо IP и установите для порта значение unix.

Интерфейс управления обеспечивает специальный режим, в котором канал управления TCP может работать через сам туннель. Чтобы включить этот режим - установите IP = "tunne1". Туннельный режим заставит интерфейс управления прослушивать TCP-соединение по локальному VPN-адресу интерфейса TUN/TAP.

- **management-client**: интерфейс управления будет подключаться как клиент домена TCP/Unix к IP:порт, заданный параметром --management, а не прослушивать как сервер TCP или в сокете домена Unix. Если клиентскому соединению не удается подключиться или он отключен - будет сгенерирован сигнал SIGTERM, что приведет к выходу OpenVPN.
- **management-query-passwords**: канал управления запросами для пароля с приватным ключом и --auth-user-pass имя_пользователя/пароль.
- **management-hold**: Запуск OpenVPN в режиме гибернации, пока клиент интерфейса управления не запустит его явно командой освобождения удержания.
- **management-signal**: отправляет сигнал SIGUSR1 в OpenVPN если сеанс управления отключен. Это полезно, когда вы хотите отключать сеанс OpenVPN при выходе пользователя из системы.
- **management-client-auth**: дает клиенту интерфейса управления ответственность за проверку подлинности клиентов после проверки их клиентского сертификата.

Пересогласование ключей сеанса

Чтобы обеспечить безопасность каждого соединения OpenVPN, сервер периодически пересматривает секретный ключ для канала данных с каждым клиентом. Это контролируется с помощью трех опций:

- **reneg-sec N**: повторно согласовать ключ канала данных через N секунд (по умолчанию 3600)
- **reneg-bytes N**: пересмотреть ключ канала данных после N байт (по умолчанию = 0 = выкл.)
- **reneg-pkts N**: пересмотреть ключ канала данных после N пакетов (по умолчанию = 0 = выкл.)

Если VPN-клиент испытывает периодические таймауты при подключении к серверу - часто бывает полезно изменить эти параметры. Однако если вы установите параметр renege-sec на очень короткий интервал - производительность VPN будет сильно ухудшена.

Параметры renege могут быть указаны либо на стороне клиента, либо на стороне сервера, либо на обоих. Опция renege, запускаемая наиболее часто с обеих сторон - сбрасывает счетчики на обоих концах. Если сервер указывает значение renege-sec 500, а клиент - renege-sec 60, то повторное согласование канала данных будет происходить примерно каждые 60 секунд.

Мы создадим пример, добавив три строки в файл конфигурации basic-udp-server.conf:

```
reneg-sec 10
reneg-pkts 1000
reneg-bytes 1000000
```

Сохраним конфигурацию как `tnovpn-04-09-server.conf` и восстановим VPN-соединение. Журнал сервера теперь будет содержать много строк с указанием TLS: `soft reset`:

```
Tue Oct 21 16:53:29 2014 <IP>:41679 [client1] Peer Connection  
Initiated with [AF_INET]<IP>:41679  
[...]  
Tue Oct 21 16:53:39 2014 client1/<IP>:41679 TLS: soft reset sec=0  
bytes=0/100000 pkts=0/100  
[...]  
Tue Oct 21 16:53:49 2014 client1/<IP>:41679 TLS: soft reset  
sec=0bytes=53/100000 pkts=1/100  
[...]  
Tue Oct 21 16:53:59 2014 client1/<IP>:41679 TLS: soft reset sec=0  
bytes=105/100000 pkts=2/100
```

При установленном параметре `reneg-sec 10` из временных меток журнала сервера видно, что ключ канала данных пересматривается каждые 10 секунд.

На стороне клиента мы также можем увидеть влияние, которое это пересогласование ключей оказывает на производительность VPN-соединения. Запустив простую команду `ping` после установления соединения - мы можем увидеть когда происходит повторное согласование ключа, основываясь на пиках времени отклика `ping`:

```
[client]$ ping 10.200.0.1  
PING 10.200.0.1 (10.200.0.1) 56(84) bytes of data.  
64 bytes from 10.200.0.1: icmp_seq=1 ttl=64 time=3.29 ms  
64 bytes from 10.200.0.1: icmp_seq=2 ttl=64 time=3.55 ms  
64 bytes from 10.200.0.1: icmp_seq=3 ttl=64 time=61.6 ms  
64 bytes from 10.200.0.1: icmp_seq=4 ttl=64 time=16.6 ms  
64 bytes from 10.200.0.1: icmp_seq=5 ttl=64 time=3.23 ms  
64 bytes from 10.200.0.1: icmp_seq=6 ttl=64 time=3.22 ms  
64 bytes from 10.200.0.1: icmp_seq=7 ttl=64 time=3.74 ms  
64 bytes from 10.200.0.1: icmp_seq=8 ttl=64 time=3.25 ms  
64 bytes from 10.200.0.1: icmp_seq=9 ttl=64 time=3.21 ms  
64 bytes from 10.200.0.1: icmp_seq=10 ttl=64 time=3.26 ms  
64 bytes from 10.200.0.1: icmp_seq=11 ttl=64 time=3.26 ms  
64 bytes from 10.200.0.1: icmp_seq=12 ttl=64 time=3.55 ms  
64 bytes from 10.200.0.1: icmp_seq=13 ttl=64 time=3.27 ms  
64 bytes from 10.200.0.1: icmp_seq=14 ttl=64 time=3.26 ms  
64 bytes from 10.200.0.1: icmp_seq=15 ttl=64 time=3.31 ms  
64 bytes from 10.200.0.1: icmp_seq=16 ttl=64 time=3.28 ms  
64 bytes from 10.200.0.1: icmp_seq=17 ttl=64 time=77.1 ms  
...
```

То же самое произойдет, когда *граница* пакета или байта будет пересечена, и в этот момент ключи канала данных также будут пересмотрены.

Замечание об устройствах PKCS#11

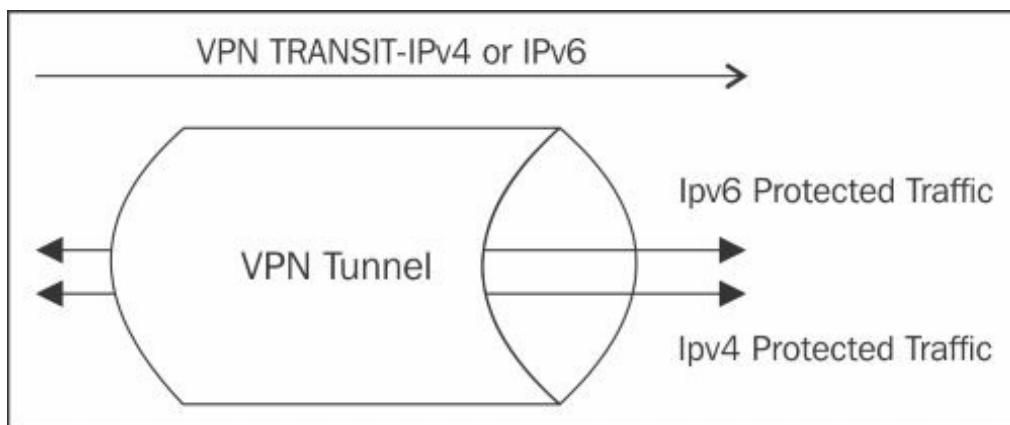
Повторное согласование ключей может быть особенно громоздким при использовании устройств PKCS#11. Некоторые устройства PKCS#11 накладывают большие расходы на повторное согласование ключа, в результате чего процесс пересогласования занимает несколько секунд. В течение этого времени VPN не отвечает.

Установка значения `reneg-sec` равным 0 эффективно отключит пересогласование ключа, но это делает саму VPN восприимчивой к атакам типа атаки посредника и атаки по времени, что делает дополнительную безопасность использования аппаратного защитного устройства бесполезной.

Использование IPv6

В OpenVPN 2.3 появилась надежная поддержка IPv6 как внутри туннеля OpenVPN, так и для транзита самого туннеля. OpenVPN вплоть до 1.x имел элементарную поддержку IPv6, которая была в значительной степени переписана. В целом, внутри туннеля OpenVPN администратор может выбрать поддержку Ethernet (уровень 2), IPv4 (уровень 3) и IPv6 (уровень 3).

Диаграмма иллюстрирует логическую взаимосвязь путей транзитной сети и путей защищенной сети. Необходимо использовать только один метод транзита, а одна конфигурация OpenVPN может содержать записи `--remote` как для IPv4, так и IPv6. Весь трафик, независимо от типа, будет защищен в туннеле. Вполне приемлемо иметь туннель полностью IPv6, использующий IPv6 как для транзитного, так и для защищенного трафика. Благодаря дополнительной маршрутизации и проксированию даже возможно использовать OpenVPN для помощи в преобразовании IPv6 в IPv4.



Защищенный трафик IPv6

Основываясь на примерах из предыдущего раздела, мы можем предоставлять клиентам адреса IPv6 и защищать этот трафик в туннеле. Для этого мы добавляем параметр `--server-ipv6` в конфигурацию нашего сервера. Это работает аналогично директиве `--server`, только для IPv6 вместо IPv4 и принимает в качестве аргументов сетевой адрес IPv6 и маску сети. Как и `--server`, `--server-ipv6` - это макрос для других параметров, которые могут передаваться индивидуально: `--ifconfig-ipv6`, `--ifconfig-ipv6-pool`, `--tun-ipv6` и `-push tun-ipv6`.

Как и в случае IPv4, маршруты IPv6 могут передаваться из конфигурации основного сервера или из файлов конфигурации для каждого клиента в каталоге `client-config`. В нашем примере мы просто отправим маршрут по умолчанию для всего трафика IPv6.

В настоящее время в OpenVPN для IPv6 нет опции перенаправления шлюза. Маршруты добавляются аналогично IPv6, с ключевым словом `route-ipv6` вместо `route`.

Теперь измените файл `movpn-04-01-server.conf` и добавьте инструкцию `--server-ipv6`:

```
proto udp
port 1194
dev tun
server 10.200.0.0 255.255.255.0
server-ipv6 2001:DB8:100::/64
push "route-ipv6 ::/0"
topology subnet
persist-key
persist-tun
keepalive 10 60

dh      /etc/openvpn/movpn/dh2048.pem
ca      /etc/openvpn/movpn/movpn-ca.crt
```

```

cert      /etc/openvpn/movpn/server.crt
key       /etc/openvpn/movpn/server.key

user nobody
group nobody

verb 3
daemon
log-append /var/log/openvpn.log

```

После сохранения файла требуется перезапуск процесса сервера OpenVPN. Если процесс начался правильно с новой опцией - вы должны увидеть что-то вроде этого в вашем файле журнала:

```
IFCONFIG POOL IPv6: (IPv4) size=252, size_ipv6=65536, netbits=64,
base_ipv6=2001:db8:100::1000
```

На этом этапе клиенты могут передавать трафик, используя IPv6 внутри туннеля, и сервер передает маршрут по умолчанию клиентам для IPv6. Добавление параметров конфигурации сервера не требует дополнительных соответствующих параметров в конфигурациях клиента. Подключенный клиент будет отображать адреса IPv4 и IPv6 на интерфейсе tunX.

Вот пример FreeBSD:

```

utun1: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1331
    inet 10.200.0.2 --> 10.200.0.2 netmask 0xffffffff00
    inet6 fe80::5ab0:35ff:fef5:811f%utun1 prefixlen 64 scopeid 0x9
    inet6 2001:db8:100::1001 prefixlen 64
    nd6 options=1<PERFORMNUD>

```

Вот пример для Windows 7:

```

Administrator: C:\Windows\System32\cmd.exe
C:\Windows\system32>ipconfig

Windows IP Configuration

Ethernet adapter OpenVPN:
  Connection-specific DNS Suffix . . . . . : 2001:db8:100::1000
  IPv6 Address . . . . . : fe80::7c7f:fc20:27b0:5579%15
  Link-local IPv6 Address . . . . . : fe80::7c7f:fc20:27b0:5579%15
  IPv4 Address . . . . . : 10.200.0.2
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : fe80::8%15

Ethernet adapter Bridge Ethernet:
  Connection-specific DNS Suffix . . . . . :
  Link-local IPv6 Address . . . . . : fe80::5544:8add:f72e:1820%13
  Autoconfiguration IPv4 Address . . . . . : 169.254.24.32
  Subnet Mask . . . . . : 255.255.0.0

```

Обратите внимание, что всплывающее окно от наведения на значок на панели задач не отображает IPv6-адрес, но команда ipconfig в терминале показывает оба адреса.



Использование IPv6 в качестве транзита

OpenVPN в настоящее время не имеет возможности прослушивать адреса IPv4 и IPv6

одновременно, но большинство современных ядер могут справиться с этим с помощью сопоставленных IPv6-адресов. Как это делается: берется IP-адрес версии 4, такой как 192.168.200.4 и отображается как IPv6-адрес ::ffff:192:168.:200::4. Кроме того, вместо proto udp будет proto udp6 как на клиенте так и на сервере.

После изменения оператора proto из нашего примера конфигурации инициализируется client2 и мы видим что адрес назначен. Оба адреса сервера OpenVPN v4 и v6 могут быть проверены, и мы можем подтвердить с помощью tcpdump что транзит через туннель осуществляется через IPv6.

Вот клиентский интерфейс tun:

```
ecrist@phillip:~-> ifconfig tun4
tun4: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> metric 0 mtu
1500
    options=80000<LINKSTATE>
    inet6 fe80::216:3eff:fe09:5d4e%tun4 prefixlen 64 scopeid 0x9
    inet 10.200.0.2 --> 10.200.0.2 netmask 0xffffffff
    inet6 2001:db8:100::1000 prefixlen 64
    nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
    Opened by PID 45391
```

Вот IPv4 внутри туннельного пинга:

```
ecrist@phillip:~-> ping -c 1 10.200.0.1
PING 10.200.0.1 (10.200.0.1): 56 data bytes
64 bytes from 10.200.0.1: icmp_seq=0 ttl=64 time=0.490 ms

--- 10.200.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.490/0.490/0.490/0.000 ms
```

Вот IPv6 внутри туннельного пинга:

```
ecrist@phillip:~-> ping6 -c 1 2001:db8:100::1
PING6(56=40+8+8 bytes) 2001:db8:100::1000 --> 2001:db8:100::1
16 bytes from 2001:db8:100::1, icmp_seq=0 hlim=64 time=0.591 ms

--- 2001:db8:100::1 ping6 statistics ---
1 packets transmitted, 1 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 0.591/0.591/0.591/0.000 ms
```

Вот вывод tcpdump (обратите внимание на ключевое слово IPv6 в выводе):

```
root@terrance:/usr/local/etc/openvpn-> tcpdump -i xn0 host
2001:db8:5555:5555::1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on xn0, link-type EN10MB (Ethernet), capture size 65535 bytes
19:14:05.449553 IP6 phillip.1194 > terrance.1194: UDP, length 53
19:14:05.449692 IP6 terrance.1194 > phillip.1194: UDP, length 53
19:14:08.389222 IP6 phillip.1194 > terrance.1194: UDP, length 93
19:14:08.389394 IP6 terrance.1194 > phillip.1194: UDP, length 93
19:14:09.389858 IP6 phillip.1194 > terrance.1194: UDP, length 93
```

Расширенные параметры конфигурации

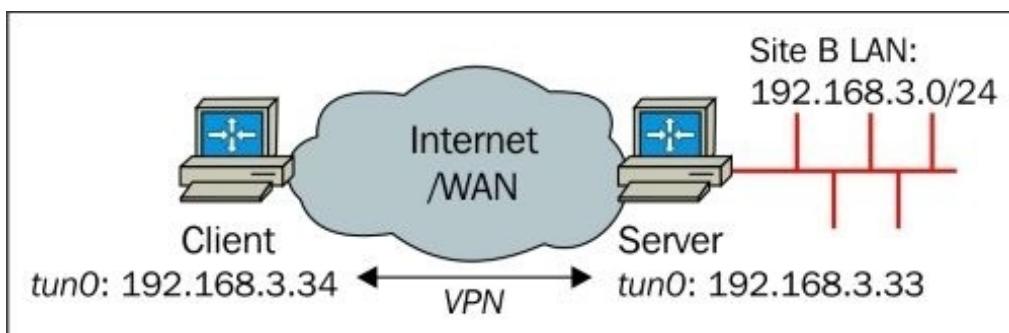
Следующие несколько разделов иллюстрируют некоторые дополнительные параметры конфигурации. Предполагается, что вы полностью понимаете их влияние на сеть, прежде чем применять их в производственной среде. Эти варианты редко используются, но могут быть чрезвычайно полезны в нужных обстоятельствах.

ARP-прокси

Часто желательно чтобы VPN-клиенты выглядели так, как будто они являются частью серверной сети. Это облегчает просмотр папок и обмен файлами и принтерами. Для достижения этой цели, в некоторых установках прибегают к Ethernet-бридже (см [Главу 6](#), Режим клиент/сервер с tap-устройствами), которое имеет свои недостатки. Производительность в мостовой конфигурации может быть значительно ниже по сравнению с немостовой установкой.

Когда сервер OpenVPN работает в Linux или Unix - существует альтернативное решение: большинство ядер Unix имеют возможности Proxy ARP, которую можно использовать для назначения клиента OpenVPN с IP-адресом в локальной сети на стороне сервера и отображения его так, как если бы он был частью этой локальной сети. Обратите внимание - это работает только для сетей IPv4, так как сети IPv6 не используют ARP.

Рассмотрим следующую схему сети:



В этом макете *стандартная* подсеть VPN 10.200.0.0/24 не может быть использована, так как мы должны интегрировать VPN-клиентов в существующую подсеть, для данного примера - это **192.168.3.0/24**. Текущие машины в этой подсети находятся в диапазоне 192.168.3.10 - 192.168.3.24, таким образом, мы разместим VPN-адреса немного за пределами этого диапазона. Убедитесь, что адреса VPN не должны устанавливаться сервером DHCP в локальной сети на стороне сервера, поскольку мы хотим чтобы OpenVPN назначал адреса для клиентов VPN.

В этом примере мы будем использовать возможность OpenVPN для запуска скриптов при подключении клиента или отключения. Скриптовые способности OpenVPN объясняются более подробно в [Главе 7, Скрипты и плагины](#).

1. Начнем со следующего файла конфигурации сервера:

```
proto udp
port 1194
dev tun
server 192.168.3.32 255.255.255.224
push "route 192.168.3.0 255.255.255.0"
topology subnet
persist-key
persist-tun
keepalive 10 60

tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

verb 3
daemon
log-append /var/log/openvpn.log
```

```
script-security 2
client-connect /etc/openvpn/movpn/proxyarp-connect.sh
client-disconnect /etc/openvpn/movpn/proxyarp-disconnect.sh
```

Обратите внимание - мы добавили три оператора для установки уровня безопасности для скриптов и запуска собственного скрипта всякий раз, когда клиент подключается или отключается.

2. Сохраните этот файл конфигурации как `movpn-04-10-server.conf`.
3. Затем создайте скрипт `proxyarp-connect.sh`, выполняемый при каждом подключении VPN-клиента:

```
#!/bin/bash
/sbin/arp -i eth0 -Ds ${ifconfig_pool_remote_ip} eth0 pub
/sbin/ip route add ${ifconfig_pool_remote_ip}/32 dev tun0
```

4. Сохраните скрипт как `/etc/openvpn/movpn/proxyarp-connect.sh`. Расположение скрипта должно соответствовать абсолютному пути, указанному в файле `movpn-04-10-server.conf`.
 5. Затем создайте скрипт `proxyarp-disconnect.sh`, выполняемый при отключении клиента:
- ```
#!/bin/bash
/sbin/arp -i eth0 -d ${ifconfig_pool_remote_ip}
/sbin/ip route del ${ifconfig_pool_remote_ip}/32 dev tun0
```

6. Сохраните скрипт как `/etc/openvpn/movpn/proxyarp-disconnect.sh`.

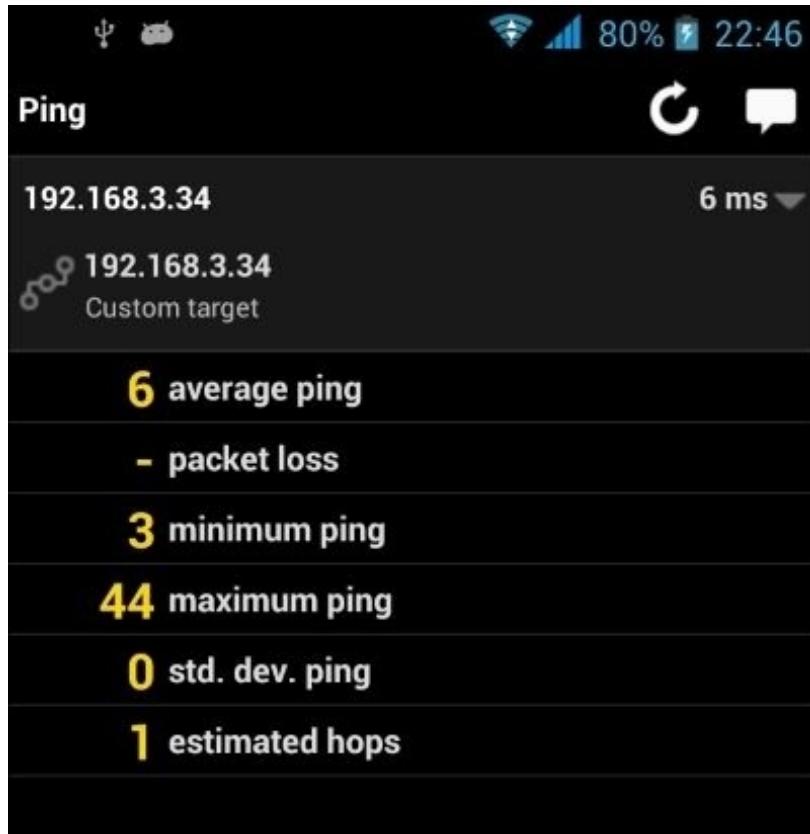
### Заметка

Имена устройств `eth0` и `tun0` жестко заданы в скрипте. Это необходимо, поскольку устройство, на котором должен быть задан дополнительный ARP-адрес, неизвестно OpenVPN. Также возможно опубликовать дополнительный ARP-адрес на нескольких интерфейсах (`eth0`, `eth1`, `wlan0` и т.д.), дублируя строку `/sbin/arp` в обоих скриптах.

Сделайте оба скрипта исполняемыми и запустите сервер OpenVPN, используя следующие команды:

```
[root@server]# chmod a+x /etc/openvpn/movpn/proxyarp-connect.sh
[root@server]# openvpn --config movpn-04-10-server.conf
```

7. Как всегда, используйте файл конфигурации `basic-udp-client.conf` (или `basic-udp-client.ovpn`) для подключения к серверу. После успешного подключения VPN-клиента мы проверяем, что клиент **виден** другими устройствами в локальной сети. Для этого мы использовали смартфон Android с установленным приложением Ping:



### Заметка

На устройстве Android не было добавлено никаких дополнительных сетевых маршрутов. Клиент VPN действительно интегрирован в существующую подсеть.

8. Мы также можем проверить, что компьютер сервера OpenVPN теперь публикует дополнительный IP-адрес в своих таблицах ARP:

```
[server]$ /sbin/arp -an | grep PERM
? (192.168.3.34) at * PERM PUP on eth0
```

### Как работает Proxy ARP?

Proxy ARP - это функция, поддерживаемая большинством ядер Unix и Linux. Чаще всего она используется для подключения клиентов удаленного доступа к локальной сети, а в настоящее время также ADSL и провайдерами кабельного Интернета.

Сервер OpenVPN заимствует IP-адрес из диапазона локальной сети при подключении клиента. Этот IP-адрес затем назначается клиенту OpenVPN. Сервер также создает специальную запись в ARP-таблицах системы, чтобы сообщить остальным локальным сетям филиала B, что сервер OpenVPN действует как прокси для IP 192.168.3.34. Это означает, что когда другая машина в локальной сети на стороне сервера хочет знать где найти хост с IP 192.168.3.34 - сервер OpenVPN ответит своим собственным MAC-адресом интерфейса, на котором был присвоен адрес Proxy ARP.

Файл конфигурации сервера содержит несколько операторов, требующих пояснения:

```
server 192.168.3.32 255.255.255.224
push "route 192.168.3.0 255.255.255.0"
```

Предыдущие строки заставляют сервер OpenVPN назначить адрес 192.168.3.33 VPN серверу, с маской сети 255.255.255.224 (или 27). Первому VPN-клиенту назначается адрес 192.168.3.34/27. Однако это означает, что сам VPN-клиент не может получить никаких IP-адресов за пределами

этого диапазона. Оператор `push route` необходим чтобы сообщить клиенту OpenVPN, что вся подсеть 192.168.3.0/24 доступна через VPN.

Пока что файл конфигурации сервера содержит следующие строки:

```
user nobody
group nobody
```

В этом файле конфигурации они отсутствуют, так как скрипты `client-connect` и `client-disconnect` должны запускаться от пользователя `root`. Альтернативный подход состоит в том, чтобы настроить права `sudo` для выполнения пользователем `user nobody` команды `/sbin/arp` с привилегиями `root`.

Наконец, строки:

```
script-security 2
client-connect /etc/openvpn/movpn/proxyarp-connect.sh
client-disconnect /etc/openvpn/movpn/proxyarp-disconnect.sh
```

Настроим функции скриптов OpenVPN. Первая строка устанавливает уровень безопасности скриптов равным 2: это означает что определенные переменные среды доступны для скрипта.

В строках `client-connect` и `client-disconnect` указан абсолютный путь к выполняемым скриптам.

## Назначение публичных IP-адресов клиентам

В качестве продолжения примера Proxy ARP теперь рассмотрим как возможно раздавать публичные адреса IPv4 клиентам OpenVPN. Давайте предположим, что нам доступен следующий набор (только для примера) публичных IPv4-адресов:

Наша общедоступная сеть IPv4 - 192.0.2.160/28, что дает нам 16 адресов. Эти адреса используются следующим образом:

| IP-адрес                | Использование                                   |
|-------------------------|-------------------------------------------------|
| 192.0.2.160             | Сетевой адрес подсети.                          |
| 192.0.2.161             | Используется для IP-адреса VPN сервера          |
| 192.0.2.162             | Недоступен                                      |
| 192.0.2.163             | Недоступен                                      |
| 192.0.2.164-192.0.2.170 | Доступны для клиентов VPN                       |
| 192.0.2.171             | Это адрес локальной сети самого сервера OpenVPN |
| 192.0.2.172             | Недоступен                                      |
| 192.0.2.173             | Недоступен                                      |
| 192.0.2.174             | Маршрутизатор для удаленной локальной сети      |
| 192.0.2.175             | Сетевой широковещательный адрес                 |

Теперь мы должны настроить сервер OpenVPN, который может раздавать адреса со 192.0.2.164 по 192.0.2.170, а сам сервер OpenVPN по адресу 192.0.2.161:

1. Сначала мы создаем файл конфигурации сервера:

```
proto udp
port 1194
dev tun

mode server
tls-server
ifconfig 192.0.2.161 255.255.255.240
```

```

ifconfig-pool 192.0.2.164 192.0.2.170
push "route 192.0.2.171 255.255.255.255 net_gateway"
push "route-gateway 192.0.2.174"
push "redirect-gateway def1"
push "topology subnet"

topology subnet
persist-key
persist-tun
keepalive 10 60

tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

verb 3
daemon
log-append /var/log/openvpn.log

script-security 2
client-connect /etc/openvpn/movpn/proxyarp-connect.sh
client-disconnect /etc/openvpn/movpn/proxyarp-disconnect.sh

```

- Сохраните этот файл как `movpn-04-11-server.conf`. Мы будем повторно использовать сценарии `proxyarp-connect.sh` и `proxyarp-disconnect.sh` из предыдущего примера. Создайте `proxyarp-connect.sh`, который выполняется при каждом подключении VPN-клиента:

```

#!/bin/bash
/sbin/arp -i eth0 -Ds ${ifconfig_pool_remote_ip} eth0 pub
/sbin/ip route add ${ifconfig_pool_remote_ip}/32 dev tun0

```

- Сохраните его как `/etc/openvpn/movpn/proxyarp-connect.sh`.
- Затем создайте скрипт `proxyarp-disconnect.sh`, выполняемый при отключении клиента:

```

#!/bin/bash
/sbin/arp -i eth0 -d ${ifconfig_pool_remote_ip}
/sbin/ip route del ${ifconfig_pool_remote_ip}/32 dev tun0

```

- Сохраните его как `/etc/openvpn/movpn/proxyarp-disconnect.sh`. Сделайте оба скрипта исполняемыми файлами и запустите сервер OpenVPN:

```

[root@server]# chmod a+x /etc/openvpn/movpn/proxyarp-connect.sh[
[root@server]# openvpn --config movpn-04-11-server.conf

```

- Используйте файл базовой конфигурации для подключения клиента OpenVPN к серверу. Первому клиенту будет присвоен адрес 192.0.2.164.

Проверьте IP-адрес первого клиента, перейдя по адресу <https://www.whatismyip.com>.

Файл конфигурации сервера аналогичен файлу `movpn-04-10-server.conf` за исключением этого блока:

```

mode server
tls-server

```

```
ifconfig 192.0.2.161 255.255.255.240
ifconfig-pool 192.0.2.164 192.0.2.170
push "route 192.0.2.171 255.255.255.255 net_gateway"
push "route-gateway 192.0.2.174"
push "redirect-gateway def1"
push "topology subnet"
```

Ранее в этой главе объяснялось, что макрос `server 10.200.0.0 255.255.255.0` расширяется следующим образом:

```
mode server
tls-server
push "topology subnet"

ifconfig 10.200.0.1 255.255.255.0
ifconfig-pool 10.200.0.2 10.200.0.254 255.255.255.0
push "route-gateway 10.200.0.1"
```

Раздавая публичные IP-адреса, мы не можем позволить себе роскошь тратить IPv4-адреса. Опция `topology subnet`, представленная в OpenVPN 2.1, окажется здесь весьма полезной.

Изучив наше общедоступное пространство IPv4, мы оставляем оператор `server` и включаем нашу собственную версию параметров `ifconfig` и `ifconfig-pool`:

- `ifconfig 192.0.2.161 255.255.255.240`: указывает IP-адрес VPN-сервера - 192.0.2.161/28.
- `ifconfig-pool 192.0.2.164 192.0.2.170`: указывает, что пул доступных IP-адресов для VPN-клиентов: от 192.0.2.164 до 192.0.2.170, всего семь адресов. Обратите внимание, что диапазон `ifconfig-pool` должен быть непрерывным.

## Заметка

Если есть «дыры» в диапазоне, то часто бывает проще назначать IP-адреса с помощью скрипта, как мы узнаем в [Главе 7, Скрипты и плагины](#).

- `push "route 192.0.2.171 255.255.255.255 net_gateway"`: маршрут к адресу локальной сети VPN-сервера, который необходимо явно передать клиентам. Обычно этот маршрут автоматически добавляется клиентом OpenVPN для гарантии, что трафик, предназначенный для самого сервера OpenVPN, не будет снова введен в туннель, что вызовет цикл обработки пакетов. С нашей специальной настройкой `ifconfig` и `ifconfig-pool` желательно добавить явный маршрут к адресу локальной сети сервера OpenVPN.
- `push "route-gateway 192.0.2.174"`: `route-gateway` указывает адрес шлюза, используемого для направления всего туннельного трафика. Обычно `route-gateway` соответствует IP-адресу VPN-сервера. В таком случае это вызовет только скачок маршрута, поскольку VPN-сервер немедленно перенаправит его на реальный шлюз 192.0.2.174, который находится в той же подсети. Следовательно, мы указываем IP-адрес шлюза LAN.
- `push "redirect-gateway def1"`: чтобы клиент OpenVPN использовал публичный адрес для всего своего трафика - он должен направить весь трафик через VPN-туннель.
- `push "topology subnet"`: как правило, макрос сервера позаботится об этом `push` за нас, но так как мы здесь не используем макрос сервера, то должны явно передать эту опцию. Если этот параметр опущен, то сервер будет назначать адреса линейно (поскольку в его конфигурации есть строка `topology subnet`), однако клиенты VPN

будут предполагать, что им назначены адреса топологии net30, что является текущим значением по умолчанию. Явный push обходит эту потенциально-неверную конфигурацию.

## Резюме

В этой главе были рассмотрены различные функции и опции режима клиент-сервер с устройствами tun. Мы установили базовый набор файлов конфигурации как для сервера OpenVPN, так и для клиента, как для протокола UDP, так и TCP в качестве транспорта, так и для клиентов Windows и Linux/Unix. Этот набор основных файлов конфигурации будет использоваться на протяжении всей книги.

Мы обсудили, как настроить сервер OpenVPN, обслуживающий как адреса IPv4, так и адреса IPv6. Мы рассмотрели маршрутизацию на стороне сервера и на стороне клиента, включая перенаправление всего трафика через VPN-туннель. Также мы увидели как раздавать публичные IPv4-адреса с помощью OpenVPN.

В следующей главе мы рассмотрим расширенные функции, предлагаемые OpenVPN. Кроме того, в [Главе 6, Режим клиент/сервер с tap-устройствами](#) будут объяснены несколько вариантов и примеры, что полезно также для режима tun.

# Глава 5. Расширенные сценарии развертывания в tun режиме

Базовая конфигурация VPN относительно проста, но интеграция этого VPN с остальной частью сети является гораздо более сложной задачей. В этой главе мы рассмотрим некоторые расширенные сценарии развертывания OpenVPN, выходящие за рамки базовой установки и настройки VPN. Некоторые из этих сценариев основаны на реальных вопросах пользователей из списков рассылки OpenVPN, форума и канала IRC. Мы рассмотрим следующие темы:

- Включение (Windows) общего доступа к файлам через VPN
- Интеграция с механизмами внутренней аутентификации, такими как PAM и LDAP
- Фильтрация VPN-трафика (межсетевой экран)
- Маршрутизация на основе политик для повышения безопасности
- Работа с общедоступными и частными сетевыми адаптерами в Windows 7
- Использование OpenVPN с прокси HTTP или SOCKS

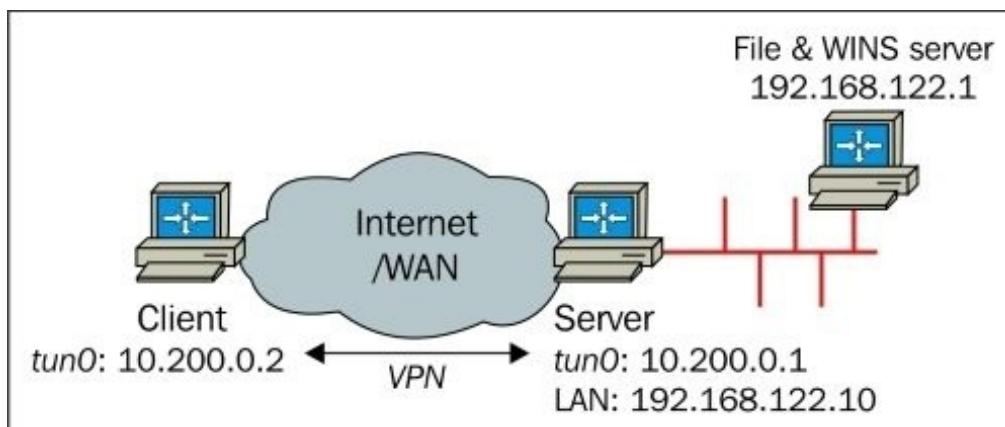
Примеры, представленные в этой главе, основаны на примерах из предыдущей главы, [Глава 4, Режим клиент/сервер с устройствами tun](#). В частности, будут использоваться базовые файлы конфигурации производственного уровня и добавление дополнительных разделов безопасности.

## Включение общего доступа к файлам через VPN

Как указано в разделе [Маршрутизация и маршрутизация на стороне сервера](#) в предыдущей главе, VPN действительно полезен только тогда, когда клиенты VPN имеют доступ к ресурсам на стороне сервера. Для доступа к этим серверным ресурсам необходима маршрутизация. Она обеспечивает правильный поток сетевого трафика между серверной локальной сетью и VPN.

Одним из наиболее распространенных вариантов использования VPN является предоставление удаленным работникам доступа к ресурсам в корпоративной сети. Файлы в корпоративной сети часто хранятся на файловом сервере под управлением Windows. Для просмотра общих файловых ресурсов Windows с использованием сетевых имен потребуется сервер WINS.

Опять же, очень распространенная схема доступа к ресурсам в сети на стороне сервера изображена здесь:



Локальная сеть на стороне сервера - **192.168.122.0/24**, и в этой подсети расположены ресурсы, к которым VPN-клиенты должны получить доступ.

Мы начнем с файла `basic-udp-server.conf` и добавим три строки:

```

proto udp
port 1194
dev tun
server 10.200.0.0 255.255.255.0
topology subnet
persist-key
persist-tunkeepalive 10 60

remote-cert-tls client
tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

user nobody
group nobody # use 'group nogroup' on Debian/Ubuntu

verb 3
daemon
log-append /var/log/openvpn.log

push "route 192.168.122.0 255.255.255.0"
push "redirect-gateway"
push "dhcp-option WINS 192.168.122.1"

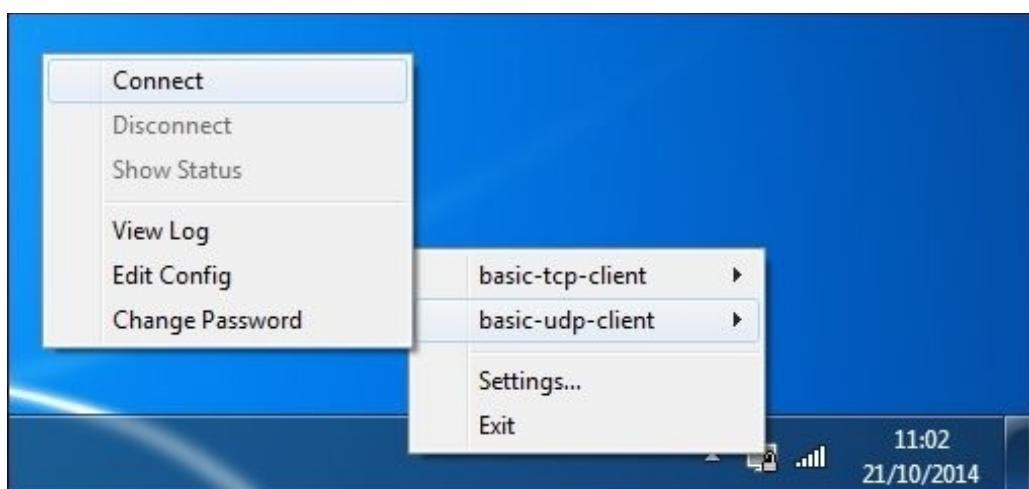
```

Сохраните этот файл как `movpn-05-01-server.conf`.

Первая дополнительная строка добавляет серверную локальную сеть в набор сетей, которые необходимо маршрутизировать через VPN. Вторая строка - перенаправляет весь сетевой трафик через VPN-туннель. Эта строка необходима для гарантии того, что адаптер OpenVPN TAP-Win считается **приватным**. Общий доступ к файлам возможен только при использовании **приватных** сетевых адаптеров (в отличие от **общественных** сетевых адаптеров). Последняя добавленная строка указывает серверу OpenVPN передать дополнительную опцию DHCP, содержащую IP-адрес сервера WINS, клиенту OpenVPN.

Мы запускаем сервер OpenVPN, используя этот файл конфигурации. Снова используем машину на базе Windows 7 Professional 64 бит в качестве клиента OpenVPN, на которой установлена версия OpenVPN 2.3.5-I001 X86\_64.

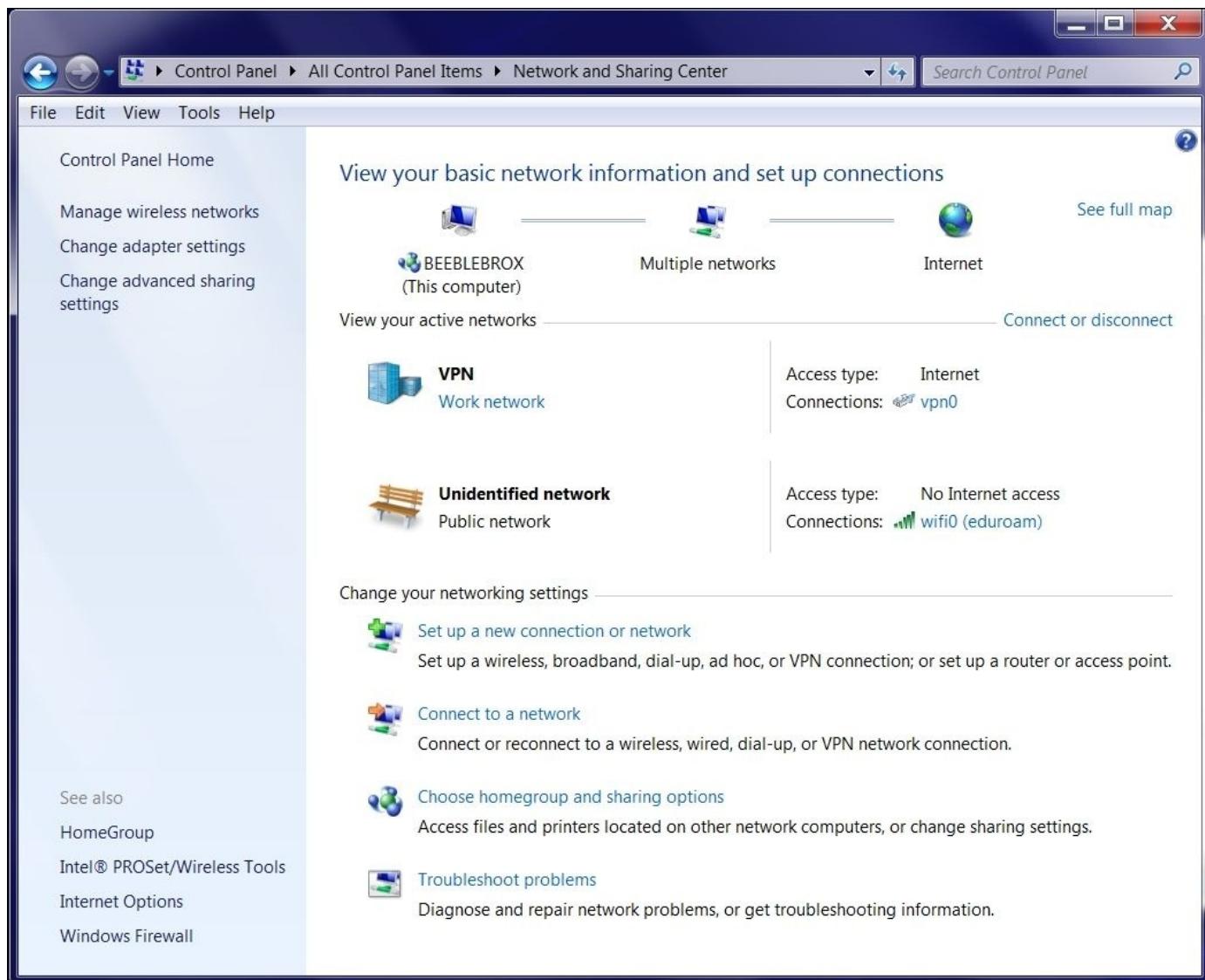
Запустите приложение OpenVPN GUI, выберите конфигурацию **basic-udp-client** и нажмите **Connect**.



Как мы делали в примере из раздела *Маршрутизация и маршрутизация на стороне сервера* в

предыдущей главе, мы гарантируем, что сервер OpenVPN пересыпает IP-трафик и что на шлюзе на стороне сервера добавлен дополнительный маршрут для правильной маршрутизации VPN-трафика обратно через VPN-сервер.

После того, как VPN-соединение было установлено, мы переходим к **Центру управления сетями и общим доступом** чтобы убедиться что адаптер TAP (названный vpn0 в следующем скриншоте) отмечен как **необщественный** и является частью либо **Домашней**, либо **Рабочей** сети/группы/места/. Если адаптер TAP помечен как **общественный** - это означает, что Windows не доверяет трафику, поступающему от этого адаптера и будет отказывать в обмене файлами через VPN. В разделе этой главы [Сетевые расположения Windows - общие и частные](#), мы подробно рассмотрим эту тему.



Как мы видим - соединение OpenVPN vpn0 является частью **Рабочей** сети - это означает что общий доступ к файлам разрешен.

Первый способ проверить работает ли общий доступ к файлам - это перейти к файловому серверу, используя его IP-адрес. Вы можете сделать это, открыв окно командной строки и введя следующие строки:

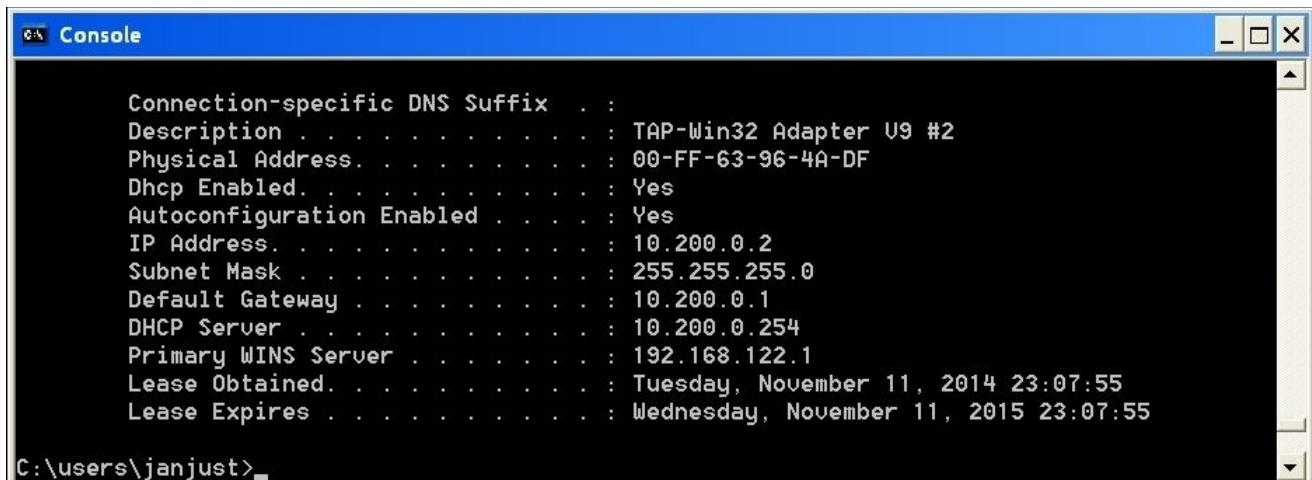
```
C:> start \\192.168.122.1
```

На этом этапе появится диалоговое окно аутентификации.

Введите свои учетные данные для файлового сервера. Теперь откроется окно проводника Windows с содержимым удаленных общих ресурсов.

## Использование имен NetBIOS

Вместо просмотра файловых ресурсов по их IP-адресам гораздо удобнее использовать сетевое имя файлового сервера Windows. Для этого клиенту Windows необходим адрес сервера WINS. Стока push "dhcp-option WINS 192.168.122.1" передает этот адрес WINS для всех подключающихся клиентов OpenVPN. После того, как VPN-соединение установлено, мы можем проверить - используется ли соответствующий WINS-сервер, введя команду ipconfig /all в командной оболочке.

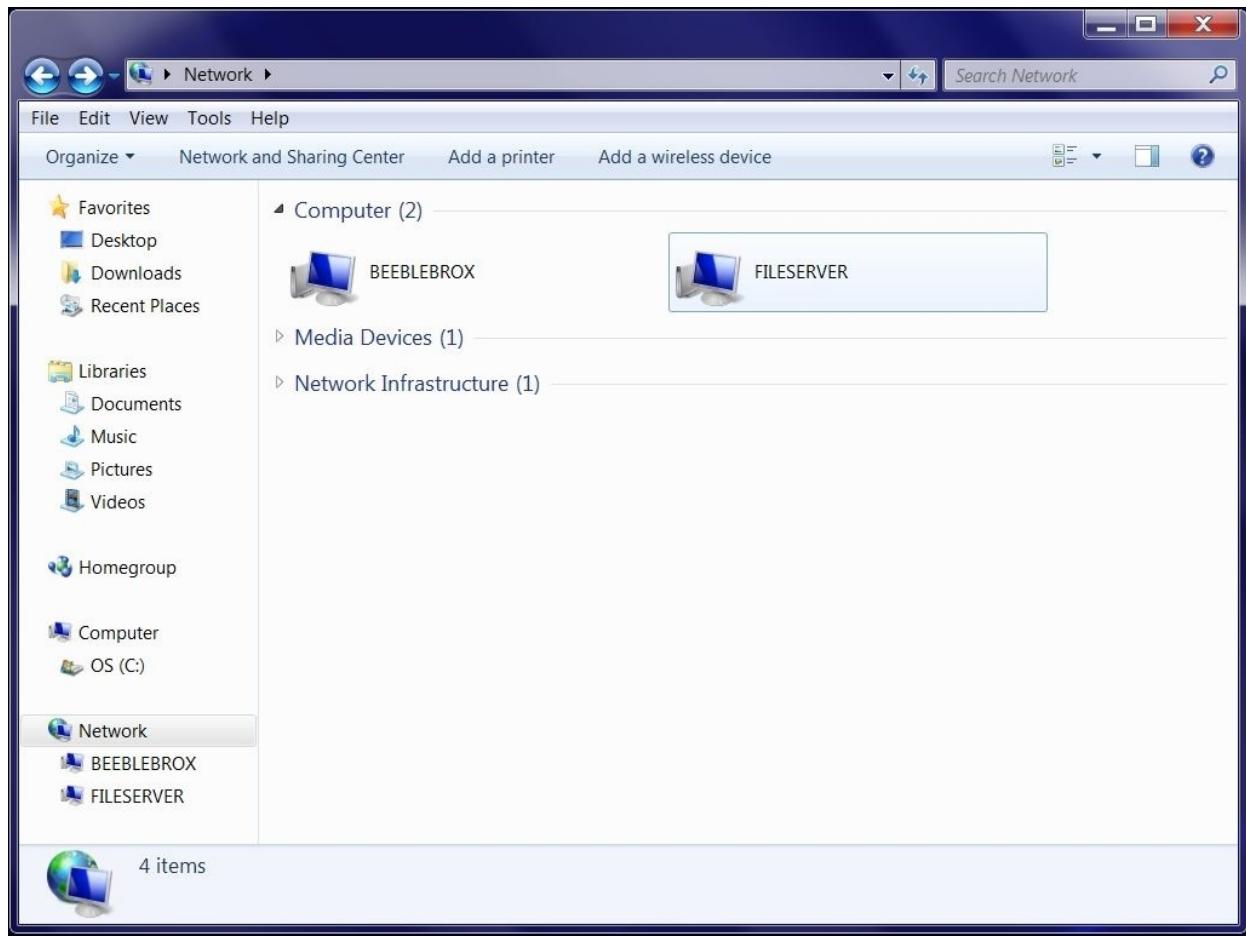


```
Connection-specific DNS Suffix . . .
Description : TAP-Win32 Adapter U9 #2
Physical Address : 00-FF-63-96-4A-DF
Dhcp Enabled. : Yes
Autoconfiguration Enabled : Yes
IP Address. : 10.200.0.2
Subnet Mask : 255.255.255.0
Default Gateway : 10.200.0.1
DHCP Server : 10.200.0.254
Primary WINS Server : 192.168.122.1
Lease Obtained. : Tuesday, November 11, 2014 23:07:55
Lease Expires : Wednesday, November 11, 2015 23:07:55
```

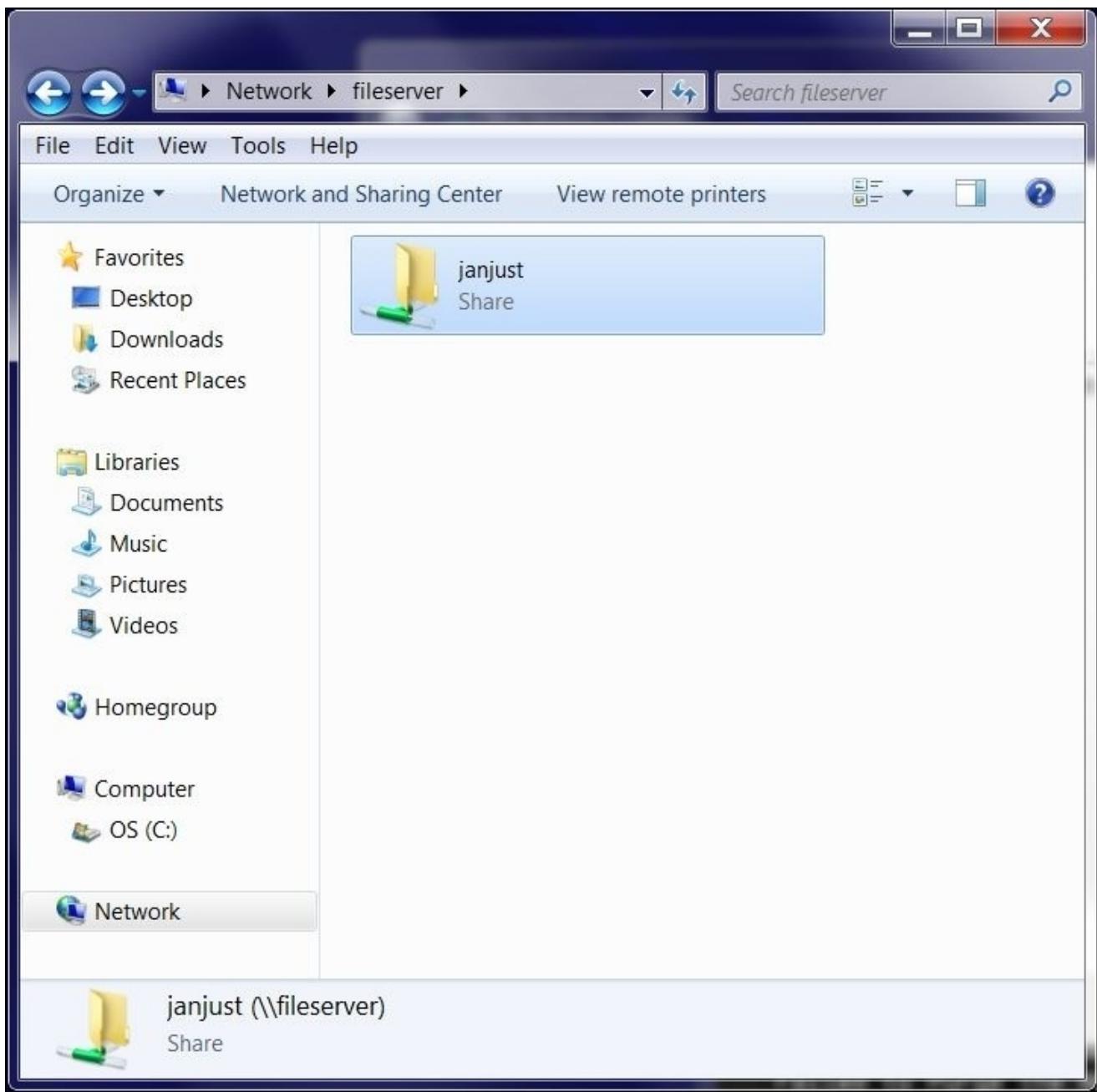
C:\users\janjust>

IP-адрес сервера WINS указан в строке **Primary WINS Server (Основной сервер WINS)**, которая указывает, что Windows будет использовать этот сервер для разрешения имен WINS.

Теперь, когда **Центр управления сетями и общим доступом** открыт - файловый сервер будет отображаться с именем NetBIOS (**FILESERVER**):



Когда мы нажмем на этот значок, то увидим доступные общие ресурсы на файловом сервере:



Когда мы снова нажмем на общий ресурс - появится диалоговое окно аутентификации. Введите свои учетные данные для файлового сервера. Откроется окно проводника Windows с содержимым удаленного общего ресурса как при использовании IP-адреса.

## Использование nbtstat для устранения проблем с подключением

Средство командной строки Windows **nbtstat** очень полезно при устранении проблем с совместным использованием файлов Windows. Вы можете найти имя Windows NetBIOS и просмотреть доступные общие ресурсы или найти имя NetBIOS, которое соответствует определенному IP-адресу. В обоих случаях вывод будет примерно таким:

The screenshot shows a Windows Command Prompt window titled "Command Prompt". The output displays the following information:

```
0x0 Command Prompt
upn0:
Node IpAddress: [10.200.0.2] Scope Id: []

NetBIOS Remote Machine Name Table

Name Type Status

FILESERVER <00> UNIQUE Registered
FILESERVER <03> UNIQUE Registered
FILESERVER <20> UNIQUE Registered
..._MSBROWSE_.<01> GROUP Registered
WORKGROUP <1B> UNIQUE Registered
WORKGROUP <1E> GROUP Registered
WORKGROUP <00> GROUP Registered

MAC Address = 00-00-00-00-00-00
```

## Использование LDAP в качестве механизма внутренней аутентификации

Обычно безопасность VPN основана на паре сертификат/закрытый ключ X.509, которой должны обладать все пользователи VPN для получения доступа. Безопасность вашей VPN может быть еще более повышена, если пользователям также необходимо будет указать имя пользователя и пароль при подключении к серверу OpenVPN.

На стороне сервера проверка имени пользователя и пароля может быть выполнена с использованием нескольких механизмов:

- Использование файла паролей на стороне сервера, который содержит имена пользователей и их хешированные пароли.
- Использование **PAM** (сокращение от **Pluggable Authentication Module**), который обычно включен во все операционные системы Linux/UNIX.
- Использование центрального сервера каталогов на основе **легкорасширяемого протокола доступа к каталогам (Lightweight Directory Access Protocol - LDAP)**. Обратите внимание, что LDAP и Active Directory также могут использоваться с различными модулями PAM.

Также возможно выполнить аутентификацию в домене Windows Active Directory, поскольку это очень похоже на использование автономного сервера LDAP. В нашем примере мы покажем как аутентифицировать пользователей на сервере LDAP.

Самый простой способ поддержки внутренней аутентификации LDAP - использовать модуль `openvpn-plugin-ldap`. В большинстве дистрибутивов Linux этот модуль необходимо устанавливать отдельно. Например, в системах на основе RPM вы должны использовать следующую команду:

```
sudo yum install openvpn-auth-ldap
```

Мы начнем с файла basic-udp-server.conf и добавим одну строку:

```
proto udp
port 1194
dev tun
server 10.200.0.0 255.255.255.0
topology subnet
persist-key
persist-tun
keepalive 10 60

remote-cert-tls client
tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

user nobody
group nobody # use 'group nogroup' on Debian/Ubuntu

verb 3
daemon
log-append /var/log/openvpn.log

plugin /usr/lib64/openvpn/plugin/lib/openvpn-authldap.so \
 "/etc/openvpn/movpn/movpn_ldap.conf"
```

Сохраните этот файл как movpn-05-02-server.conf и создайте файл movpn\_ldap.conf:

```
<LDAP>
 URL ldaps://ldap.example.org
 Timeout 15
 TLSEnable no
 FollowReferrals yes
 TLSCACertFile /etc/pki/tls/certs/ca-bundle.crt
 TLSCACertDir /etc/pki/tls/certs
</LDAP>

<Authorization>
 BaseDN "ou=LocalUsers,dc=example,dc=org"
 SearchFilter "((&(uid=%u)(authorizedService=login)))"
 RequireGroup false
</Authorization>
```

Это очень простой файл конфигурации authldap, использующий защищенный сервер LDAP, находящийся по адресу URI ldaps://ldap.example.org, порт 636 и фильтр поиска LDAP на основе идентификатора пользователя (uid=%u) и атрибута LDAP authorizedService=login. URI указывает на SSL-соединение с сервером с помощью службы ldaps://. Эти параметры сильно зависят от используемого сервера LDAP, но плагин openvpn-ldap-auth можно адаптировать практически к любой конфигурации. Например, в этой настройке для подключения к серверу LDAP не используется привязка. Тем не менее, этот, а также другие варианты подключения могут быть добавлены.

Далее мы добавляем в конфигурацию клиента basic-udp-client.ovpn строку:

**auth-user-pass**

Сохраните его как movpn-05-02-client.ovpn и запустите клиент. Сначала клиент устанавливает соединение с сервером, используя свой сертификат X.509 и файл приватного ключа, после чего пользователю предлагается ввести имя пользователя и пароль.

Если введены правильные учетные данные, то соединение будет установлено.

В противном случае сервер отказывает в доступе.

Вместо использования плагина `openvpn-ldap-auth`, мы также можем использовать плагин PAM. Затем OpenVPN запросит подсистему PAM для аутентификации. Если подсистема PAM правильно настроена для аутентификации пользователей по базе данных LDAP, то будут достигнуты те же функциональные возможности. Файл конфигурации сервера OpenVPN будет выглядеть следующим образом:

```
proto udp
port 1194
dev tun
server 10.200.0.0 255.255.255.0
topology subnet
persist-key
persist-tun
keepalive 10 60

remote-cert-tls client
tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn-movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

user nobody
group nobody # use 'group nogroup' on Debian/Ubuntu

verb 3
daemon
log-append /var/log/openvpn.log

plugin /usr/lib64/openvpn/plugin/lib/openvpn-auth-pam.so "login login
USERNAME password PASSWORD"
```

## Устранение неполадок в аутентификации LDAP

Устранение неполадок в подключаемом модуле аутентификации LDAP может быть непростым делом. Прежде всего, важно убедиться что VPN-сервер способен подключаться к LDAP-серверу и информация о пользователе может быть получена. Для этого очень удобен инструмент `ldapsearch`. Этот инструмент входит в пакет утилит клиента OpenLDAP.

Используя `BaseDN` и `SearchFilter` из файла `movpn_ldap.conf`, мы можем запросить сервер LDAP:

```
$ ldapsearch -x -H ldaps://ldap.example.org \
 -b ou=LocalUsers,dc=example,dc=org \
 "(&(uid=janjust)(authorizedService=login))"
```

Опция `-x` означает анонимную (неаутентифицированную) привязку к серверу, а опция `-H` указывает URI сервера. Обратите внимание, что URI отличается от имени хоста, так как он будет включать протокол (SSL или обычный текст), а также имя хоста. Вывод `ldapsearch` должен быть чем-то вроде этого:

```
extended LDIF
#
LDAPv3
base <ou=LocalUsers,dc=example,dc=org> with scope subtree
filter: (&(uid=janjust)(authorizedService=login))
requesting: ALL
```

```

#
janjust, LocalUsers, example.org
dn: uid=janjust,ou=LocalUsers,dc=example,dc=org
loginShell: /bin/bash
uid: janjust
cn: Jan Just Keijser
...
authorizedService: login

search result
search: 2
result: 0 Success

numResponses: 2
numEntries: 1

```

Убедитесь что это работает, прежде чем пытаться подключить клиента OpenVPN. Если это работает, но клиент не может подключиться к серверу OpenVPN, то увеличте детальность на сервере и отслеживайте сообщения LDAP. Добавьте следующую строку в конец файла конфигурации и перезапустите сервер:

#### **verb 5**

Переподключите клиента и просмотрите журнал сервера на наличие сообщений аутентификации LDAP. В случае неудачной попытки подключения журналы сервера будут содержать такие строки:

```

LDAP bind failed: Invalid credentials

Incorrect password supplied for LDAP DN
"uid=janjust,ou=LocalUsers,dc=example,dc=org".

[...] PLUGIN_CALL: POST
/usr/lib64/openvpn/plugin/lib/openvpn-auth-ldap.so/PLUGIN_AUTH_USER_PASS_VERIFY
status=1

[...] PLUGIN_CALL: plugin function PLUGIN_AUTH_USER_PASS_VERIFY failed with
status 1: /usr/lib64/openvpn/plugin/lib/openvpn-authldap.so

[...] TLS Auth Error: Auth Username/Password verification failed for peer

Whereas a successful connection attempt will show
[...] PLUGIN_CALL: POST
/usr/lib64/openvpn/plugin/lib/openvpn-authldap.so/PLUGIN_AUTH_USER_PASS_VERIFY
status=0

[...] TLS: Username/Password authentication succeeded for username 'janjust'

```

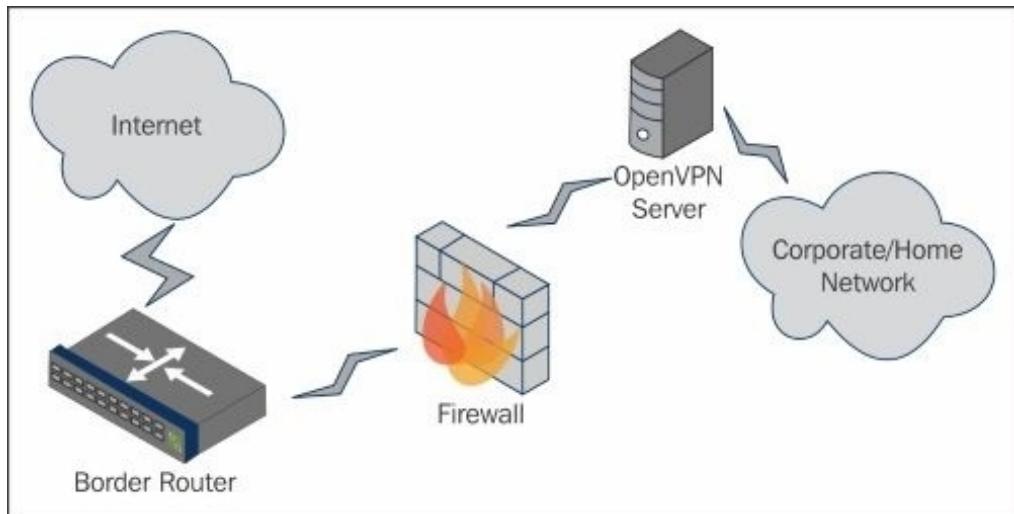
В этих сообщениях журнала детали, относящиеся к соединению, такие как IP-адрес клиента и номер порта UDP, были заменены на [...].

## **Фильтрация OpenVPN**

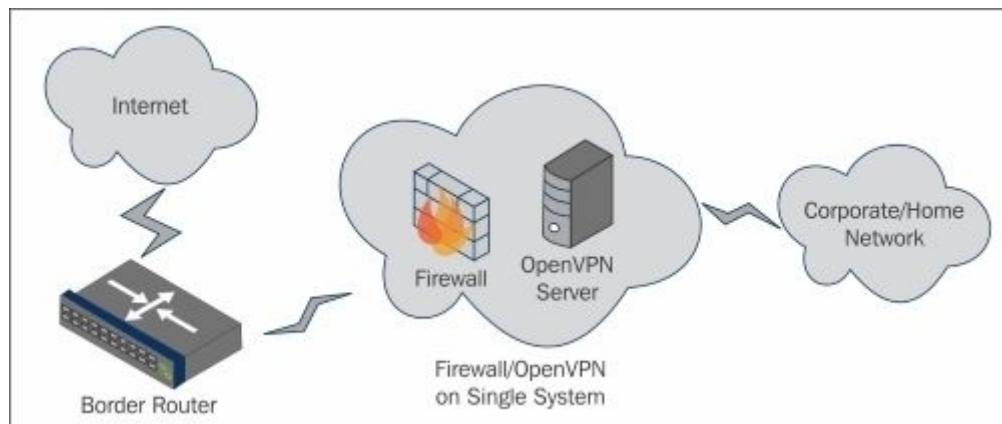
Как и любой другой интерфейс в системе или на сервере - интерфейсы адаптера tun и tap могут быть отфильтрованы с помощью соответствующего программного обеспечения брандмауэра операционной системы. Во многих случаях, как для целей маршрутизации, так и для фильтрации, лучше всего логически разместить сервер OpenVPN в центральном сетевом расположении, например, на пограничном маршрутизаторе или рядом с ним. Для дома это будет кабельный или DSL-модем. В корпоративных сетях - как правило, фактический основной маршрутизатор, такой как периферийное устройство Cisco или Juniper.

В зависимости от платформы и ваших собственных или бизнес-предпочтений, брандмауэр может быть отдельным устройством между сервером OpenVPN и незащищенным Интернетом, или же может быть программным обеспечением, работающим в той же системе, что и ваш сервер OpenVPN. Большие установки могут даже иметь несколько брандмауэров.

Первое изображение показывает сеть с отдельным межсетевым экраном, установленным между сервером OpenVPN и пограничным маршрутизатором и интернетом:



На следующем рисунке показано, как логически межсетевой экран и сервер OpenVPN могут находиться на одном компьютере:



Такие проекты, как pfSense (<https://www.pfsense.org>) и OpenWRT (<https://openwrt.org>) интегрировали интернет-соединения, локальные сети и VPN в единую систему. В этих системах используется программное обеспечение, предоставляющее простой графический интерфейс для управления беспроводными сетями, подключениями к Интернету, экземплярами VPN и набором правил брандмауэра для их защиты.

Для наших примеров мы собираемся разрешить только порты 80 и 443 от VPN-клиентов для остальной части сети.

## Пример FreeBSD

Bo FreeBSD мы будем использовать pf для фильтрации трафика внутри и вне нашей VPN. Bo FreeBSD интерфейс OpenVPN имеет значение tun0. Во-первых, pf должен быть включен в rc.conf:

```
pf_enable="YES"
```

Для начала создайте чрезвычайно простой набор правил в файле по умолчанию /etc/pf.conf:

```
pass all
```

Затем запустите pf:

```
root@server:~-> /etc/rc.d/pf start
Enabling pf
No ALTQ support in kernel
ALTQ related functions disabled
```

Используя pfctl, мы можем перечислить правила и их счетчики:

```
root@server:~-> pfctl -vvv -s rules
No ALTQ support in kernel
ALTQ related functions disabled
@0 pass all flags S/SA keep state
 [Evaluations: 209 Packets: 13 Bytes: 624
States: 2]
 [Inserted: uid 0 pid 71163 State Creations: 7]
```

На данный момент у нас есть простой набор правил, который просто пропускает все пакеты на всех интерфейсах. Поскольку это не книга по освоению pf - мы не будем вдаваться в подробности всей конфигурации. Вот пример фильтра, разрешающего весь трафик на всех интерфейсах, кроме tun0. Трафик на tun0 будет отфильтрован как входящий, поэтому с VPN-клиентов можно устанавливать подключения к локальной сети только через порты 80 и 443. Исходящий трафик на tun0 будет разрешен.

```
Mastering OpenVPN - FreeBSD Filtering Example
vpn_if="tun0"
out_if="xn0"
in_if="xn0"

lanv4="10.50.0.0/24"
lanv6="2001:db8:900::/64"
vpnv4="10.200.0.0/24"
vpnv6="2001:db8:100::/64"

pass on {$in_if, $out_if} all
pass out on $vpn_if all
block in on $vpn_if
```

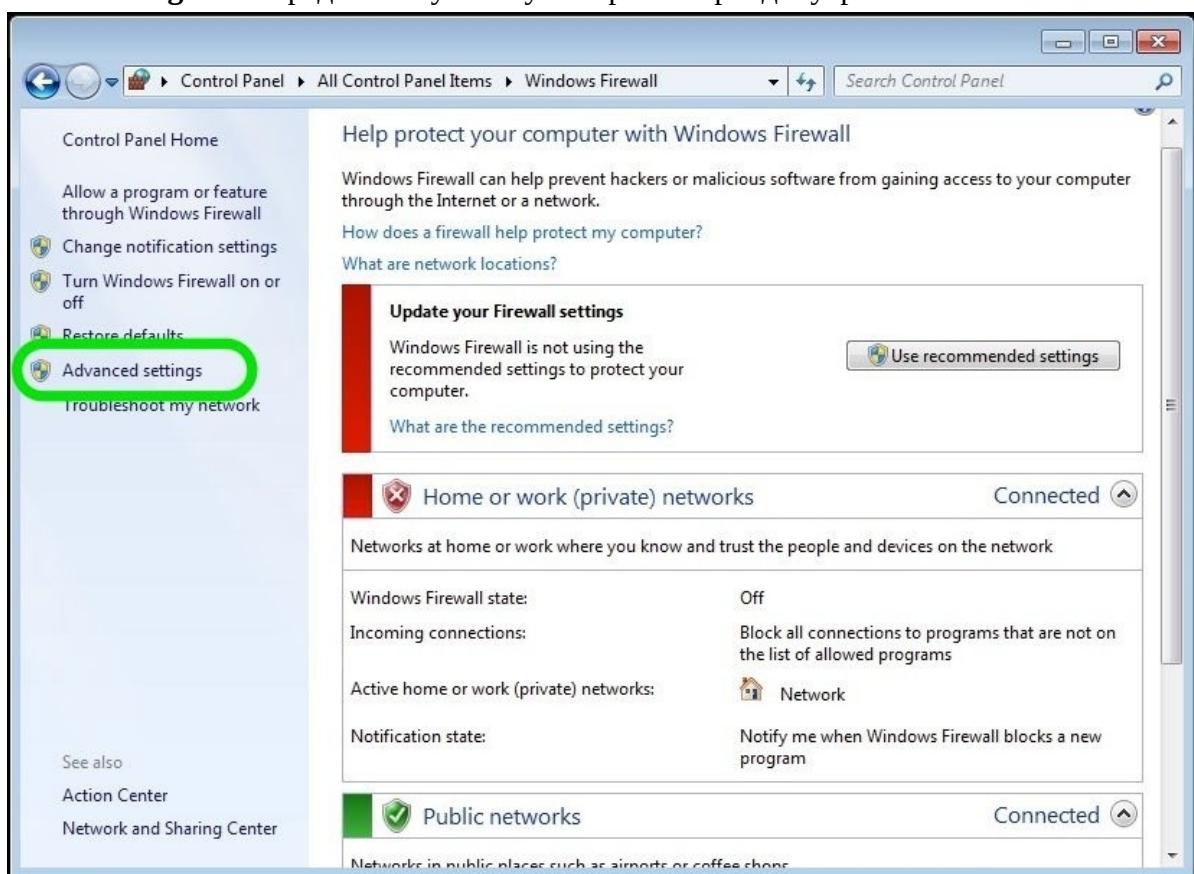
```
pass in on $vpn_if inet proto tcp to $lanv4 port {http, https}
pass in on $vpn_if inet6 proto tcp to $lanv6 port {http, https}
```

### Подсказка

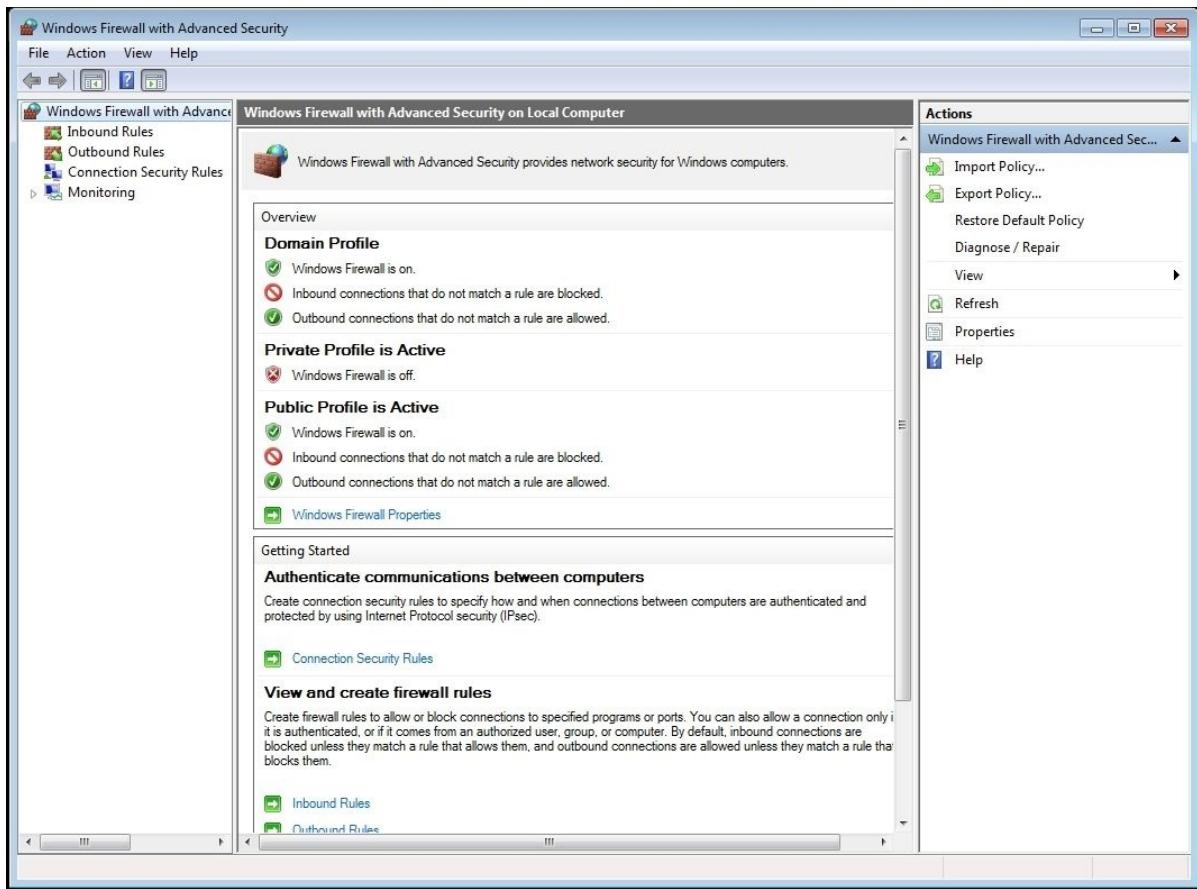
И OpenVPN и FreeBSD имеют фильтр пакетов pf. В приведенном выше примере мы используем pf, поддерживаемый ядром, а не встроенный в OpenVPN.

## Пример Windows

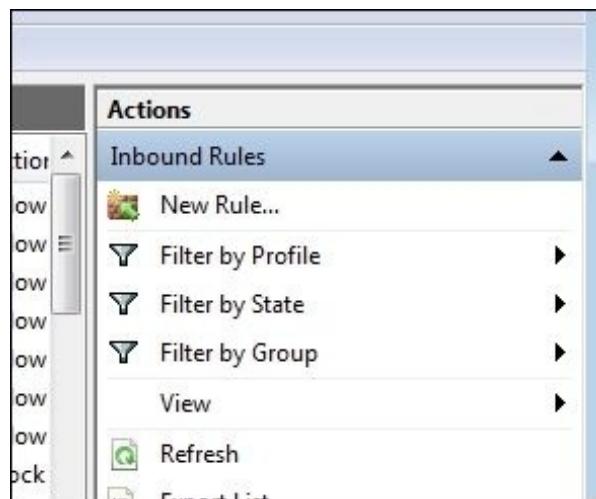
Предполагая, что у вас уже есть сервер OpenVPN работающий на системе Windows 7 Professional, вы будете получать доступ к настройке брандмауэра через **Панель управления**. После того, как **Панель управления** открыта, введите **firewall** в поле поиска и нажмите на **Windows Firewall**. Из параметров, доступных на левой боковой панели окна, нажмите **Advanced Settings**. Это представит утилиту настройки брандмауэра.



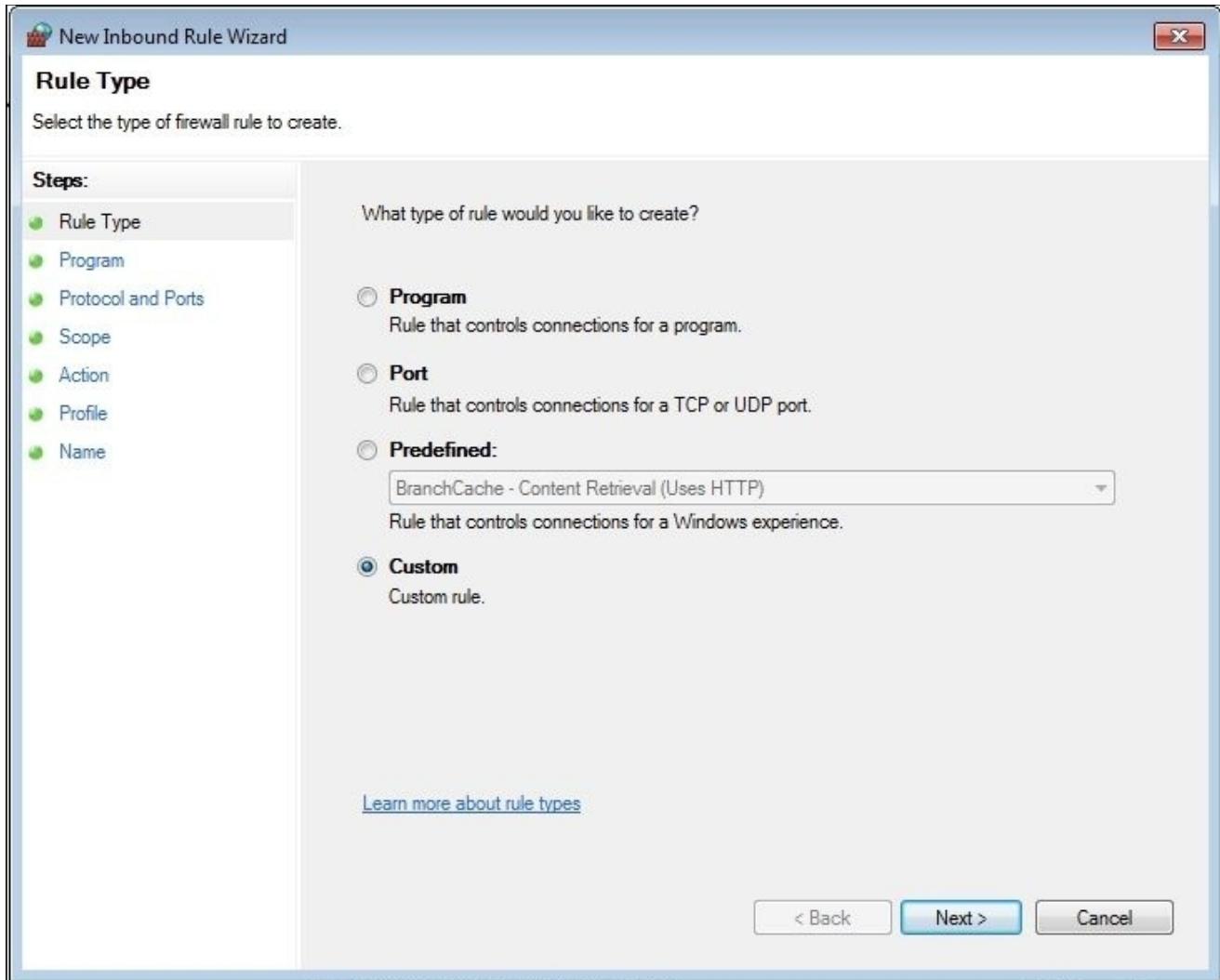
Нажмите **Advanced settings**, чтобы открыть программу **Windows Firewall with Advanced Security**, как показано на следующем рисунке:



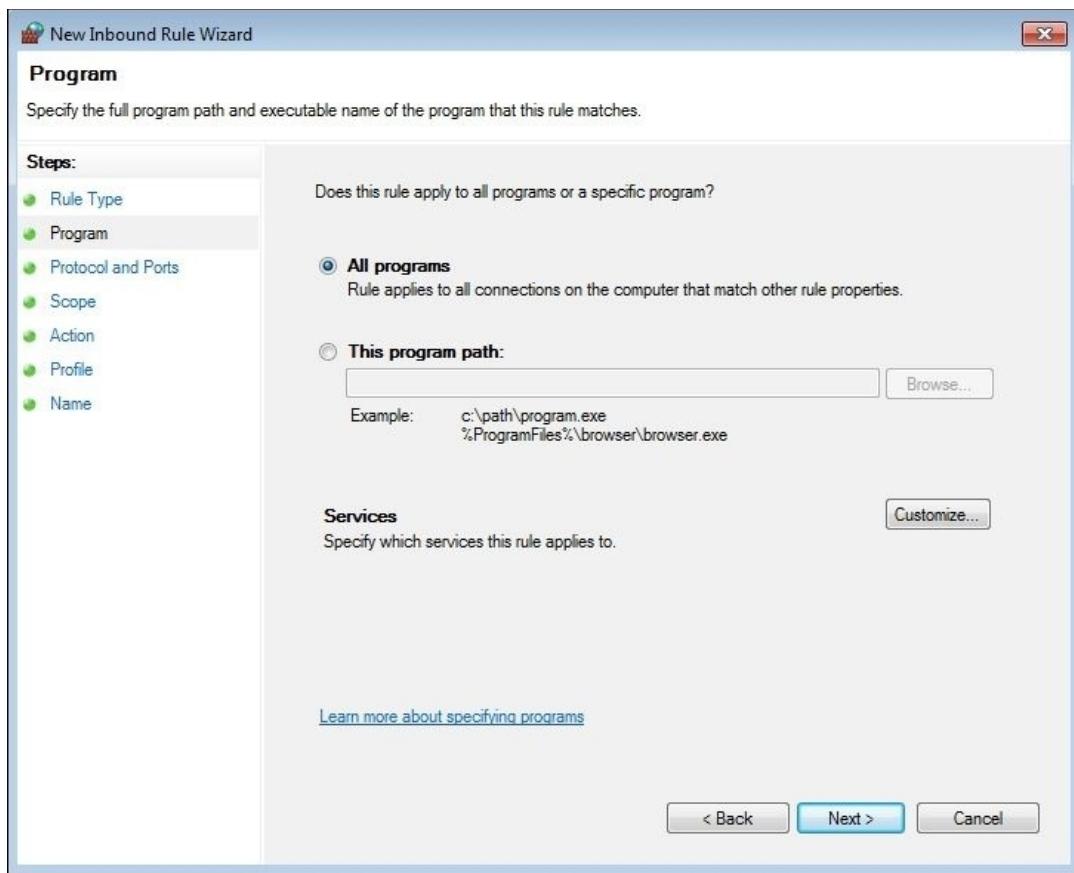
Попав в утилиту, нам нужно создать два входящих правила. Первое правило будет блокировать весь трафик от VPN, а второе - разрешит трафик через порты 80 и 443 от VPN.



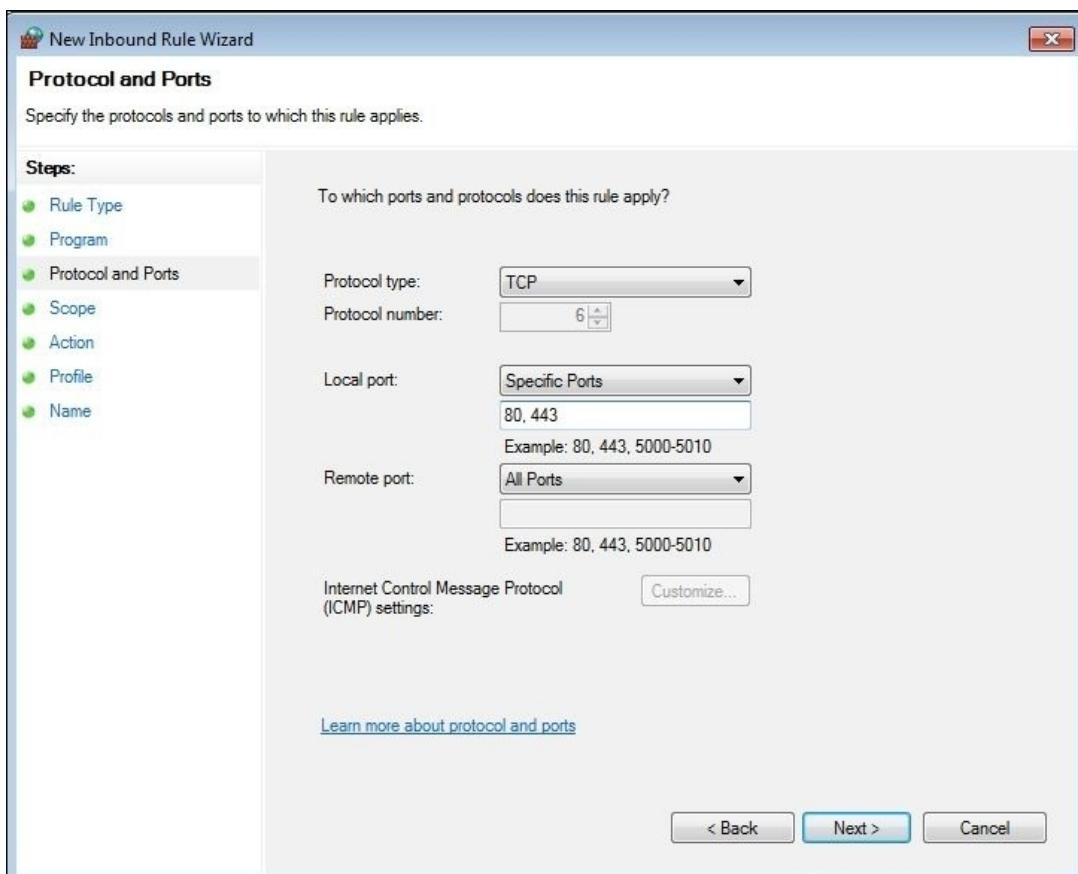
После открытия **New Inbound Rule Wizard** выберите параметр для создания настраиваемого правила, как показано на следующем снимке экрана. Это позволяет нам определить все конкретные диапазоны входящих адресов и порты, необходимые для эффективности.



На второй странице выберите переключатель **Все программы**, чтобы применить это правило ко всем программам:



На третьем экране входящего правила укажите TCP-порты 80 и 443 для входящего правила и разрешите их со всех удаленных портов. Это показано на следующем снимке экрана:



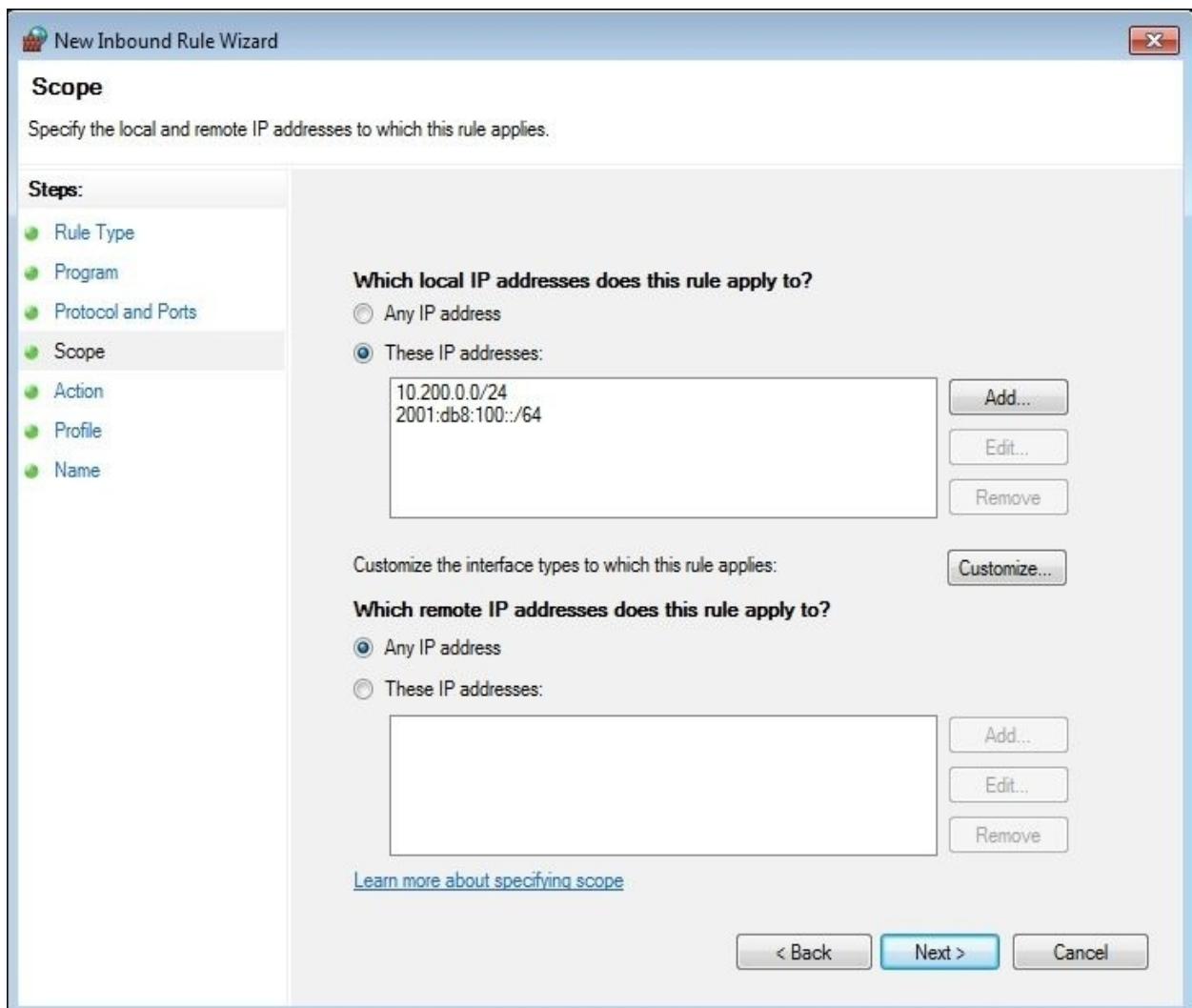
На странице диапазона укажите локальные диапазоны IP-адресов VPN:

- 10.200.0.0/24 (IPv4)
- 2001:db8:100::/64 (IPv6)

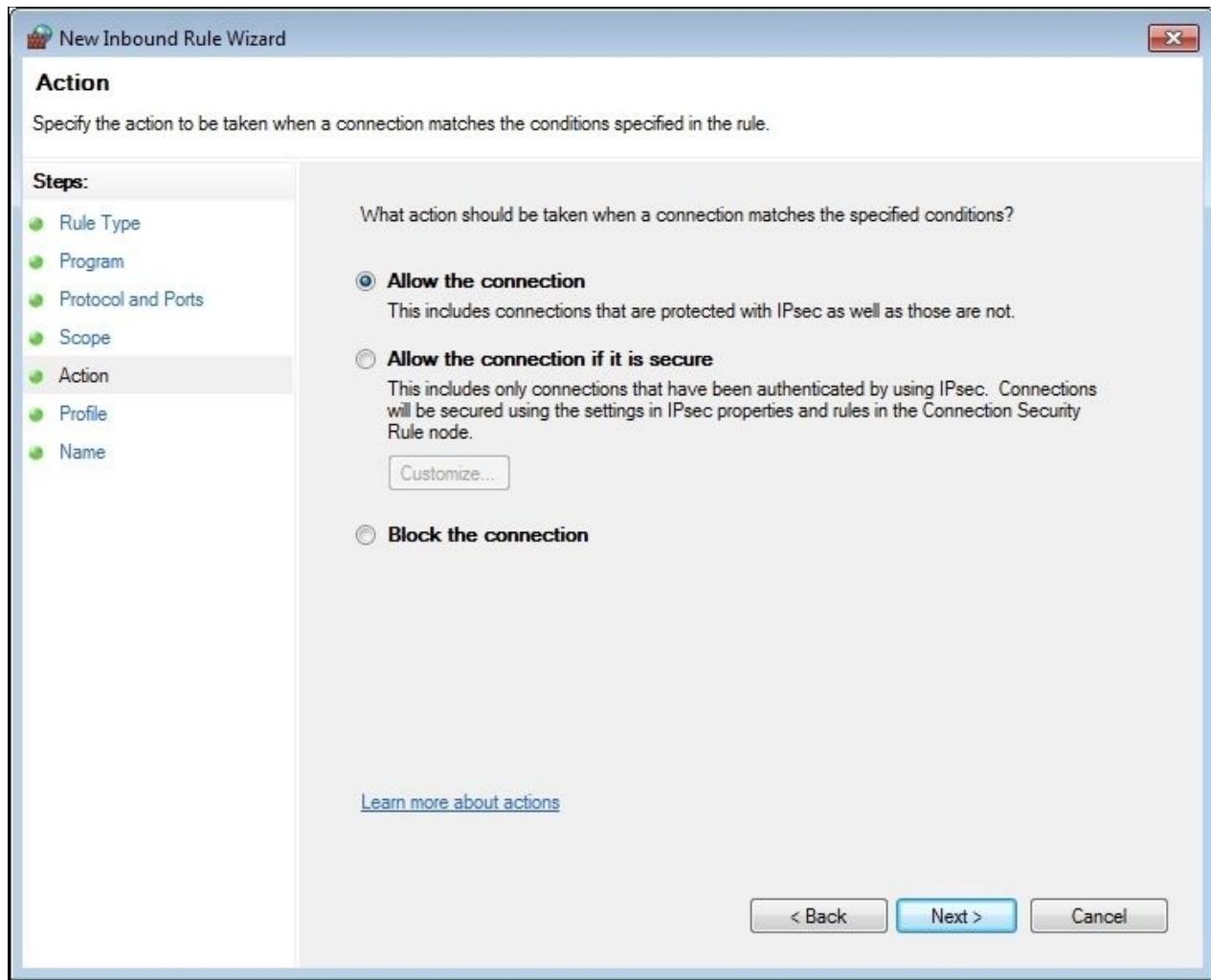
Затем разрешите все удаленные IP-адреса, выбрав **Any IP address** (это значение по умолчанию). Позже, при создании правила блокировки, вы оставите значения по умолчанию - оба для **Все порты** на этой странице.

### Подсказка

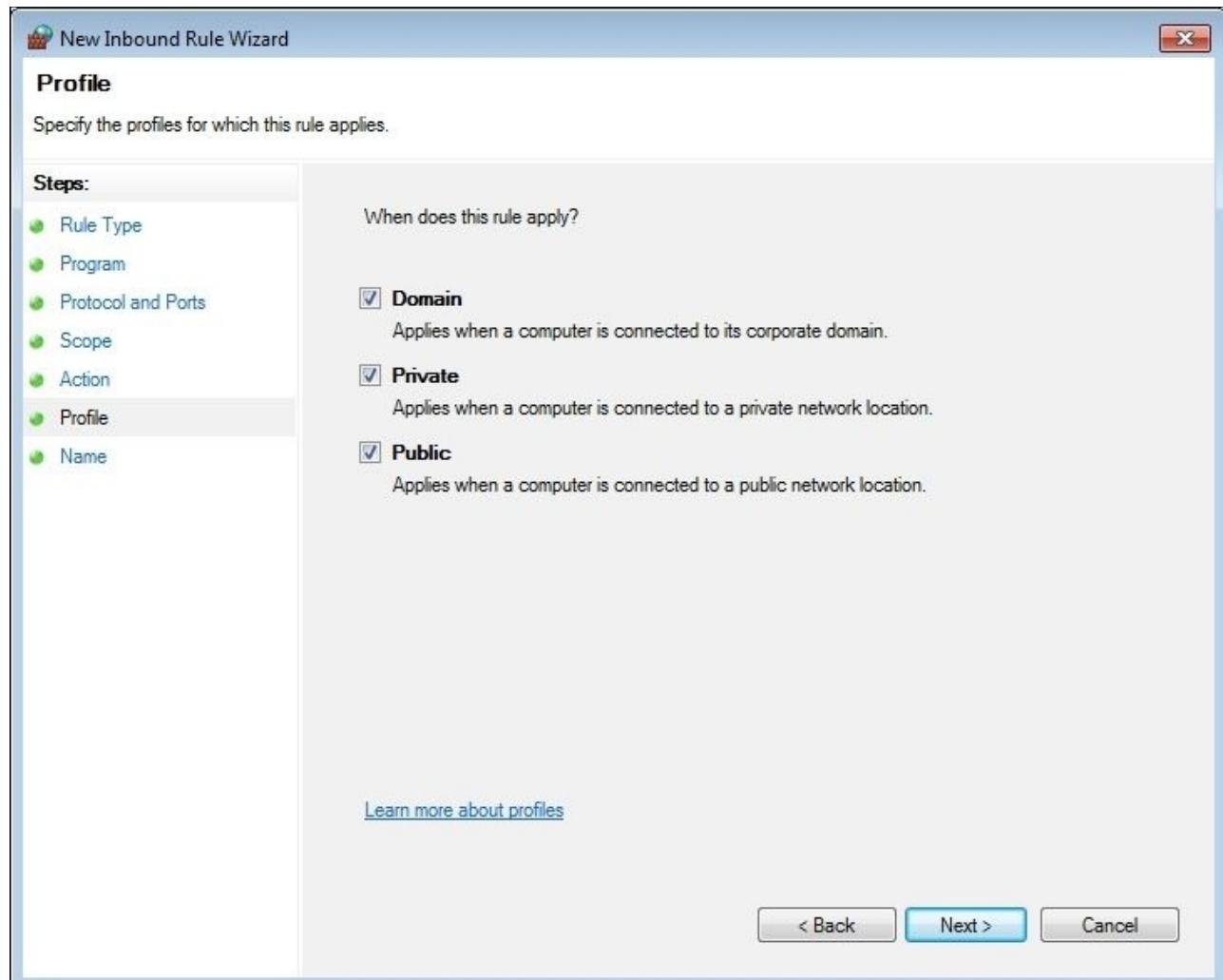
Вы также можете применить это к диапазонам IP-адресов VPN, чтобы разрешить только адресам хостов VPN связываться с ресурсами VPN. Отказ от этого ограничения разрешает доступ другим удаленным подсетям, которые вы, возможно, захотите направить (например, с помощью `--iroute`).



Далее выберите **Allow the connection** из списка действий. Позже, при создании правила блокировки, вам нужно будет выбрать **Block the connection** на этой странице.



Отметьте все три поля на странице мастера **Profile**. Это позволит правилу функционировать независимо от назначенного профиля интерфейса (**Public**, **Private** или **Domain**). Мы обсудим это позже.



Последняя страница мастера входящих правил позволяет вам задать имя. Введите какое-нибудь описание, которое позволит администратору быстро определить правило и то, как оно применяется. В нашем случае мы устанавливаем имя VPN – Allow 80, 443 для правила разрешения.

Теперь, когда правило разрешения создано – снова выполните эти шаги и создайте правило блокировки. Это правило блокирует весь трафик от VPN. В сочетании с правилом разрешения, которое мы уже создали, это позволит нашим VPN-клиентам подключаться только через VPN через порты 80 и 443.

Используйте следующие настройки для вашего правила блокировки на каждой странице мастера:

- **Rule Type** должен быть **Custom**
- Для параметра **Program** выберите **All programs**
- Выберите **All Ports** для локальных и удаленных портов
- **Scope** должна быть **Any IP address** (для локальных и удаленных)
- **Action** должно быть **Block the connection**
- Выберите **Select all profiles** в **Profile**
- Установить **Name** как VPN – Block All

После сохранения у вас будет пара правил VPN в верхней части списка.

Inbound Rules				
Name	Group	Profile	Enabled	Action
VPN - Allow 80, 443	All	Yes	Allow	Allow
VPN - Block All	All	Yes	Block	Block

Брандмауэр Windows в режиме повышенной безопасности - это первый тип брандмауэра. Он означает, что как только правило брандмауэра соответствует данному сетевому пакету, обработка набора правил прекращается и применяется указанное действие в этом правиле. Фильтр пакетов `iptables` в Linux - это еще один фильтр первого соответствия. Другие фильтры пакетов, такие как `pf` в OpenBSD, соответствуют последним, что означает, что правила продолжают обрабатываться до конца.

### Подсказка

OpenVPN имеет встроенную возможность фильтрации, но его код не затрагивался в течение ряда лет и требует дополнительных плагинов для работы. Разработчики не думают что код является жизнеспособным в настоящее время, но эта функция может быть возвращена в будущем.

Для выполнения многих задач типа «сервер» в настольной версии Windows обычно требуется, чтобы администратор пошел по проторенному пути. Например, Windows Server 2008 имеет лучшие инструменты редактирования брандмауэра, чем Windows 7 Professional, которая используется в приведенном выше примере.

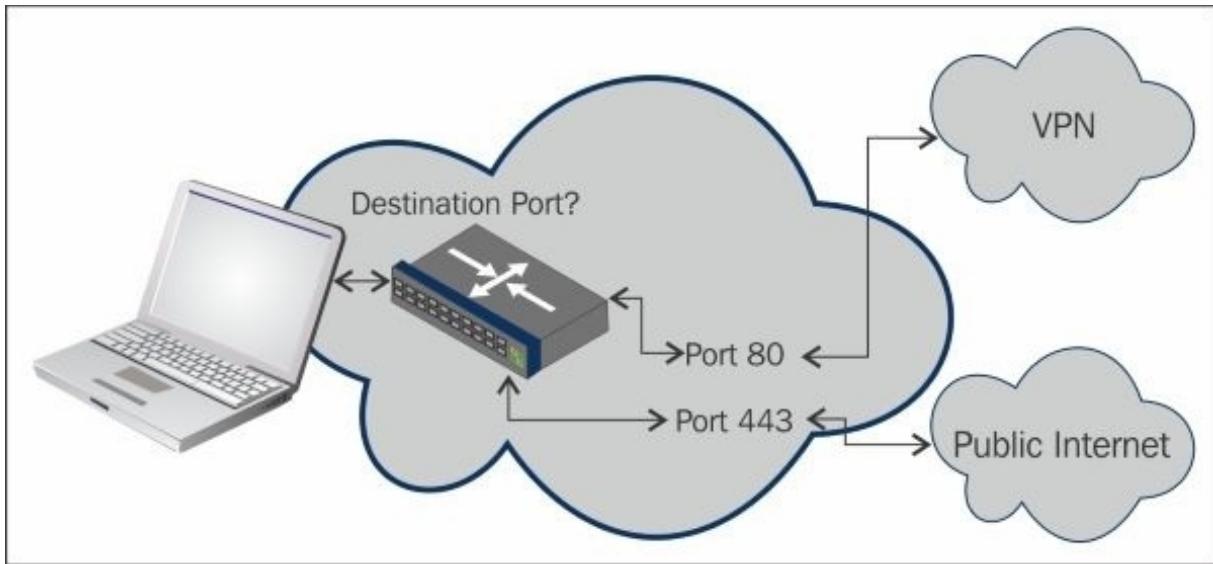
## Маршрутизация на основе политик

Маршрутизация на основе политик использует брандмауэр или другой фильтр пакетов для маршрутизации трафика на основе не только IP-адресов источника и назначения, но также портов источника и назначения. Одним из распространенных способов использования маршрутизации на основе политик является отправка всего незащищенного трафика порта 80 через VPN, но при этом другой трафик, такой как SSL-трафик через порт 443 может проходить через общий Интернет.

Маршрутизация на основе политик - это то, что нужно сделать в источнике. В большинстве случаев это означает, что он будет применяться на клиенте OpenVPN. В некоторых случаях это может быть расширено на сервере OpenVPN, но гибкость значительно снижена.

Каждый брандмауэр или программное обеспечение для фильтрации пакетов обрабатывает политику маршрутизации по-своему. Нам не удалось настроить брандмауэр Windows 7 для маршрутизации на основе портов источника или назначения и даже фильтр пакетов OpenBSD `pf` имеет определенные предостережения.

На следующем рисунке показано простое решение о маршрутизации политики порта 80 или 443, соответствующее нашему примеру сценария из предыдущего раздела:



## Сетевые расположения Windows - общественные или частные

Постоянный вопрос в списках рассылки OpenVPN состоит в том, как изменить сетевое расположение виртуального сетевого адаптера TAP-Win OpenVPN с **общественного** на **частное**. Этот вопрос возник с появлением Windows Vista. Ответ на него, к сожалению, довольно длинный. В этом разделе мы рассмотрим различные методы, которые позволяют нам изменять сетевое расположение адаптера TAP-Win на клиентах Windows.

### Бэкграунд

Начиная с Windows Vista, Microsoft представила концепцию сетевых расположений. В Windows 7 есть три сетевых расположения: **Домашняя сеть**, **Сеть предприятия** и **Общественная сеть**. Эти сетевые расположения применяются ко всем сетевым адаптерам: проводным, беспроводным, а также виртуальным сетевым адаптерам TAP-Win от OpenVPN.

**Домашнее** сетевое расположение предназначено для домашней сети и обеспечивает высокий уровень доверия. Оно также включает функцию **Домашняя группа**, благодаря которой компьютер может легко подключаться ко всем другим устройствам дома. Точно так же расположение **Рабочей** сети обеспечивает высокий уровень доверия на работе, позволяя компьютеру обмениваться файлами, подключаться к принтерам и т.д. В Windows 8 **домашняя** и **рабочая** сетевые папки объединяются в **частную** сеть.

Расположение в **общедоступной** сети не является доверенным и доступ к сетевым ресурсам, как входящим так и исходящим, строго ограничен Windows, даже если брандмаэр Windows отключен.

Когда брандмаэр Windows включен - профиль **частного** брандмаэра применяется ко всем сетевым адаптерам в **домашней** и **рабочей** сети, а профиль **общего** брандмаэра применяется ко всем сетевым адаптерам в **общедоступной** сети.

Чтобы доверять сетевому адаптеру - он должен объявить шлюз по умолчанию или сетевой адаптер быть частью домена Windows. Есть некоторая документация об этом онлайн: <http://blogs.technet.com/b/networking/archive/2009/02/20/why-is-my-networkdetected-as-unknown-by-windows-vista-or-windows-server-2008.aspx>

Когда Windows не может определить *местоположение* сети - автоматически выбирается расположение как **общедоступной** сети. К сожалению, невозможно изменить состояние, если сетевой адаптер классифицирован как **общедоступный**.

## Изменение местоположения адаптера TAP-Win с помощью redirect-gateway

OpenVPN может установить шлюз по умолчанию на удаленном адаптере TAP-Win с помощью директивы конфигурации:

```
redirect-gateway
```

Обычно рекомендуется добавить параметр def1 к этой опции. Опция def1 заставляет OpenVPN не добавлять новый шлюз по умолчанию (в терминах сети - маршрут 0.0.0.0/0.0.0.0), а вместо этого добавляет два маршрута с маской сети 128.0.0.0 как объяснено в предыдущей главе. Недостатком опции def1 является то, что Windows не распознает адаптер TAP-Win как имеющий шлюз по умолчанию. Для получения дополнительной информации о различных альтернативах опции redirect-gateway см. [Главу 4, Режим клиент/сервер с устройствами tun](#).

Чтобы протестировать эту опцию, мы добавляем строку в файл basic-udp-server.conf:

```
proto udp
port 1194
dev tun
server 10.200.0.0 255.255.255.0
topology subnet
persist-key
persist-tun
keepalive 10 60

remote-cert-tls client
tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

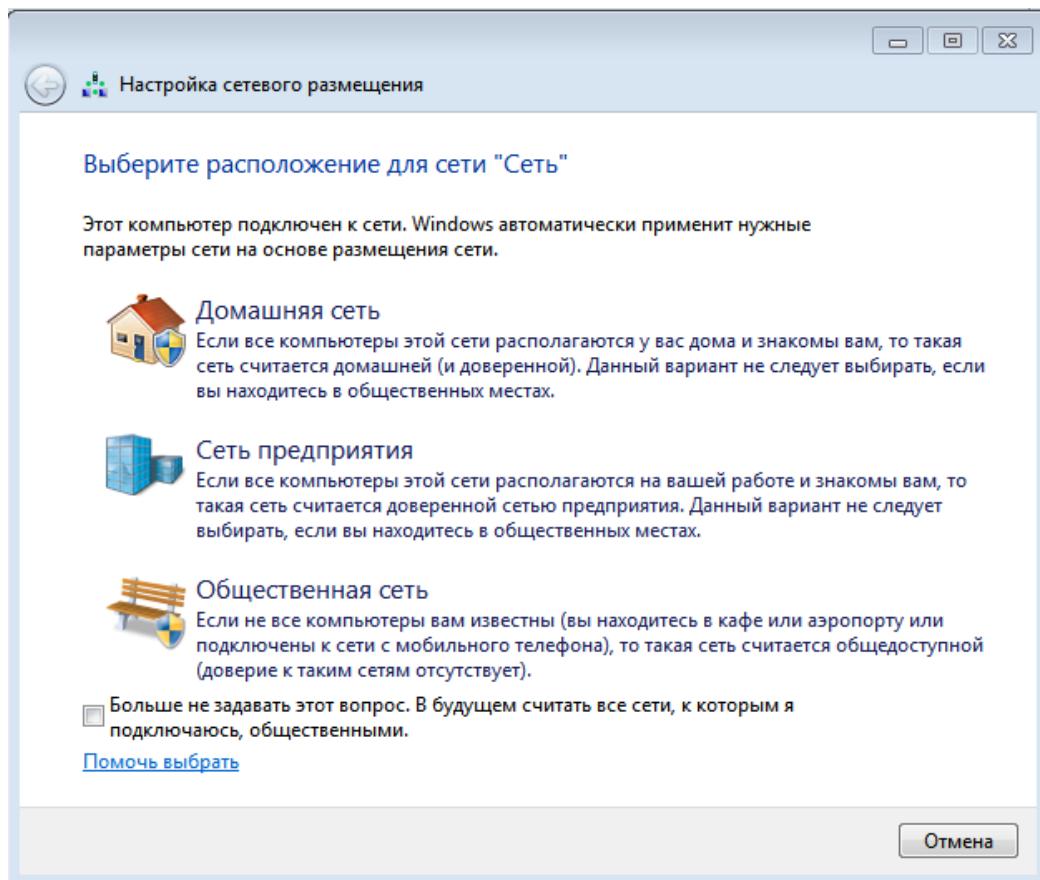
user nobody
group nobody # use 'group nogroup' on Debian/Ubuntu

verb 3
daemon
log-append /var/log/openvpn.log
```

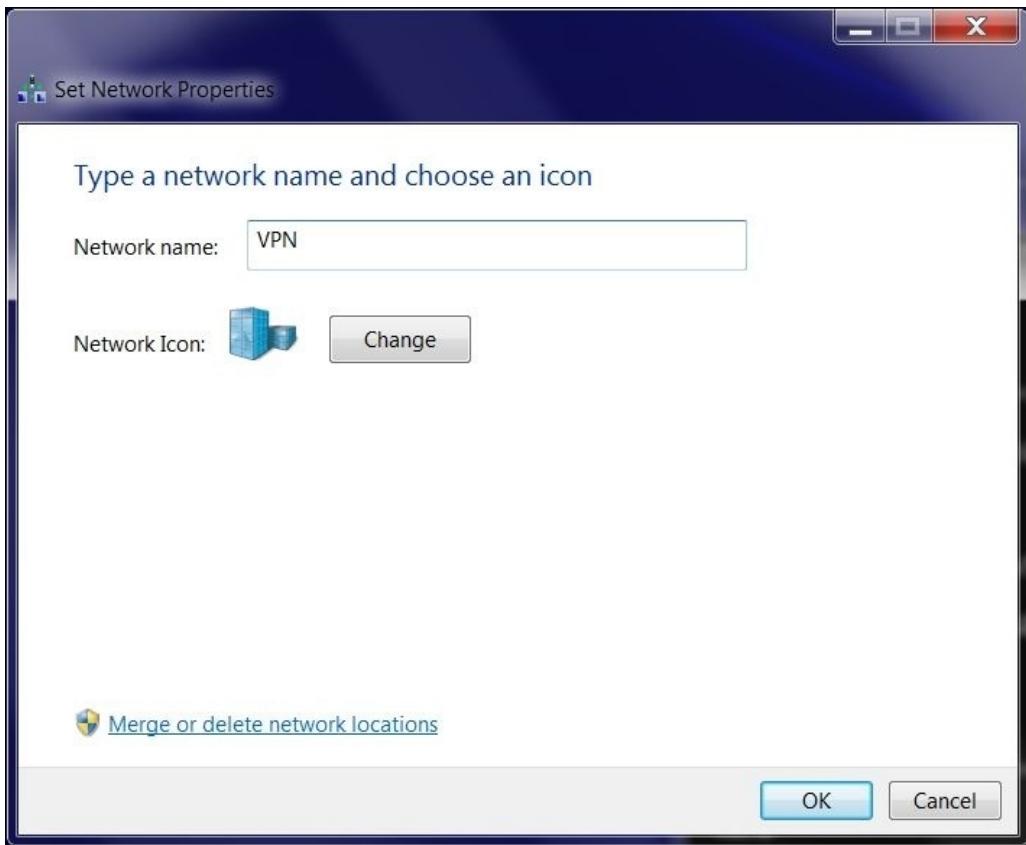
```
push "redirect-gateway"
```

Теперь сохраните его как `tnovpn-05-03-server.conf`. Запустите сервер OpenVPN, используя этот файл конфигурации и подключите клиента Windows 7, используя конфигурацию по умолчанию `basic-udp-client.ovpn`.

На этот раз, после успешного подключения клиента, Windows спросит, в каком месте разместить новую сеть:



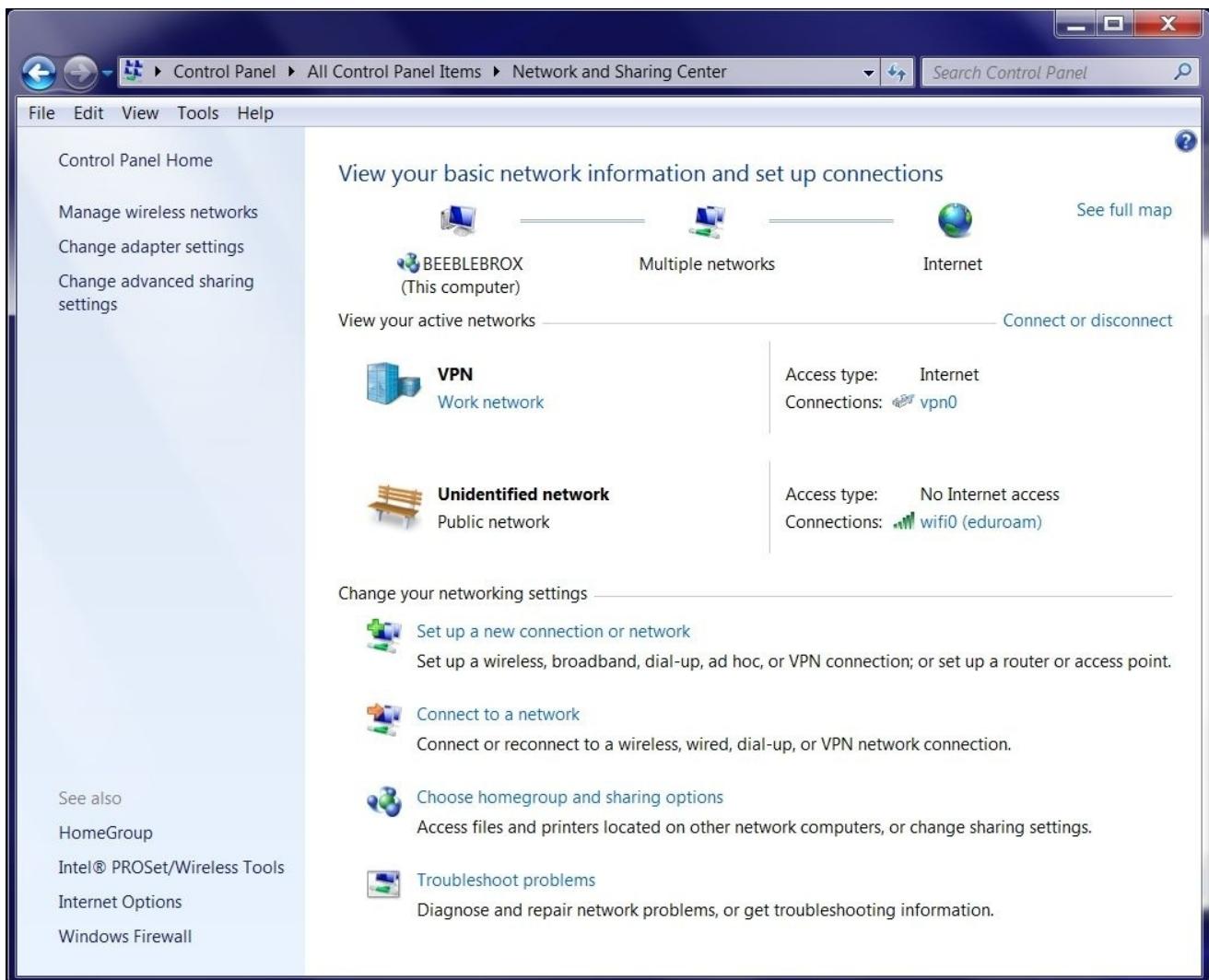
Выберите **Сеть предприятия**, после чего Windows позволит вам выбрать собственное имя и значок для сети VPN:



Мы выбрали имя VPN, выберите другой значок и нажмите **OK**.

Адаптер TAP-Win теперь является доверенным и в этой сети разрешен общий доступ к файлам Windows, а также другим доверенным протоколам.

Одним из недостатков использования *redirect-gateway* является то, что адаптер VPN теперь является единственным адаптером со шлюзом по умолчанию и, следовательно, является доверенным. Адаптер беспроводной сети, подключенный к сети Wi-Fi eduroam, теперь не имеет маршрута по умолчанию и внезапно становится частью *неопознанной сети* и больше не является доверенным. Это можно увидеть на следующем скриншоте **Центра управления сетями и общим доступом** Windows 7:



Самый большой недостаток использования *redirect-gateway* без *def1* становится очевидным когда VPN-соединение останавливается или прерывается. Поскольку шлюз по умолчанию на клиенте Windows OpenVPN был заменен - будет невозможно восстановить шлюз по умолчанию, существовавший до запуска VPN и потери всех подключений к Интернету. В большинстве случаев подключение (беспроводное) к локальной сети необходимо перезапустить для восстановления шлюза.

### **Использование редактора групповой политики для принудительного использования адаптера в качестве частного**

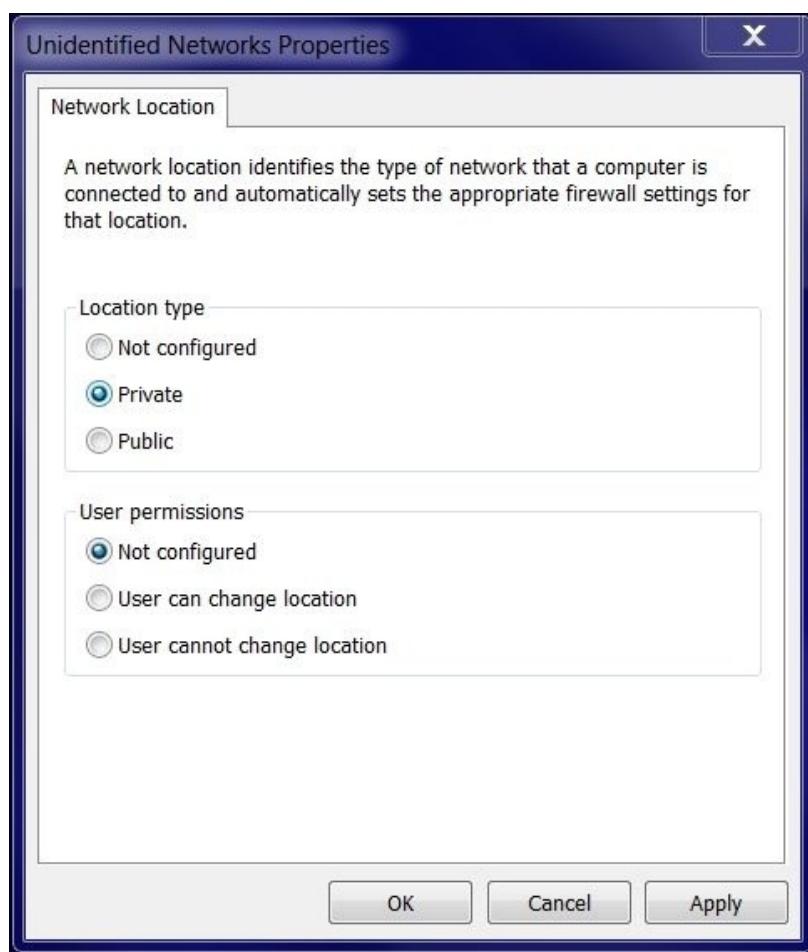
Второй подход к изменению местоположения адаптера TAP-Win заключается в использовании редактора групповой политики Windows. Используя этот инструмент можно заставить адаптер TAP-Win (или любой неопознанный адаптер) быть либо **частным**, либо **общественным**:

1. Откройте **командную строку** и запустите **редактор групповой политики**:

**gpedit.msc**

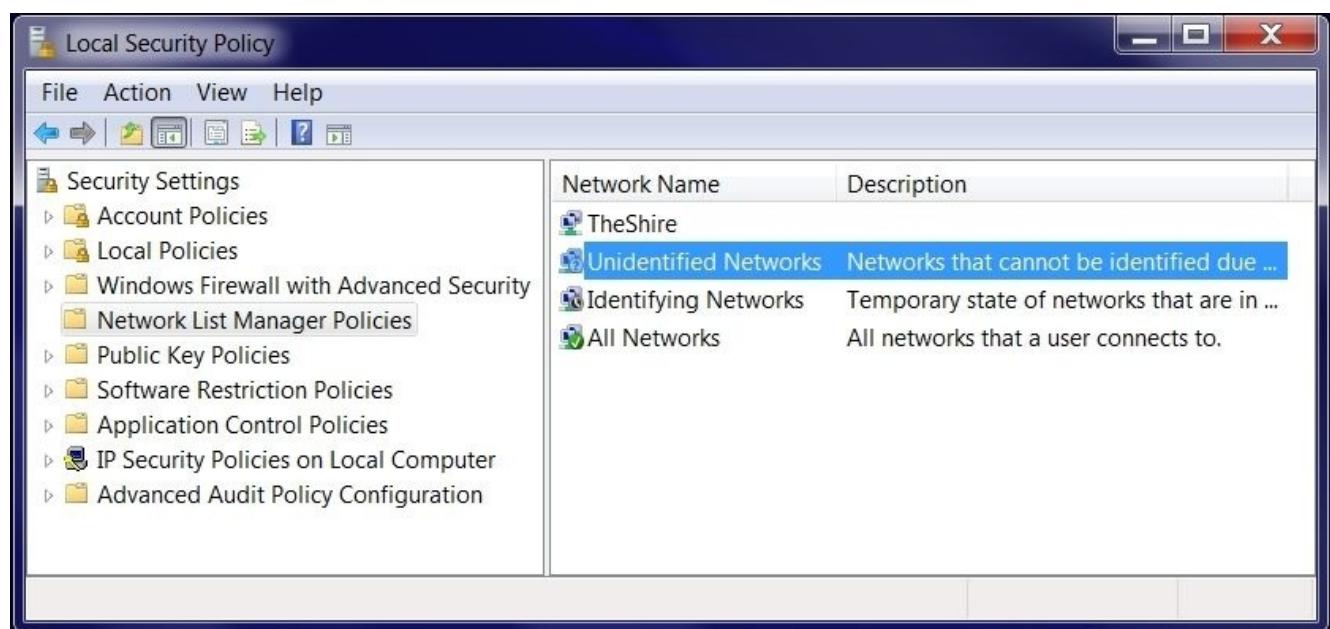
- Выберите **Политики диспетчера списка сетей** и дважды щелкните **Неопознанные сети** на правой панели, как показано на следующем снимке экрана:

- Откроется новое окно **Свойства неопознанных сетей**. Установите **Тип расположения** на **Частный** и нажмите **OK**, как показано на следующем скриншоте:



Перезапустите клиент OpenVPN без флага *redirect-gateway*.

Теперь Windows автоматически отметит адаптер как **Частный** и снова спросит в какую сеть



следует установить адаптер: **Домашнюю** или **Сеть предприятия**. Однако на этот раз значок и имя выбранного типа сети изменить нельзя.

## **Подсказка**

В предыдущих инструкциях все неопознанные сети были настроены как частные и, возможно, могут иметь негативные побочные эффекты безопасности. Убедитесь, что вы понимаете все последствия этого изменения прежде чем устанавливать его.

## **Изменение местоположения адаптера TAP-Win с использованием дополнительных шлюзов**

Более элегантный подход к изменению местоположения адаптера заключается в добавлении дополнительного адреса шлюза в конфигурацию сервера:

```
push "route 0.0.0.0 0.0.0.0"
```

Когда клиент OpenVPN подключается - он добавляет дополнительный маршрут по умолчанию в системные таблицы маршрутизации. Этот маршрут всегда будет иметь более высокую метрику, чем обычный шлюз по умолчанию, но адаптер теперь является доверенным.

В Windows 7 метрика адаптера обычно рассчитывается автоматически как сумма метрики шлюза и метрики интерфейса. Метрики интерфейса основаны на типе и скорости адаптера. Поскольку TAP-Win адаптер OpenVPN зарегистрирован как адаптер 10 Мбит/с, он всегда будет иметь более высокую метрику (то есть будет менее предпочтительным), чем проводные или беспроводные адаптеры, имеющие более высокие скорости.

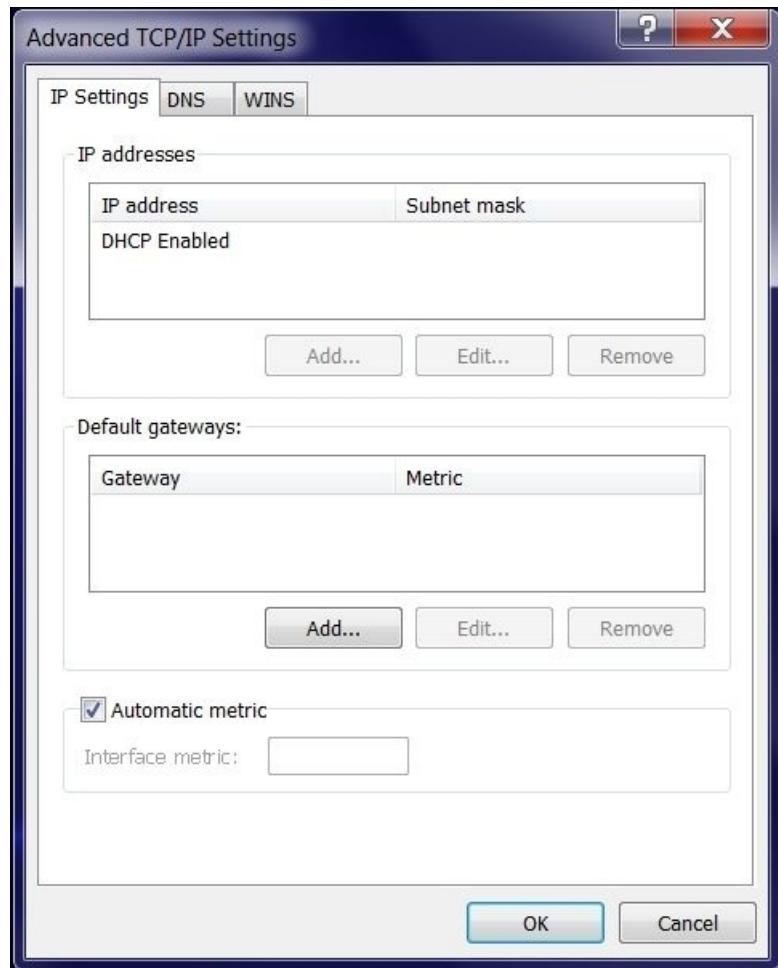
Эти метрики могут быть отображены с помощью команды netsh int ip show config:

```
C:\>netsh int ip show config

Configuration for interface "vpn0"
 DHCP enabled: Yes
 IP Address: 10.200.0.2
 Subnet Prefix: 10.200.0.0/24 (...)
 Default Gateway: 10.200.0.1
 Gateway Metric: 70
 InterfaceMetric: 30
 DNS servers configured through DHCP: 192.0.2.12
 Register with which suffix: Primary only
 WINS servers configured through DHCP: 192.0.2.60

Configuration for interface "wifi0"
 DHCP enabled: Yes
 IP Address: 192.0.2.233
 Subnet Prefix: 192.0.2.0/24 (...)
 Default Gateway: 192.0.2.254
 Gateway Metric: 0
 InterfaceMetric: 20
 DNS servers configured through DHCP: 192.0.2.17
 192.0.2.12
 Register with which suffix: None
 WINS servers configured through DHCP: 192.0.2.121
 192.0.2.20
```

Можно отключить автоматический расчет метрики и вернуться к поведению более старых версий Windows. Для этого перейдите в диалоговое окно **Дополнительные параметры TCP/IP** пункта **Свойства TCP/IPv4** сетевого адаптера:



В этом случае метрика, указанная в окне **Свойства TCP/IPv4**, будет определять маршрут по умолчанию в системе. Если показатель адаптера TAP-Win выше, чем у адаптера без VPN, то фактически весь трафик направляется через VPN!

Преимущество этого подхода заключается в том, что *старый* шлюз по умолчанию остается нетронутым, что позволяет избежать проблемы утерянного шлюза по умолчанию при падении или остановке VPN-подключения.

### **Перенаправление всего трафика в сочетании с дополнительными шлюзами**

В качестве последнего примера того, как можно повлиять на местоположение в сети с помощью параметров конфигурации OpenVPN, мы добавим дополнительный шлюз и перенаправим шлюз по умолчанию с помощью def1:

```

proto udp
port 1194
dev tun
server 10.200.0.0 255.255.255.0
topology subnet
persist-key
persist-tun
keepalive 10 60

remote-cert-tls client
tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

```

```

user nobody
group nobody # используйте 'group nogroup' в Debian/Ubuntu

verb 3
daemon
log-append /var/log/openvpn.log

push "route 0.0.0.0 0.0.0.0"
push "redirect-gateway def1"

```

Сохраните файл как `ovpn-05-04-server.conf`. Запустите сервер OpenVPN, используя этот файл конфигурации и подключите клиента Windows 7, используя конфигурацию по умолчанию `basic-udp-client.ovpn`.

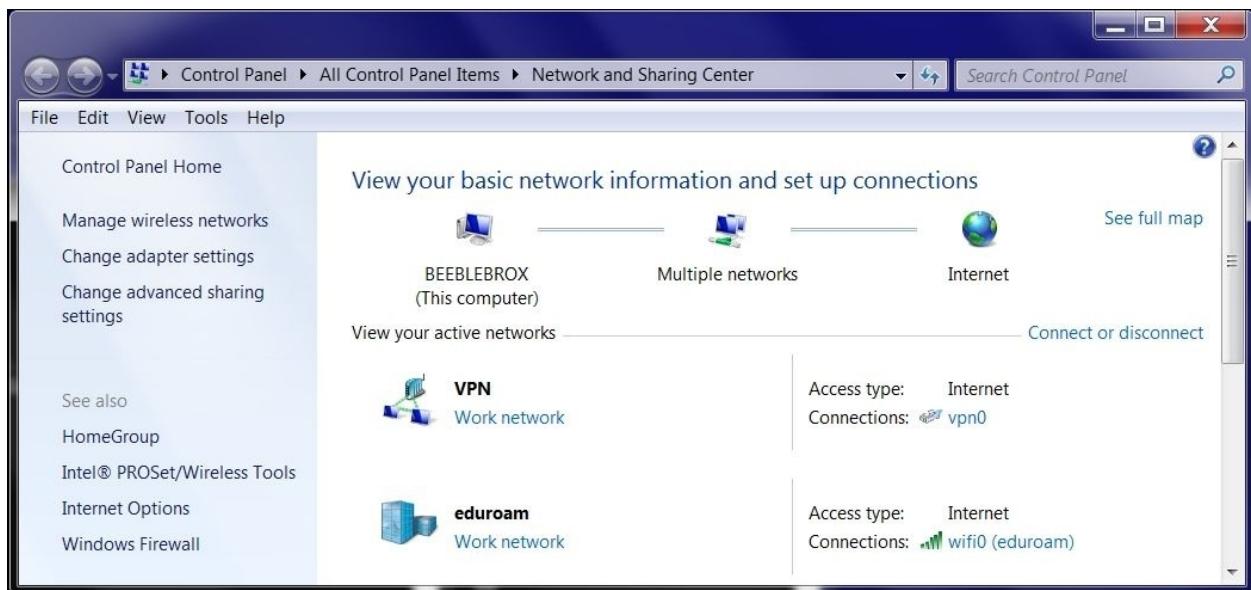
После переподключения клиента OpenVPN таблица маршрутизации IPv4 теперь содержит следующие записи:

Network Destination	Netmask	Gateway	Interface	Metric
0.0.0.0	0.0.0.0	192.0.2.254	192.0.2.233	20
0.0.0.0	0.0.0.0	10.200.0.1	10.200.0.2	100
0.0.0.0	128.0.0.0	10.200.0.1	10.200.0.2	30
128.0.0.0	128.0.0.0	10.200.0.1	10.200.0.2	30
10.200.0.0	255.255.255.0	On-link	10.200.0.2	286
10.200.0.2	255.255.255.255	On-link	10.200.0.2	286
10.200.0.255	255.255.255.255	On-link	10.200.0.2	286

Первый маршрут - это первоначальный шлюз. Второй маршрут - это дополнительный маршрут `0.0.0.0/0` для адаптера TAP-Win. Он заставляет Windows доверять адаптеру. Третий и четвертый маршруты устанавливаются командой `redirect-gateway def1` путем добавления маршрутов `0.0.0.0/1` и `128.0.0.0`.

OpenVPN предоставляет два новых маршрута, которые являются более конкретными, чем маршрут по умолчанию `0.0.0.0/0`. Маршрутизация TCP/IP указывает что всегда следует выбирать более конкретный маршрут, независимо от метрики интерфейса. Поэтому весь трафик перенаправляется через VPN-туннель.

В окне **Центр управления сетями и общим доступом** теперь сети как LAN, так и VPN отображаются как доверенные:



Преимущества этого подхода заключаются в следующем:

- Как исходная сеть (Wi-Fi eduroam в предыдущем примере) так и сеть VPN являются доверенными. Это означает, что общий доступ к файлам и принтерам доступен в обеих сетях.
- Весь сетевой трафик правильно перенаправляется через VPN, независимо от метрики интерфейса.
- Когда соединение VPN обрывается или останавливается, исходный шлюз по умолчанию корректно восстанавливается.

Быстрая проверка ссылки <https://www.whatismyip.com> покажет, что весь трафик теперь маршрутизируется через VPN. Еще один способ убедиться в этом - использовать `tracert` в командной оболочке Windows:

```
c:\>tracert -4 -d www
```

Трассировка маршрута к `www.nikhef.nl [192.16.199.160]`  
с максимальным числом прыжков 30:

```
1 95 ms 119 ms 83 ms 10.200.0.1
3 103 ms 119 ms 83 ms 192.0.2.133
4 115 ms 101 ms 101 ms 192.16.199.160
```

Трассировка завершена.

### Подсказка

В Mac OS X клиент Tunnelblick для OpenVPN имеет возможность проверить, изменился ли внешний IP после подключения к VPN, а затем уведомить пользователя, если он не изменился.

Первый переход в выводе `traceroute` - это адрес VPN-сервера, который доказывает, что шлюзом по умолчанию в системе действительно является VPN-адаптер.

## Использование OpenVPN с HTTP или SOCKS прокси

OpenVPN поддерживает работу через прокси HTTP или SOCKS без аутентификации, с базовой аутентификацией и с аутентификацией NTLM. Мы рассмотрим как прокси-серверы HTTP, так и SOCKS, как с аутентификацией, так и без нее.

## HTTP прокси

HTTP прокси требуют использования TCP для туннельного транспорта OpenVPN. Если вы в настоящее время используете UDP - необходимо обновить аргумент протокола как на сервере, так и в конфигурации клиента:

```
proto tcp
```

После настройки добавьте поддержку прокси для клиента - добавив директиву конфигурации --http-proxy. В качестве примера давайте предположим, что вашей локальной сети требуется анонимный прокси-сервер для исходящих соединений и этот сервер находится на 192.168.4.4 на порту по умолчанию 1080. Ваша конфигурация будет выглядеть примерно так:

```
http-proxy 192.168.4.4 1080 none
```

Это позволит вашему клиентскому соединению OpenVPN подключаться к удаленному серверу OpenVPN через прокси-сервер в вашей локальной сети. HTTP-прокси с аутентификацией не сильно отличается - просто замените none в предыдущей команде на информацию аутентификации:

```
http-proxy 192.168.4.4 1080 stdin basic
```

Предыдущая команда подключается к тому же прокси-серверу и порту что и раньше, но запрашивает стандартный ввод для имени пользователя и пароля, которые используются для базовой аутентификации HTTP. Кроме того, поддерживаемые методы аутентификации включают в себя файл аутентификации, который похож на основной файл паролей OpenVPN - просто имя пользователя и пароль в виде текста на двух отдельных строках:

```
некий_пользователь
некий_пароль
```

Путь к файлу аутентификации и имя файла передаются вместо ключевого слова `stdin` в предыдущем примере. Настройка `auto` позволяет OpenVPN определять, откуда запрашивать учетные данные, в том числе через консоль управления.

Некоторые прокси-серверы HTTP могут ограничивать доступ или аутентификацию на основе переданного агента пользователя HTTP или других параметров HTTP. Они могут быть определены с помощью аргумента конфигурации `http-proxy-option` для почти любой произвольной опции HTTP. Типичными примерами являются строка агента пользователя и строка версии HTTP:

```
http-proxy-option VERSION 1.1
http-proxy-option AGENT "Definitely NOT OpenVPN"
```

Все эти параметры могут быть определены в стандартной конфигурации клиента OpenVPN или в командной строке во время выполнения.

Если вы хотите, чтобы OpenVPN повторял соединения с прокси-сервером HTTP, укажите параметр `--http-proxy-retry`. Это приведет к тому, что OpenVPN будет имитировать сброс SIGUSR1 в процессе OpenVPN, вызывая переподключение туннеля.

## SOCKS прокси

В дополнение к HTTP-прокси - OpenVPN поддерживает SOCKS-прокси. Если вы хотите больше узнать о различиях между прокси-серверами HTTP и SOCKS, в Википедии есть хорошее сравнение по адресу <https://ru.wikipedia.org/wiki/SOCKS>, или вы можете просмотреть **Рабочее предложение (RFC)** для конкретного протокола по следующим URL:

- SOCKS5: <https://www.ietf.org/rfc/rfc1928.txt>
- HTTP: <https://www.ietf.org/rfc/rfc2616.txt>

Для наших примеров мы снова предположим, что прокси-сервер 192.168.4.4 использует порт

по умолчанию 1080. Но на этот раз это будет SOCKS5 прокси. В отличие от примера прокси-сервера HTTP для туннельного транспорта со стандартным прокси-сервером SOCKS5 разрешено использовать UDP или TCP. Однако есть предостережения по этому поводу.

Добавьте следующую строку в конфигурацию клиента OpenVPN чтобы указать его для нашего примера прокси-сервера:

```
socks-proxy-server 1080 socks_auth.txt
```

Как и в случае с другими файлами аутентификации - достаточно простого текстового файла с именем пользователя и паролем в отдельных строках:

```
socks5user
socks5pass
```

Как и раньше, `stdin` работает вместо пути и имени файла для файла аутентификации. Кроме того, если требуется переподключение/повторная попытка в случае потери соединения или сбоя прокси-сервера, укажите параметр `--socks-proxy-retry`, чтобы позволить OpenVPN имитировать SIGUSR1 для перезапуска VPN.

### Подсказка

Вы можете использовать протокол SSH для создания локального сервера SOCKS5 и туннелирования OpenVPN через этот туннель. Одно из ограничений заключается в том, что для прокси-сервера SSH SOCKS5 требуются только TCP-соединения. Ваш трафик в OpenVPN может быть TCP/UDP, но сам туннель OpenVPN должен быть `--proto tcp`.

## Резюме

В этой главе вы узнали, как интегрировать OpenVPN в существующую сеть и компьютерную инфраструктуру. Это относится к серверной части. Мы также увидели как использовать LDAP в качестве бэкэнд-системы аутентификации и как использовать маршрутизацию на основе политик для плавной интеграции VPN, предлагаемой OpenVPN, в обычную сеть. На стороне клиента мы рассмотрели интеграцию OpenVPN в операционную систему Windows, а также сценарий, при котором с сервером OpenVPN нельзя связаться напрямую.

Это, конечно, лишь несколько примеров расширенных сценариев развертывания и мы ограничились только режимом `tun`. Это было сделано специально чтобы показать, что режим `tun` подходит для большинства развертываний OpenVPN. Существуют сценарии развертывания для которых требуется режим `tap` или даже мостовое соединение и они рассматриваются в следующей главе.

# Глава 6. "Режим клиент-сервер" с tap-устройствами

Другая модель развертывания для OpenVPN является - один сервер с несколькими удаленными клиентами, способных к маршрутизации трафика Ethernet. Мы называем эту модель развертывания *режимом клиент-сервер с tap-устройствами*.

Основное различие между режимами tun и tap заключается в типе используемого адаптера. tap-адаптер обеспечивает полностью виртуальный интерфейс Ethernet (уровень 2), в то время как адаптер tun рассматривается большинством операционных систем как двухточечный (уровень 3).

Компьютеры, подключенные с использованием (виртуальных) адаптеров Ethernet, могут образовывать единый широковещательный домен, который необходим для определенных приложений. С двухточечными адаптерами это невозможно. Также обратите внимание, что не все операционные системы поддерживают tap-адаптеры. Например, iOS и Android поддерживают только tun-устройства.

В этой главе мы начнем с базовой настройки клиент-сервер, которая очень похожа на базовую настройку, описанную в [Главе 4, Режим клиент-сервер с устройствами tun](#). Однако существуют тонкие различия, которые будут обсуждаться на нескольких примерах. Кроме того, режим tap включает настройку мостового соединения, при котором обычный сетевой адаптер соединяется с виртуальным tap-адаптером. Эта тема будет подробно обсуждаться как для операционных систем Linux, так и для Windows.

В этой главе будут рассмотрены следующие темы:

- Базовая настройка
- Включение трафика клиент-клиент с использованием pf
- Бриджинг
- Бриджинг в Linux
- Бриджинг в Windows
- Использование внешнего DHCP-сервера
- Проверка широковещательного и не IP-трафика
- Сравнение режима tun с режимом tap

## Базовая настройка

Базовая настройка OpenVPN в режиме tap практически такая же, как и в режиме tun. В режиме tap мы используем следующую строку в файле конфигурации сервера:

```
dev tap
```

В режиме tun мы используем следующие строки:

```
dev tun
topology subnet
```

Опция topology subnet не требуется, но представляет схему сетевой адресации, являющуюся более разумной и которая будет использоваться по умолчанию в будущей версии OpenVPN.

Для полноты картины сначала создадим файл конфигурации сервера:

```
proto udp
```

```

port 1194
dev tap
server 10.222.0.0 255.255.255.0

persist-key
persist-tun
keepalive 10 60

remote-cert-tls client
tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn-movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

user nobody
group nobody

verb 3
daemon
log-append /var/log/openvpn.log

```

Мы будем использовать этот основной файл конфигурации сервера в режиме tap в этой главе и других. Сохраните его как `tap-udp-server.conf`, чтобы мы могли использовать его позже.

### Заметка

Параметр `topology subnet` был удален, поскольку параметр `topology` является параметром конфигурации, специфичным для `tun`. В режиме `tap` сервер всегда раздает каждому клиенту по одному IP-адресу с соответствующей маской сети.

Точно так же мы создаем файл конфигурации клиента, который снова почти идентичен файлу `basic-udp-client.conf` из [Главы 4, Режим клиент-сервер с устройствами tun](#):

```

proto udp
port 1194
dev tap

client
remote openvpnserver.example.com
nobind

remote-cert-tls server
tls-auth /etc/openvpn/movpn/ta.key 1
ca /etc/openvpn/movpn-movpn-ca.crt
cert /etc/openvpn/movpn/client1.crt
key /etc/openvpn/movpn/client1.key

```

Сохраните этот файл как `tap-udp-client.conf`. Аналогично, для клиентов Windows создайте файл конфигурации `tap-udp-client.ovpn`.

Запустите сервер OpenVPN и подключите клиент, используя эти файлы конфигурации. Журнал соединений на стороне сервера покажет некоторые тонкие различия по сравнению с настройкой в режиме `tun`, которые выделены в следующем разделе:

```

OpenVPN 2.3.6 x86_64-redhat-linux-gnu [SSL (OpenSSL)] [LZO] [EPOLL]
[PKCS11] [MH] [IPv6] built on Dec 2 2014
library versions: OpenSSL 1.0.1e-fips 11 Feb 2013, LZO 2.03
[...]
TUN/TAP device tap0 opened
TUN/TAP TX queue length set to 100

```

```

do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
/sbin/ip link set dev tap0 up mtu 1500
/sbin/ip addr add dev tap0 10.222.0.1/24 broadcast 10.222.0.255
GID set to nobody
UID set to nobody
UDPV4 link local (bound): [undef]
UDPV4 link remote: [undef]

MULTI: multi_init called, r=256 v=256
IFCONFIG POOL: base=10.222.0.2 size=253, ipv6=0
Initialization Sequence Completed
CLIENT_IP:60728 TLS: Initial packet from [AF_INET]CLIENT_IP:60728,
sid=d4d7f1fd 988e4ff3
[...]
client1/CLIENT_IP:60728 PUSH: Received control message:
'PUSH_REQUEST'
client1/CLIENT_IP:60728 SENT CONTROL [client1]: 'PUSH_REPLY,routegateway
10.222.0.1,ping 10,ping-restart 60,ifconfig 10.222.0.2
255.255.255.0' (status=1)
client1/CLIENT_IP:60728 MULTI: Learn: 8e:66:e4:43:35:a1 ->
client1/CLIENT_IP:60728

```

Последняя строка журнала подключения к серверу - самая интересная: строка **MULTI: Learn** показывает, что сервер теперь использует MAC-адрес удаленного клиента, чтобы отличать его от других клиентов, тогда как в режиме tun он может полагаться исключительно на IP-адрес, назначенный клиенту. Это необходимо, так как клиент на основе tap может также отправлять не-IP трафик, в котором не используется IP-адрес.

## Включение трафика клиент-клиент

Когда к серверу подключено несколько клиентов виртуальной частной сети (VPN) - им не разрешается обмениваться трафиком. Это верно как для режима tap, так и для tun. Чтобы включить трафик между клиентами, есть два варианта:

- Использование параметра конфигурации **client-to-client**. Он позволяет OpenVPN обрабатывать трафик клиент-клиент внутри системы, минуя таблицы системной маршрутизации, а также правила системного брандмауэра/iptables.
- Использование системной таблицы маршрутизации и правил брандмауэра/iptables для отправки трафика от одного клиента другому и обратно.

Первый вариант - самый быстрый как с точки зрения конфигурации, так и с точки зрения производительности. Если нет ограничений на трафик между VPN-клиентами - добавьте строку **client-to-client** в файл конфигурации **tap-udp-server.conf**, сохраните его как **movpn-06-01-server.conf** и перезапустите сервер OpenVPN, используя этот файл конфигурации:

```
$ openvpn --config movpn-06-01-server.conf
```

Переподключите VPN-клиентов. Первому клиенту назначен IP-адрес 10.222.0.2, а второму - 10.222.0.3. Теперь клиенты могут связаться друг с другом:

```
C:\Program Files\OpenVPN\config>ping 10.222.0.2
Pinging 10.222.0.2 with 32 bytes of data:
Reply from 10.222.0.2: bytes=32 time=652ms TTL=64
Reply from 10.222.0.2: bytes=32 time=321ms TTL=64
Reply from 10.222.0.2: bytes=32 time=325ms TTL=64
Reply from 10.222.0.2: bytes=32 time=327ms TTL=64

Ping statistics for 10.222.0.2:
 Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
 Approximate round trip times in milli-seconds:
 Minimum = 321ms, Maximum = 652ms, Average = 406ms
```

Высокая задержка (то есть время проверки связи более 300 мс) на предыдущем скриншоте немедленно показывает один из недостатков использования трафика клиент-клиент по VPN. Весь трафик проходит через сервер OpenVPN, таким образом, пинг от client1 к client2 занимает больше времени:

1. Сообщение запроса ping отправляется с client1 на сервер OpenVPN.
2. Сервер OpenVPN пересыпает сообщение на client2.
3. client2 отправляет обратно ответное сообщение ping снова на сервер.
4. Сервер OpenVPN пересыпает ответ ping обратно на client1.

Если клиенты VPN подключены через сеть с высокой латентностью, то использование VPN модели клиент-сервер увеличит задержку при отправке трафика между клиентами. OpenVPN - это такая же модель VPN клиент-сервер как и большинство доступных коммерческих решений VPN. Существуют некоторые одноранговые VPN-решения, но они выходят за рамки этой книги.

## Фильтрация трафика между клиентами

Недостатком опции `client-to-client` является отсутствие фильтрации. Когда эта опция добавлена, весь трафик между всеми клиентами разрешен в обход правил системного брандмауэра/iptables.

Вторым способом обеспечения прохождения трафика между клиентами является использование таблиц маршрутизации системы. В режиме tun это сделать довольно просто, но немного сложнее при использовании режима tap. Когда client1 желает связаться с client2, он сначала должен знать MAC (аппаратный) адрес client2. Запрос ARP отправляется через адаптер tap клиента и достигает сервера OpenVPN. Серверный процесс OpenVPN перенаправляет ARP-запрос из своего собственного tap-адаптера и ожидает ответа. Однако ответ должен прийти от другого VPN-клиента, который подключен к тому же адаптеру. Таким образом - ARP-запрос должен быть отправлен всем подключенными клиентам сервера OpenVPN. Обычно повторная выдача ARP-запроса не выполняется и трафик клиент-клиент терпит неудачу.

В современных ядрах Linux (2.6.34+ или в ядрах с опциями обратного переноса) для каждого интерфейса может быть установлен специальный флаг `proxy_arp_pvlan`. Этот флаг указывает ядру Linux повторно отправить ARP-запрос обратно с того же интерфейса, откуда он поступил. Именно этот флаг необходим для работы трафика клиент-клиент. Таким образом мы включаем трафик клиент-клиент в режиме tap, не используя опцию `client-to-client`, устанавливая этот флаг:

```
echo 1 > /proc/sys/net/ipv4/conf/tap0/proxy_arp_pvlan
```

### Заметка

Этот системный флаг можно установить только после настройки адаптера tap0. Адаптер tap

может быть создан до запуска OpenVPN (см. Раздел *Мостовое соединение в Linux*) или флаг может быть установлен после запуска OpenVPN. В этом случае, он может быть установлен автоматически с использованием скрипта, как описано в [Главе 7, Скрипты и плагины](#).

Когда client1 хочет связаться с client2 - поток сетевого трафика с этим установленным флагом выглядит следующим образом:

1. client1 отправляет ARP-запрос со своего tap-адаптера.
2. Сервер OpenVPN получает ARP-запрос и перенаправляет его с собственного адаптера tap0.
3. ARP-запрос проходит через системную маршрутизацию и таблицу пересылки iptables.
4. Если запрос разрешен - он отправляется всем сетевым интерфейсам на сервере OpenVPN, включая адаптер tap0, откуда был отправлен. Последнее вызвано флагом proxy\_arp\_pvlan.
5. OpenVPN получает ARP-запрос и перенаправляет его всем подключенными клиентам OpenVPN.
6. client2 получает запрос и отвечает. ARP-ответ теперь отправляется обратно на сервер OpenVPN.
7. Сервер OpenVPN пересыпает ARP-ответ client1.
8. client1 теперь знает, где найти client2 и может отправлять сетевой трафик client2.

Второй шаг позволяет нам отфильтровать трафик между разными клиентами. Правила фильтрации (например, с использованием `iptables`) могут быть добавлены для разрешения только определенных типов трафика или только трафика между специальными клиентами. Например, следующее правило `iptables` блокирует трафик между первым и вторым клиентом OpenVPN:

```
iptables -I FORWARD -i tap0 -o tap0 -s 10.222.0.2 -d 10.222.0.3 -j DROP
```

Обратите внимание, что блокируя трафик в одном направлении, оба клиента не смогут связаться друг с другом. Для однонаправленной блокировки требуются более продвинутые правила `iptables`.

### Заметка

Нет эквивалента для флага `proxy_arp_pvlan` в операционных системах Windows или Mac OS, но это неточно.

### **Недостаток метода `proxy_arp_pvlan`**

Основным недостатком использования этого флага ядра является то, что он не превращает VPN в один широковещательный домен Ethernet. С флагом `proxy_arp_pvlan` клиенты VPN могут связываться друг с другом с помощью ARP-сообщений. Однако они не будут получать широковещательный трафик, приходящий от других клиентов. Когда используется опция `client-to-client` - все подключенные VPN-клиенты автоматически получают широковещательные сообщения друг друга, но фильтрация трафика сложнее (как мы увидим в следующем разделе).

### **Фильтрация трафика с использованием фильтра OpenVPN `pf`**

Второй метод фильтрации трафика от клиентов OpenVPN - это использование встроенного фильтра OpenVPN `pf`. Этот фильтр также полностью поддерживается в OpenVPN Access Server - коммерческом предложении от OpenVPN Technologies Inc. Поддержка фильтра `pf` является элементарной по сравнению с большинством брандмауэров, но она полностью функциональна и

доступна на всех платформах. Теперь мы пройдемся по шагам для использования этого фильтра в свободной версии OpenVPN. Этот пример приведен только в качестве доказательства концепции; станет ясно, что для обслуживания на уровне продакшена необходим другой подход и/или инструмент.

Чтобы использовать фильтр `rf` - должен использоваться интерфейс управления OpenVPN. Это достигается с помощью следующего файла конфигурации:

```
proto udp
port 1194
dev tap
server 10.222.0.0 255.255.255.0

persist-key
persist-tun
keepalive 10 60

remote-cert-tls client
tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

user nobody
group nobody

verb 3
daemon
log-append /var/log/openvpn.log

client-to-client
management 127.0.0.1 12000 stdin
management-client-auth
management-client-pf
```

Сохраните этот файл как `movpn-06-02-server.conf` и запустите сервер OpenVPN. Сервер OpenVPN запросит (новый) пароль управления. Этот пароль будет использоваться для аутентификации всех соединений с интерфейсом управления; VPN-клиенты аутентифицируются отдельно. Параметр `management-client-pf` требует чтобы также был установлен параметр `management-client-auth`. Недостатком же является то, что теперь каждый клиент должен предоставить (фальшивые) имя пользователя и пароль и каждому клиенту должен быть предоставлен доступ на стороне сервера с использованием интерфейса управления.

Файл конфигурации клиента теперь становится:

```
proto udp
port 1194
dev tap

client
remote openvpnserver.example.com
nobind

remote-cert-tls server
tls-auth /etc/openvpn/movpn/ta.key 1
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/client1.crt
key /etc/openvpn/movpn/client1.key
```

## auth-user-pass

Сохраните его как movpn-06-02-client.conf (или movpn-06-02-client.ovpn для Windows).

На стороне сервера сначала запустите интерфейс управления, используя telnet:

```
telnet 127.0.0.1 12000
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
ENTER PASSWORD:
SUCCESS: password is correct
>INFO:OpenVPN Management Interface Version 1 -- type 'help' for more info
```

Затем запустите клиент OpenVPN. Соединение с сервером не будет завершено пока клиенту не будет предоставлен доступ через интерфейс управления. В интерфейсе управления вы увидите это:

```
>CLIENT:CONNECT,0,0
>CLIENT:ENV,n_clients=0
>CLIENT:ENV,IV_VER=2.3.6
>CLIENT:ENV,IV_PLAT=linux
>CLIENT:ENV,IV_PROTO=2
[...]
```

После получения всех строк >CLIENT авторизуйте клиента для подключения. Для этого требуются **идентификатор клиента (CID)** и **идентификатор ключа (KID)**. Это параметры в самых первых строках >CLIENT при подключении клиента OpenVPN. В этом примере и CID, и KID равны 0. Чтобы предоставить этот клиентский доступ, команда client-auth-nt CID KID должна быть введена в интерфейсе управления:

```
client-auth-nt 0 0
SUCCESS: client-auth command succeeded
>CLIENT:ESTABLISHED,0CLIENT:CONNECT,0,0
```

Первый клиент OpenVPN теперь имеет доступ. Теперь мы можем применить правила контроля доступа к этому клиенту, используя команду client-pf CID. Это многострочная команда. После первой строки мы сначала указываем подсети, к которым этому клиенту разрешен доступ:

```
[SUBNETS ACCEPT]
-10.0.0.0/8
```

Мы предоставляем клиенту доступ ко всем подсетям, кроме 10.0.0.0/8.

Далее мы указываем, к каким клиентам этот клиент может обратиться:

```
[CLIENTS ACCEPT]
-client3
```

Мы разрешаем клиенту связываться со всеми другими клиентами VPN, кроме клиента с именем сертификата /CN = client3. С помощью двух операторов END, одного с квадратными скобками и одного - без, мы закрываем команду client-pf:

```
client-pf 0
[SUBNETS ACCEPT]
-10.0.0.0/8
[CLIENTS ACCEPT]
-client3
[END]
END
SUCCESS: client-pf command succeeded
```

Этот клиент OpenVPN теперь сможет обращаться ко всем подсетям на стороне сервера, кроме

10.0.0.0/8, и может связываться со всеми другими клиентами OpenVPN, кроме client3.

У этого подхода много недостатков, но он работает на всех платформах. Основными недостатками являются следующие:

- Каждый клиент должен предоставить фальшивые имя\_пользователя/пароль
- Каждый клиент должен быть аутентифицирован с использованием интерфейса управления
- Для каждого клиента должен быть установлен фильтр pf
- Интерфейс управления в настоящее время не имеет команд для просмотра текущих фильтров

В настоящее время для свободной версии OpenVPN нет инструмента для отправки этих команд в интерфейс управления. Однако коммерческое программное обеспечение OpenVPN Access Server обеспечивает необходимый механизм для применения правил фильтрации.

## Использование устройства tap (мост)

Особый вариант использования конфигурации на основе tap - мостовое соединение. Термин мостовое соединение применяется к функции операционной системы для соединения двух сетевых адаптеров вместе. Когда два (или более) адаптера соединены мостом - весь трафик Ethernet, полученный на одном из адаптеров, перенаправляется на все остальные адAPTERы, которые являются частью этого моста. Это позволяет объединить два сегмента сети вместе и создать впечатление что это один широковещательный домен Ethernet. Типичные случаи использования мостов:

- Клиенты VPN должны быть полностью и прозрачно интегрированы в LAN на стороне сервера. Обратите внимание, что тот же эффект часто может быть достигнут с помощью настройки proxy-arp.
- Некоторые старые компьютерные игры разрешают многопользовательские игры только тогда, когда все компьютеры являются частью одного и того же широковещательного домена.
- Некоторые устаревшие сетевые протоколы, особенно оригинальный протокол Microsoft NetBIOS (не основанный на TCP/IP), не работают должным образом на сетевых маршрутизаторах или даже предполагают полностью «плоское» сетевое пространство со всеми клиентами, подключенными напрямую.

Мостовые соединения также имеют недостатки, в частности, снижение производительности. Весь сетевой трафик, поступающий на один из мостовых интерфейсов, дублируется на все остальные интерфейсы. Из-за этого довольно легко перегрузить мост многоадресным или широковещательным трафиком. В особенности при настройке VPN с клиентами, использующими соединения с высокой задержкой или низкой пропускной способностью (например, работники на выезде) - эта потеря производительности может быстро сделать настройку OpenVPN непригодной для использования.

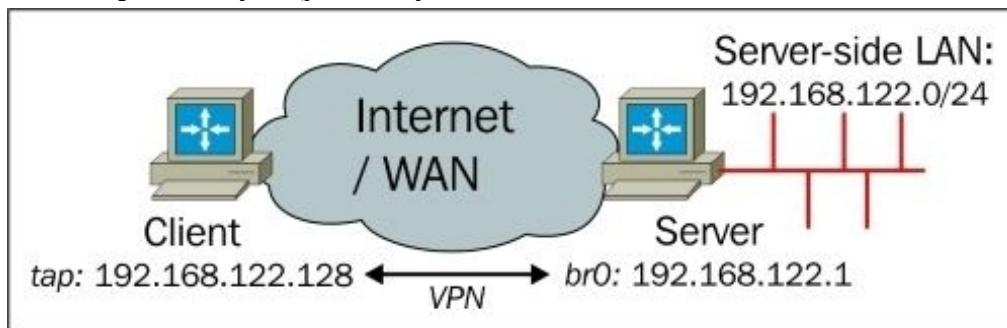
Следует также отметить, что мостовое соединение часто не требуется. Благодаря современным операционным системам и протоколам совместного доступа к файлам настройка на основе tun может достичь тех же результатов, прилагая меньше усилий и повышая производительность.

К сожалению, до сих пор распространено заблуждение что для использования общего доступа к файлам Windows через установку OpenVPN необходимо использовать мосты. В разделе *Включение общего доступа к файлам через VPN в Главе 5, Расширенные сценарии развертывания в tun режиме*, подробное объяснение дается о том, как достичь общего доступа к файлам с помощью tun-установки и сервера WINS.

В некоторых случаях мостовое соединение остается желательным или необходимым. Теперь мы покажем, как настроить мостовую конфигурацию OpenVPN на платформах Linux и Windows.

## Соединение в Linux

Рассмотрим следующую схему сети:



На стороне сервера используется сетевой мост между адаптером LAN `eth0` и виртуальным адаптером OpenVPN `tap`. В Linux это достигается созданием адаптера `tap` до запуска OpenVPN. Для этого должен быть установлен системный пакет `bridge-utils`. Шаги следующие:

1. Сначала мы создаем скрипт для запуска сетевого моста:

```
#!/bin/bash

br="br0"
tap="tap0"
eth="eth0"
br_ip="192.168.122.1"
br_netmask="255.255.255.0"
br_broadcast="192.168.122.255"
Create the tap adapter
openvpn --mktun --dev $tap
Create the bridge and add interfaces
brctl addbr $br
brctl addif $br $eth
brctl addif $br $tap
Configure the bridge
ifconfig $tap 0.0.0.0 promisc up
ifconfig $eth 0.0.0.0 promisc up
ifconfig $br $br_ip netmask $br_netmask broadcast $br_broadcast
```

2. Сохраните его как `movpn-bridge-start` и убедитесь, что он исполняется с помощью следующей команды:

```
chmod 755 movpn-bridge-start
```

3. Затем запустите мост, используя следующую команду:

```
./movpn-bridge-start
Mon Jan 5 18:40:02 2015 TUN/TAP device tap0 opened
Mon Jan 5 18:40:02 2015 Persist state set to: ON
```

4. Теперь мы создаем файл конфигурации для мостовых настроек, используя следующие команды:

```
tls-server
proto udp
port 1194

dev tap0 ## "0" чрезвычайно важен

server-bridge 192.168.122.1 255.255.255.0 192.168.122.128 192.168.122.200
```

```

remote-cert-tls client
tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

persist-key
persist-tun
keepalive 10 60

user nobody
group nobody

verb 3
daemon
log-append /var/log/openvpn.log

```

5. Сохраните его как `movpn-06-03-server.conf`. Аргументами к `server-bridge` являются сетевой шлюз, маска подсети, начало пула и конец пула. Адреса пула - это адреса, которые могут быть назначены клиентам.

### **Заметка**

Строка `dev tap0` в предыдущем примере имеет решающее значение для работы мостового соединения. Мы создали адаптер `tap` для моста до запуска OpenVPN. Чтобы использовать этот адаптер - мы должны явно указать имя адаптера. В противном случае OpenVPN создаст новый адаптер без моста при запуске.

5. Запустите сервер OpenVPN и подключите клиента с помощью файла конфигурации `tap-udp-client.conf`, созданного ранее в этой главе. На клиенте Linux в журнале подключений будет показано следующее:

```

TUN/TAP device tap0 opened
do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
/sbin/ip link set dev tap0 up mtu 1500
/sbin/ip addr add dev tap0 192.168.122.128/24 broadcast 192.168.122.255
Initialization Sequence Completed

```

Клиенту назначается первый адрес - 192.168.122.128 из пула доступных адресов.

7. Наконец, мы проверяем, что можем достичь хоста в локальной сети на стороне сервера:

```

[client]$ ping -c 4 192.168.122.246
PING 192.168.122.246 (192.168.122.246) 56(84) bytes of data.
64 bytes from 192.168.122.246: icmp_req=1 ttl=64 time=287 ms
64 bytes from 192.168.122.246: icmp_req=2 ttl=64 time=289 ms
64 bytes from 192.168.122.246: icmp_req=3 ttl=64 time=285 ms
64 bytes from 192.168.122.246: icmp_req=4 ttl=64 time=287 ms
--- 192.168.122.246 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 285.397/287.496/289.568/1.570 ms

```

### **Разрыв моста**

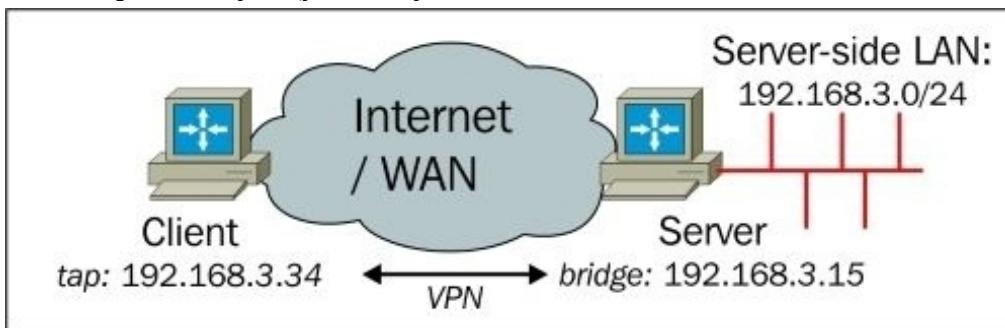
Когда процесс сервера OpenVPN останавливается - сетевой мост также не отключается автоматически. Поскольку мост был создан до запуска самого OpenVPN - он сохраняется до тех пор, пока не будет разорван вручную. Следующие команды останавливают и удаляют мост, созданный командой `movpn-start-bridge`:

```
ifconfig br0 down
```

```
brctl delif br0 eth0
brctl delif br0 tap0
brctl delbr br0
openvpn --rmtn --dev tap0
```

## Соединение в Windows

Рассмотрим следующую схему сети:

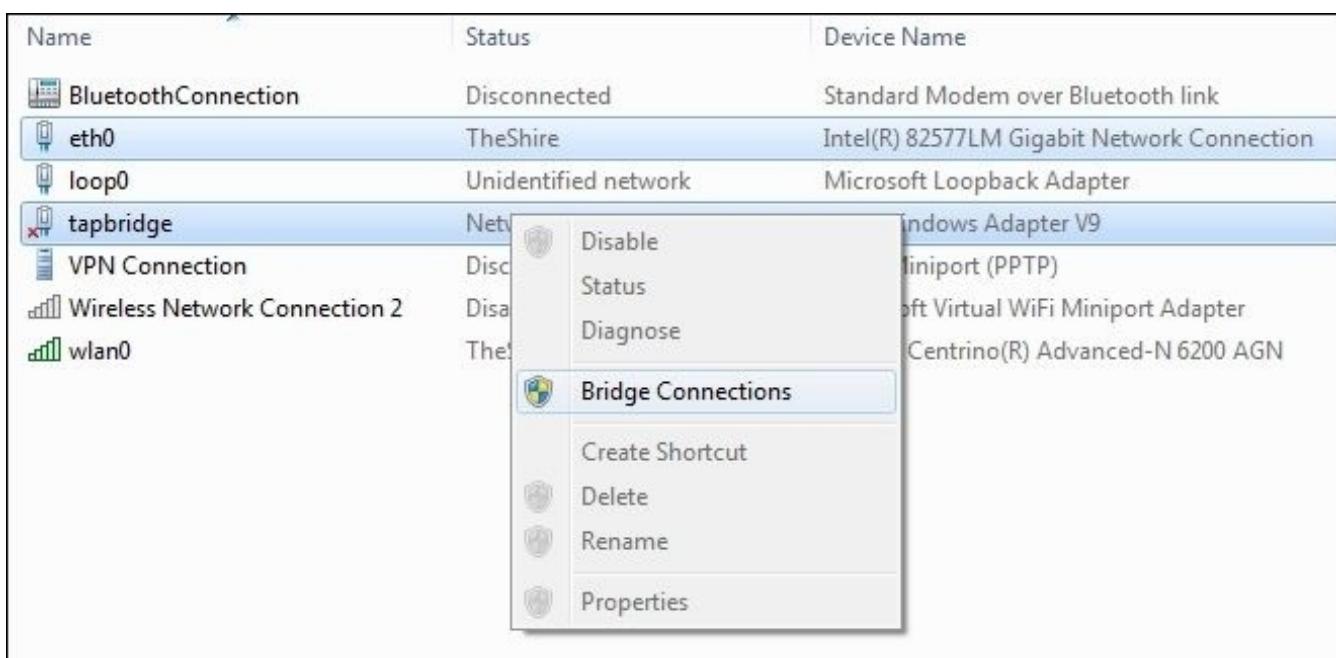


Единственная разница между предыдущим сетевым макетом и этим - выбор используемых IP-адресов.

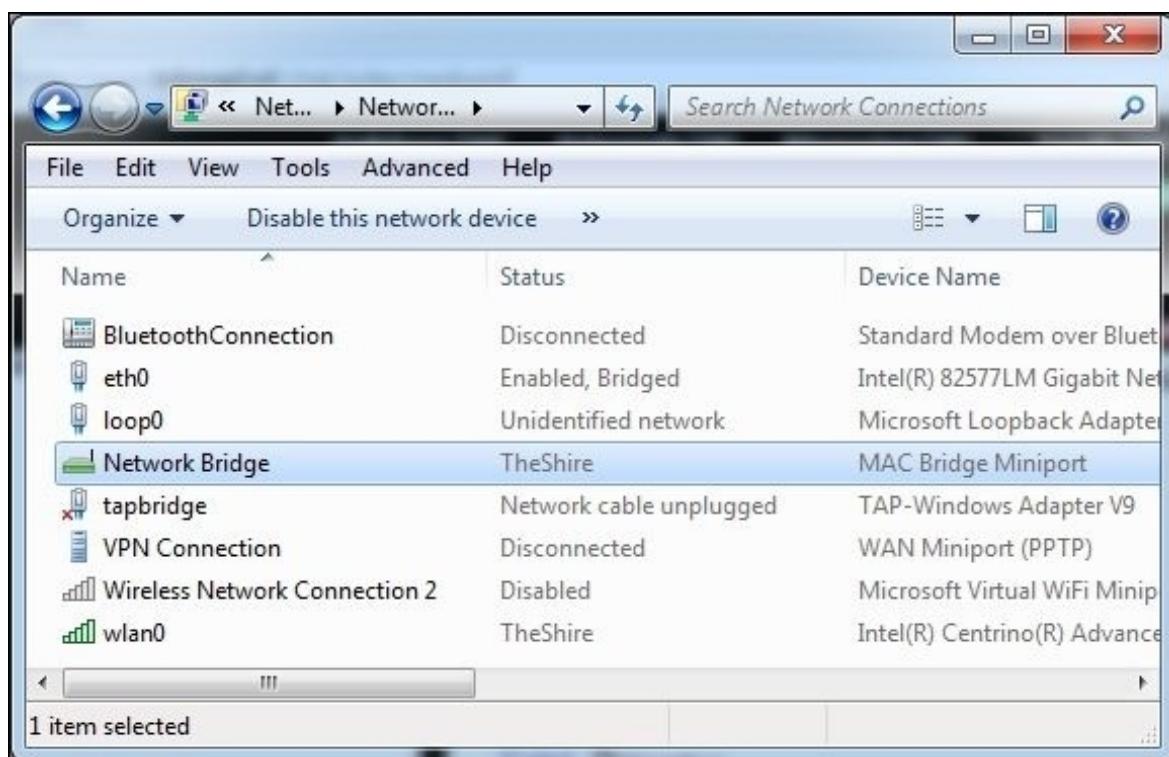
В Windows адаптер OpenVPN TAP-Win устанавливается при установке самого OpenVPN. Обычно имя для адаптера TAP-Win назначается операционной системой и будет примерно таким же, как для Подключение по локальной сети 4. Аналогично имя адаптера Ethernet, к которому подключена локальная сеть, также будет иметь имя подобное Подключение по локальной сети 2.

Для ясности (и некоторого здравого смысла) мы хотим переименовать интерфейс VPN (ТАР):

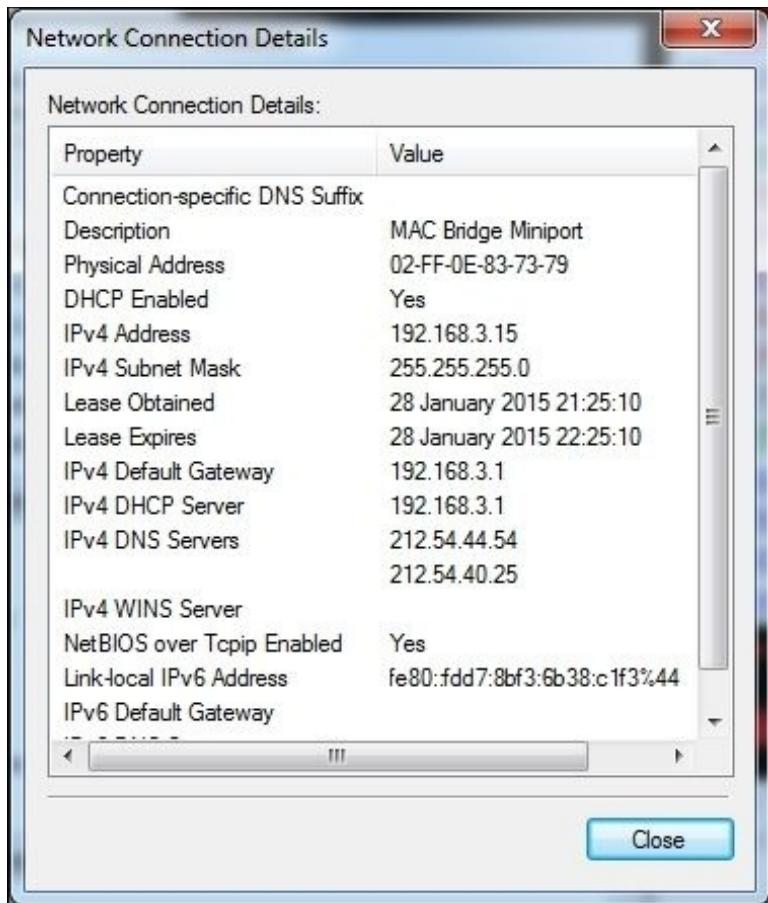
1. Сначала мы идем в **Центр управления сетями и общим доступом**, а затем **Изменение параметров адаптера**.
2. Переименуйте адаптер TAP-Win как **tapbridge** щелкнув по нему правой кнопкой мыши и выбрав **Переименовать**. На используемом тестовом компьютере адаптер Ethernet, подключенный к локальной сети, был переименован в **eth0**. В столбце **Состояние** укажите сетевую группу, к которой принадлежит интерфейс. В нашем случае он принадлежит **TheShire**.
3. Выберите два адаптера, которые необходимо соединить, зажав клавишу **Ctrl** и щелкнув по каждому адаптеру, а затем щелкнув правой кнопкой мыши и выбрав **Настройка моста**.



4. Убедитесь, что вновь созданный **Сетевой мост** является частью той же сети, что и исходный адаптер **eth0**. Вы также можете видеть что оригинальный сетевой адаптер теперь имеет метку **Мост** в поле **Состояние**:



5. Нет необходимости настраивать статический IP-адрес для моста. **Сетевой мост** имеет свой собственный (виртуальный) MAC-адрес (или **Физический адрес** на следующем скриншоте) и, следовательно, ему присваивается свой собственный IP-адрес DHCP-сервером в локальной сети на стороне сервера:



6. Создайте файл конфигурации сервера OpenVPN с помощью текстового редактора или Блокнота:

```
tls-server
proto udp
port 1194

dev tap
dev-node tapbridge ## == имя адаптера TAP-Win

server-bridge 192.168.3.15 255.255.255.0 192.168.3.128 192.168.3.250

remote-cert-tls client
tls-auth "c://program files/openvpn/config/ta.key" 0
dh "c://program files/openvpn/config/dh2048.pem"
ca "c://program files/openvpn/config/movpn-ca.crt"
cert "c://program files/openvpn/config/server.crt"
key "c://program files/openvpn/config/server.key"

persist-key
persist-tun
keepalive 10 60

verb 3
```

Сохраните файл конфигурации как `movpn-06-04-server.ovpn` в каталоге конфигурации OpenVPN (обычно это `C:\Program Files\OpenVPN\config`).

#### Заметка

Файл конфигурации сервера для версии OpenVPN для Windows аналогичен файлу конфигурации для Linux. Основными отличиями являются полные пути к файлам сертификатов и ключей, а также способ указания адаптера TAP-WIN с помощью ключевых

слов dev и dev-node. Также обратите внимание, что параметры user/group и daemon/logging были удалены.

7. Запустите сервер OpenVPN (с повышенными привилегиями):

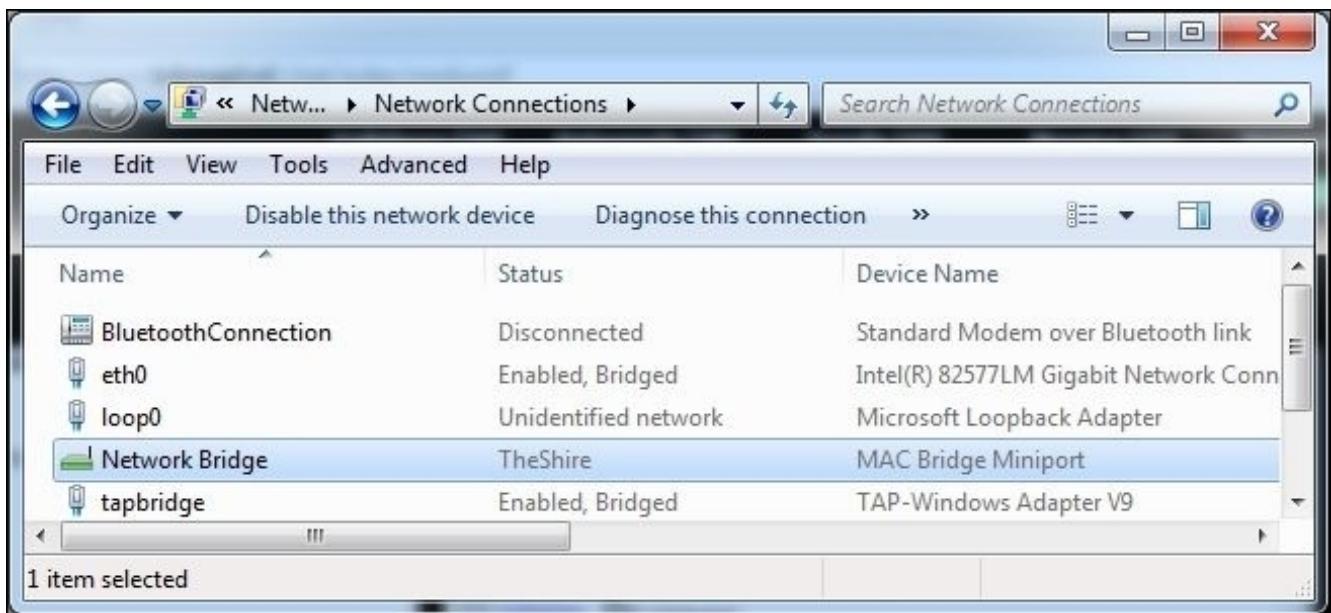
```
C:> cd \program files\openvpn\config
C:> ..\bin\openvpn --config mopenvpn-06-04-server.ovpn
```

Обратите внимание, что версия OpenVPN для Windows в командной строке выглядит и ведет себя почти так же, как версия для командной строки Linux.

8. Брандмауэр Windows отобразит предупреждение безопасности, когда OpenVPN попытается настроить VPN. Нажмите **Разрешить доступ**, чтобы предоставить разрешение OpenVPN для настройки VPN как показано на следующем снимке экрана:

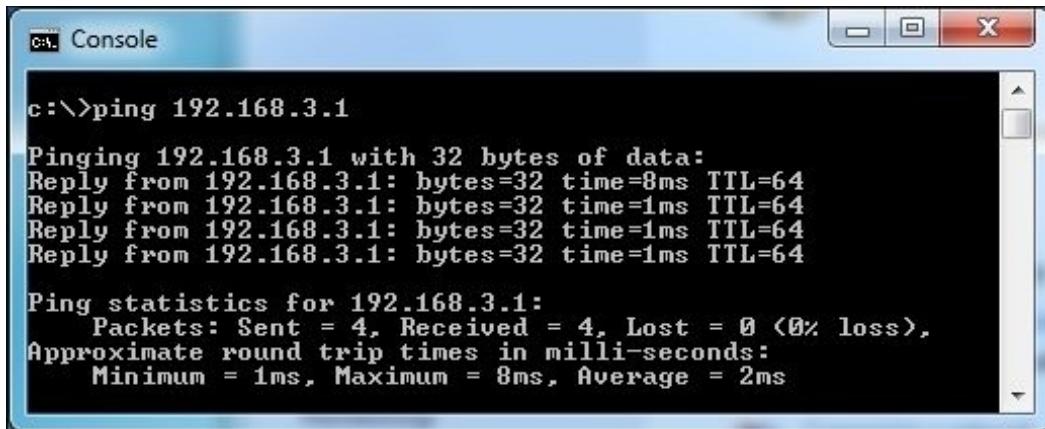


9. Если мы вернемся к экрану **Настройки адаптера** - теперь мы видим что и адаптер локальной сети **eth0** и адаптер TAP-Win **tapbridge** в состоянии **Включен** и имеют статус **Мост**. Это показано на следующем снимке экрана:



10. Затем подключите клиент Windows с помощью файла конфигурации **tap-udp-client.ovpn**, созданного ранее в этой главе. Клиенту будет назначен первый адрес - 192.168.3.128 из пула доступных адресов.

11. Наконец, мы проверяем, что можем достичь хоста в локальной сети на стороне сервера:



12. На сервере нажмите функциональную клавишу **F4** в командном окне для остановки процесса сервера OpenVPN. Для настройки производственного уровня желательно запускать и останавливать OpenVPN с помощью службы OpenVPN, устанавливаемой вместе с OpenVPN.

## Использование внешнего DHCP-сервера

В мостовой конфигурации можно еще больше интегрировать клиентов в серверную сеть. В большинстве сетей для назначения IP-адресов используется DHCP-сервер. Обычно OpenVPN назначает IP-адреса своим клиентам с помощью одной из следующих команд:

```
server 10.200.0.0 255.255.255.0
```

Или с помощью следующей команды:

```
server-bridge 192.168.3.15 255.255.255.0 192.168.3.128 192.168.3.250
```

Также можно использовать внешний DHCP-сервер для назначения адресов клиентам OpenVPN. Для этого просто удалите спецификацию любых диапазонов IP-адресов после параметра **server-bridge** как показано в следующем (ориентированном на Linux) файле конфигурации:

```
tls-server
proto udp
port 1194
dev tap0 ## the '0' is extremely important

server-bridge

remote-cert-tls client
tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

persist-key
persist-tun
keepalive 10 60

user nobody
group nobody

verb 3
daemon
log-append /var/log/openvpn.log
```

Сохраните его как **movpn-06-05-server.conf** и запустите сервер OpenVPN.

Когда клиент подключается и запрашивает IP-адрес с помощью DHCP - запрос будет перенаправлен на сервер DHCP в локальной сети на стороне сервера. DHCP-сервер назначает адрес, который отправляется обратно клиенту через сервер OpenVPN.

На клиенте OpenVPN это можно проверить, посмотрев IP-адрес соединения **vpn0**:



Чтобы убедиться, что этот адрес был назначен сервером DHCP на стороне сервера, мы проверяем таблицу клиентов DHCP на сервере DHCP:

DHCP Client Table					
DHCP Server IP Address: 192.168.3.1					
MAC Address	IP Address	Subnet Mask	Duration	Expires	
0011e6dead07	192.168.3.11	255.255.255.0	D:00 H:01 M:00 S:00	Wed Jan 28 23:05:45 2015	
5c260a307224	192.168.3.14	255.255.255.0	D:00 H:01 M:00 S:00	Wed Jan 28 23:16:38 2015	
00ff178255db	192.168.3.34	255.255.255.0	D:00 H:01 M:00 S:00	Wed Jan 28 23:20:13 2015	

Третья запись в **DHCP Client Table** на предыдущем скриншоте содержит **MAC Address** адаптера TAP-Win клиента OpenVPN. Это доказывает, что серверный DHCP-сервер назначил адрес клиенту OpenVPN.

## Проверка широковещательного и не IP-трафика

Инструменты tcpdump и wireshark полезны для устранения неполадок в "почти работающей" настройке OpenVPN. Wireshark доступен для Linux, Mac OS X и Windows. Его можно использовать как инструмент командной строки, но чаще всего используется версия с графическим интерфейсом. На большинстве Unix/Linux-платформ также доступен инструмент командной строки tcpdump.

Теперь мы будем использовать tcpdump и wireshark для просмотра потока пакетов через установку VPN на основе tap.

## Протокол разрешения адресов трафика

Одним из самых основных типов трафика Ethernet, присутствующего во всех сетях, является трафик **протокола разрешения адресов (ARP)**. ARP является ярким примером протокола Ethernet, который не передается по двухточечным каналам связи (например, при настройке OpenVPN на основе туннеля). Физический уровень (уровень 1) обычно представляет собой электрическую или оптическую связь между системами. В случае VPN туннель занимает место этого физического соединения. Следующим шагом в модели OSI является уровень Ethernet (уровень 2). Протокол ARP часто используется для обнаружения других систем на этом уровне.

### Подсказка

Ethernet является сетевым протоколом уровня 2, а точка-точка - сетевым протоколом уровня 3. Различные уровни протокола определяются моделью [Open Systems Interconnection \(OSI\)](#).

Чтобы наблюдать за потоком ARP-трафика мы сначала запускаем сервер OpenVPN, используя ранее созданный файл конфигурации `ovpn-06-01-server.conf`. Затем подключаем двух клиентов Linux к серверу. После того, как все соединения были успешно установлены - мы запускаем `tcpdump` на одном из клиентов:

```
tcpdump -nnel -i tap0
```

Теперь мы отправляем один пакет ping от одного клиента - другому и смотрим на вывод

tcpdump:

```
ecrist@honeybadger - csh - 76x24
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tap0, link-type EN10MB (Ethernet), capture size 65535 bytes
20:00:53.216803 de:c4:8d:29:a4:ee > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806)
, length 42: Request who-has 10.200.0.11 tell 10.200.0.10, length 28
20:00:53.251952 00:bd:48:d9:dc:00 > de:c4:8d:29:a4:ee, ethertype ARP (0x0806)
, length 42: Reply 10.200.0.11 is-at 00:bd:48:d9:dc:00, length 28
20:00:53.251995 de:c4:8d:29:a4:ee > 00:bd:48:d9:dc:00, ethertype IPv4 (0x0800)
, length 98: 10.200.0.10 > 10.200.0.11: ICMP echo request, id 54985, seq 0
, length 64
20:00:53.287295 00:bd:48:d9:dc:00 > de:c4:8d:29:a4:ee, ethertype IPv4 (0x0800)
, length 98: 10.200.0.11 > 10.200.0.10: ICMP echo reply, id 54985, seq 0,
length 64
20:00:54.221075 de:c4:8d:29:a4:ee > 00:bd:48:d9:dc:00, ethertype IPv4 (0x0800)
, length 98: 10.200.0.10 > 10.200.0.11: ICMP echo request, id 54985, seq 1
, length 64
20:00:54.257353 00:bd:48:d9:dc:00 > de:c4:8d:29:a4:ee, ethertype IPv4 (0x0800)
, length 98: 10.200.0.11 > 10.200.0.10: ICMP echo reply, id 54985, seq 1,
length 64
20:00:55.223086 de:c4:8d:29:a4:ee > 00:bd:48:d9:dc:00, ethertype IPv4 (0x0800)
, length 98: 10.200.0.10 > 10.200.0.11: ICMP echo request, id 54985, seq 2
, length 64
20:00:55.258556 00:bd:48:d9:dc:00 > de:c4:8d:29:a4:ee, ethertype IPv4 (0x0800)
, length 98: 10.200.0.11 > 10.200.0.10: ICMP echo reply, id 54985, seq 2,
length 64
```

На снимке экрана показан трафик ARP между client1 (в данном случае 10.200.0.10) и client2 (10.200.0.11).

- Первый пакет в выводе выше от клиента, которым был инициирован ping. Клиент должен знать Ethernet MAC-адрес компьютера, который мы проверяем, и, следовательно, он отправляет запрос ARP.
- Поскольку мы указали client-to-client в файле конфигурации сервера `tnovpn-06-01-server.conf` - ARP-запрос пересыпается всем подключенным клиентам и второй клиент OpenVPN отвечает своим MAC-адресом.
- Второй пакет - это ответ от второго клиента, указывающий его собственный MAC-адрес.
- Теперь, когда адрес известен, client1 отправляет ping. Он отображается как IPv4 ICMP echo request.
- Ответ получен от второго клиента. Это четвертый пакет (IPv4 ICMP echo reply).

## Трафик NetBIOS

**Common Internet File Sharing (CIFS)** начинался как закрытый протокол NetBEUI. Поддержка совместного использования файлов и принтеров осуществлялась через протокол Novell **Internetwork Packet eXchange (IPX)**, а затем был добавлен протокол TCP/IP. В настоящее время протокол обмена файлами Windows развился и поддерживается только через TCP/IP. Поддержка устаревших файловых протоколов все еще присутствует в более старых версиях Windows и именно эту устаревшую поддержку мы будем использовать для запуска трафика не-IP.

Сначала мы устанавливаем и включаем транспортный протокол NWLink IPX/SPX на адаптере TAP-WIN. Затем мы подключаем клиента Windows к установке OpenVPN, которая была запущена с использованием файла конфигурации `tnovpn-06-01-server.conf`. В этой

конфигурации включен `client-to-client`; таким образом, все подключенные клиенты должны видеть весь широковещательный трафик Ethernet, поступающий от этого клиента.

Когда клиент Windows успешно подключится к VPN-серверу - он начнет отправлять трафик для объявления своего имени и другой информации об обмене файлами Windows. Он попытается сделать это через TCP/IP, но также с использованием сообщений IPX.

Мы используем Wireshark на втором VPN-клиенте и отслеживаем трафик на интерфейсе tap. На следующем снимке экрана показано, что Windows-клиент WINDOWSXP действительно отправляет широковещательный трафик NetBIOS через TCP/IP. Это записи с адресом источника 10.222.0.3 и адресом назначения 10.222.0.255. Последний адрес является широковещательным адресом TCP/IP для настроенной нами VPN. Мы также видим, что трафик передается по протоколу IPX. Этот трафик выбран и выделен на скриншоте:

The screenshot shows the Wireshark interface with the following details:

- File Edit View Go Capture Analyze Statistics Telephony Tools Help**
- Toolbar:** Includes icons for file operations, search, and analysis.
- Filter:** Set to "Expression...".
- Table Headers:** No., Time, Source, Destination, Protocol, Info.
- Data:** A list of network frames. Frame 22 is highlighted in blue. The "Info" column shows entries like "Registration NB WINDOWSXP<20>" and "Host Announcement WINDOWSXP, Workstation".
- Frame Details:** Shows the selected frame (Frame 22) with the following details:
  - Frame 22: 234 bytes on wire (1872 bits), 234 bytes captured (1872 bits)
  - Type: IEEE 802.3 Ethernet
  - Subtype: Logical-Link Control
  - Protocol: Internet Packet eXchange
  - Source: 00:00:00:00:00:ff (10.222.0.3)
  - Destination: ff:ff:ff:ff:ff:ff (10.222.0.255)
- Hex Editor:** Shows the raw bytes of the selected frame.
- Statistics:** Shows 53 total packets, 53 displayed, and 0 marked.
- Profile:** Default.

Широковещательные сообщения IPX являются широковещательными сообщениями *Ethernet*, но они не основаны на IP. Это показывает, что установка OpenVPN в режиме tap с `client-to-client` разделяет весь трафик Ethernet, включая широковещательный трафик между подключенными клиентами (и самим сервером OpenVPN).

## Сравнение режима tun с режимом tap

Как мы уже видели в этой главе - есть много сходств, но также есть и существенные различия между VPN в режиме tun и VPN в режиме tap. В этом разделе мы обсудим эти сходства и различия. Большинство различий проистекает из единственного факта что VPN в режиме tun является не широковещательной, а двухточечной IP-сетью, в то время как сеть в режиме tap обеспечивает полностью виртуальную Ethernet-подобную сеть с поддержкой широковещания. Короче говоря, сеть в режиме tun обеспечивает сетевое подключение уровня 3, тогда как сеть в

режиме tap обеспечивает практически все функциональные возможности сети уровня 2.

Особенно с опцией `topology subnet` настройка на основе TUN напоминает установку без перемычек:

- Опция `server 10.200.0.0 255.255.255.0` устанавливает VPN с адресом сервера 10.200.0.1. Каждый клиент получит один адрес из 24 IP-адресной адресации, начиная с 10.200.0.2.
- Способ шифрования VPN-трафика и цифровой подписи (HMAC) идентичен.
- Большинство возможностей сценариев применимы к обоим типам VPN. Однако есть некоторые тонкие различия в параметрах для сценария `client-connect`.
- При правильной настройке конечный пользователь не будет испытывать различий между настройкой на основе tun и VPN-подключением на основе tap.

Эти различия, конечно, гораздо интереснее обсуждать. Некоторые различия очевидны, но есть и некоторые тонкости, которые могут оказывать существенное влияние при настройке VPN.

## Слой 2 против слоя 3

В сети уровня 2 (т.е. в режиме tap) соседние клиенты могут связаться друг с другом узнав адрес соседа, используя широковещательные ARP-запросы. Широковещательные ARP-запросы позволяют клиентам обнаруживать MAC-адреса других клиентов. Это позволяет клиентам связываться друг с другом по протоколам IP и не-IP.

В сети уровня 3 (в режиме tun) клиенты могут связываться друг с другом только с помощью IP-адресов. MAC-адрес адаптера tun никогда не раскрывается другим VPN-клиентам и даже самому серверу OpenVPN. Из-за этого сетевой пакет уровня 3 немного короче, чем уровня 2. При нормальных обстоятельствах более длинные сетевые пакеты уровня 2 не оказывают негативного влияния на производительность.

## Маршрутные различия и iroute

Когда особенно необходима маршрутизация от подсети к подсети между tun и tap есть некоторые существенные различия. В сети в режиме tun необходим файл конфигурации клиента с соответствующим оператором `irooute`, чтобы позволить VPN-серверу получать доступ к клиентам, находящимся в локальной сети на стороне клиента. В качестве примера мы предполагаем, что подсеть 192.168.3.0/24 может быть достигнута через клиента OpenVPN с сертификатом CN=client1. На сервере OpenVPN мы добавили бы файл `client-config-dir` с именем `client1`, содержащий инструкцию:

```
irooute 192.168.3.0 255.255.255.0
```

И добавили бы системный маршрут в файл конфигурации сервера:

```
route 192.168.3.0 255.255.255.0
```

При настройке в режиме tap оператор `irooute` недопустим и будет просто игнорироваться сервером. Чтобы достичь подсети за VPN-клиентом необходимо добавить системный маршрут на сервере OpenVPN, а шлюз должен указывать на VPN-IP-адрес клиента. Давайте предположим, что для `client1` из предыдущего примера назначен фиксированный IP-адрес. Это может быть достигнуто с помощью файла CCD:

```
ifconfig-push 10.200.0.99 255.255.255.0
```

В файле конфигурации сервера необходимо добавить маршрут, чтобы таблицы системной маршрутизации знали, что подсеть 192.168.3.0/24 может быть доступна через клиента 10.200.0.99:

```
route 192.168.3.0 255.255.255.0 10.200.0.99
```

Это гораздо менее динамично, чем опция режима tun route + iroute.

## Фильтрация клиент-клиент

При настройке в режиме tun большая часть трафика может регистрироваться и фильтроваться с использованием правил брандмауэра или iptables. Фильтрация трафика между клиентами OpenVPN намного сложнее при настройке в режиме tap, как было показано ранее в этой главе.

## Широковещание трафика и "болтливость" сети

Сеть уровня 3 не позволяет передавать широковещательный трафик по ней. Это и преимущество и недостаток. Некоторые клиент-серверные приложения полагаются на использование широковещательного трафика для связи между сервером и клиентами. Для таких приложений требуется сеть в режиме tap.

Однако широковещательный трафик также имеет тенденцию засорять сети. Даже если на клиенте нет действий пользователя - операционная система будет непрерывно отправлять широковещательный трафик для обнаружения сетевых ресурсов, соседей и т.д. Особенно, когда используются такие протоколы как Universal Plug-and-Play или Apple Bonjour существует много скрытого широковещательного трафика. Для клиентов, подключенных к VPN через сеть с низкой пропускной способностью - это может иметь серьезные последствия для производительности.

## Мост

Ключевой особенностью сети в режиме tap является возможность создания мостов. Мостовое соединение невозможно в сети уровня 3.

В некоторых редких случаях эта функция абсолютно необходима, но при любой возможности следует избегать мостовой настройки. Основной причиной неиспользования мостовой настройки является негативное влияние на производительность. Как объяснялось ранее - в мостовой конфигурации весь трафик из локальной сети на стороне сервера перенаправляется через VPN всем клиентам и наоборот. Когда множество клиентов подключены к сетям с низкой пропускной способностью - это может привести к загрузке всей сети как на стороне клиента, так и в локальной сети на стороне сервера. Когда клиенты в локальной сети на стороне сервера пытаются обнаружить доступные ресурсы в сети (например, общие файловые ресурсы или принтеры в сети на основе CIFS) - вся сеть будет заполнена широковещательным трафиком. Клиенты локальной сети обычно ждут ответов от всех компьютеров, подключенных к сети, как локальной, так и VPN, прежде чем предлагать доступ к общим сетевым ресурсам или принтерам. Это может быстро привести к недопустимому времени отклика сети в случае когда подключается и отключается большое количество VPN-клиентов.

## Резюме

В этой главе мы рассмотрели возможности установки на основе tap в качестве альтернативной модели развертывания OpenVPN. Мы обсудили примеры, подчеркивающие как особенности, так и недостатки такой установки. Особое внимание было уделено настройке с использованием моста, так как существует несколько распространенных заблуждений относительно режима с использованием моста, о чём говорится на форумах поддержки OpenVPN в Интернете.

Мы также увидели, что расширенные функции управления, такие как фильтрация трафика между клиентами OpenVPN, гораздо сложнее реализовать в режиме tap по сравнению с режимом tun.

В следующей главе мы увидим, как можно использовать скрипты и плагины чтобы влиять на то,

как сервер OpenVPN назначает IP-адрес клиенту, а также на многие другие функции. Скрипты и плагины могут использоваться как в режиме tap, так и в режиме tun.

# Глава 7. Скрипты и плагины

После развертывания персональная или корпоративная VPN может стать мощным инструментом как с точки зрения безопасности, так и с точки зрения функциональности. Хорошо спроектированная VPN позволит пользователям безопасно подключаться к удаленным ресурсам. Иногда, однако, просто иметь VPN недостаточно. Данное приложение может требовать более строгих стандартов безопасности или требовать лучшего мониторинга и контроля.

Интеграция плагинов и сценариев с OpenVPN может решить многие из этих организационных или функциональных потребностей. В этой главе будет продемонстрировано как плагины можно использовать для улучшения аутентификации и как скрипты могут отслеживать соединения, генерировать таблицы маршрутизации и многое другое.

## Скриптинг

Скрипты, вероятно, один из лучших инструментов, доступных для администратора OpenVPN. Имея возможность назначать как клиентские, так и серверные скрипты, OpenVPN может инициировать другие ответы системы, открывая брандмауэры, запуская приложения или даже отправляя сообщение администратору.

Одним важным предупреждением при написании скриптов является время, необходимое для завершения сценария. OpenVPN - это однопоточный процесс - это означает, что во время работы скрипта вся VPN блокируется для всех подключенных или подключающихся клиентов. Скрипт медленной аутентификации может нанести вред хорошо работающему VPN. Плагины меньше подвержены этому влиянию, так как работают в отдельном потоке.

Начиная с версии 2.3.6 OpenVPN поддерживает 13 параметров скриптов на стороне сервера и 10 параметров на стороне клиента. Команды со звездочками являются параметрами настройки и позволяют следующим параметрам выполнять определенные действия. Серверные скрипты выглядят следующим образом (в порядке выполнения):

- --setenv\*
- --setenv-safe\*
- --script-security\*
- --up-restart\*
- --up
- --route-up
- --tls-verify
- --auth-user-pass-verify
- --client-connect
- --learn-address
- --client-disconnect
- --route-pre-down
- --down

На стороне клиента сценарии выглядят следующим образом (в порядке выполнения):

- --setenv\*

- `--script-security*`
- `--up-restart*`
- `--tls-verify`
- `--ipchange`
- `--setenv-safe*`
- `--up`
- `--route-up`
- `--route-pre-down`
- `--down`

Теперь мы кратко рассмотрим все эти параметры, объяснив их функции как на стороне сервера, так и на стороне клиента. Далее в этой главе мы предоставим подробный пример и обсудим поведение и тонкости каждого из этих сценариев.

## **Серверные скрипты**

Давайте посмотрим на сценарии, используемые на стороне сервера.

### **`--setenv` и `--setenv-safe`**

Параметры `setenv` и `setenv-safe` используются для установки переменных среды, которые могут использоваться как скриптами, так и плагинами. Опция `setenv` позволяет нам устанавливать практически любые переменные окружения, но эту опцию нельзя "передать" клиентам. Опция `setenv-safe` добавляет к каждой переменной среды префикс `OPENVPN_`, избегая конфликтов с системными переменными среды, такими как `PATH` и `LD_LIBRARY_PATH`. Эта опция может быть передана клиентам, что обеспечивает большую гибкость.

### **`--script-security`**

Опция `script-security` определяет какие типы приложений или скриптов могут быть выполнены из конфигурации OpenVPN. Существует четыре варианта уровней безопасности:

- **0** : означает, что никакие внешние скрипты или программы не разрешены. На компьютере с Linux/Unix это приводит к тому, что OpenVPN не работает, поскольку OpenVPN всегда нужно запускать некоторые внешние команды для установки IP-адреса. На клиентах Windows, однако, допускается работа OpenVPN в этом режиме, при условии, что шлюз по умолчанию не изменен. Для этого необходимо вызвать внешнее приложение.
- **1** : означает, что разрешены определенные встроенные исполняемые файлы, например, `ip`, `route`, `ifconfig` и другие. Это по умолчанию.
- **2** : наиболее часто требуемый уровень безопасности. Он позволяет использовать не только встроенные команды, подобные перечисленным в предыдущем пункте, но и пользовательские сценарии.
- **3** : позволяет паролям передаваться в вызываемые скрипты через переменные окружения. Это может быть небезопасно, но полезно для определенных сценариев аутентификации или даже операций смены пароля.

### **`--up-restart`**

Опция `up-restart` - это просто флаг, который можно установить. Если этот флаг установлен, то

при перезапуске OpenVPN вызываются оба сценария: down и up (в указанном порядке).

#### **--up**

Сценарий up - это первый скрипт, выполняемый после того, как OpenVPN выполнил свою первоначальную инициализацию. Обычно этот сценарий запускается сразу после того, как OpenVPN привязал себя к настроенному сетевому порту и коснулся открытия устройства TUN или TAP. На данный момент в процессе запуска ни один клиент не подключен к серверу и авторизация еще не состоялась. Некоторые люди используют сценарии up для инициализации прокси-серверов и/или правил брандмауэра.

#### **--route-up**

После того, как устройство TUN или TAP было открыто, выполняется скрипт маршрутизации для настройки любых системных маршрутов на стороне сервера.

#### **--tls-verify**

Всякий раз, когда клиент подключается к серверу, первым сценарием, который будет выполняться как на клиенте, так и на сервере, является сценарий `tls-verify`. Этот сценарий вызывается несколько раз, по одному разу для каждого сертификата, который клиент предоставляет серверу. На этом этапе удаленный узел все еще считается ненадежным. Его можно использовать для проверки информации о сертификате клиента или сервера до проверки подлинности. Если скрипт `tls-verify` возвращает ненулевой код выхода, клиентское соединение отклоняется.

#### **--auth-user-pass-verify**

Помимо относительно простой аутентификации клиентского SSL-сертификата, OpenVPN поддерживает довольно надежный набор инструментов для аутентификации по имени пользователя и паролю. Этот аргумент принимает два аргумента: команду и ее метод. Метод определяет, как OpenVPN передает учетные данные аутентификации команде. Метод может быть как `via-env`, так и `via-file`. Чтобы использовать опцию `via-env` для поддержки скрипта необходимо установить третий (3) параметр безопасности скрипта. Если скрипт `auth-user-pass-verify` возвращает ненулевой код завершения – клиентское соединение отклоняется.

Важно знать, что сценарий `auth-user-pass-verify` также выполняется всякий раз, когда клиент перезапускается или ему необходимо пересмотреть параметры безопасности с сервером. Повторное согласование ключей безопасности обычно происходит каждый час, но им можно управлять с помощью параметров `reneg-sec`, `reneg-pkts` и `reneg-bytes`.

#### **--client-connect**

Выполняется после аутентификации клиента на VPN-сервере. Большинство сценариев запускаются здесь. Скрипт `client-connect` передает один аргумент, который является именем временного файла. После завершения сценария файл обрабатывается OpenVPN и все содержимое анализируется как дополнительные параметры конфигурации. Это позволяет администратору добавлять специальные настройки для конкретного клиента, обеспечивая большую гибкость, чем файл CCD. Один из наших примеров в этой главе использует сценарий подключения клиента для обновления базы данных, используемой для отслеживания и направления статистики VPN-подключений.

#### **--learn-address**

Сценарий `learn-Address` позволяет OpenVPN помогать определять правила брандмауэра и другие специфичные для адреса опции. Он выполняется всякий раз, когда новый клиент добавляется, обновляется или удаляется из внутренних таблиц адресов OpenVPN. Более

подробная информация доступна на странице руководства. Эта опция поддерживает как IPv4, так и IPv6. Сценарий изучения адреса фактически вызывается отдельно для адресов IPv4 и IPv6 один раз с адресом IPv4 в качестве основного параметра и один раз с адресом IPv6.

### **--client-disconnect**

Как и в случае сценария `client-connect` в предыдущем разделе, сценарий `client-disconnect` является вторым наиболее часто используемым сценарием соединения. Как упоминалось ранее, в одном из примеров в этой главе `client-disconnect` клиента будет использоваться для обновления записей базы данных статистикой использования и другой информацией.

### **--route-pre-down**

Когда началось отключение туннеля, запускается сценарий `route-pre-down`. Его можно использовать для автоматизации отключения удаленных прокси-серверов и закрытия дыр в брандмауэре, которые были открыты/установлены ранее в сценарии `--up`.

### **--down**

В противоположность опции `up` – `down` запускает команду после закрытия устройства TUN/TAP. Эта опция принимает команду или скрипт в качестве аргумента, а дополнительные аргументы передаются скрипту или команде. Если вам нужно выполнить команду до закрытия устройства TUN/TAP – вы можете использовать опцию `down-pre`.

#### **Заметка**

Нет способа запустить скрипт как до, так и после закрытия устройства TUN/TAP.

## **Клиентские скрипты**

Мы обсудим клиентские скрипты в этом разделе.

### **--setenv и --setenv-safe**

Действуют точно так же, как в предыдущем разделе *Серверные скрипты*.

### **--script-security**

Действует точно так же, как в предыдущем разделе *Серверные скрипты*.

### **--up-restart**

Опция `up-restart` – это просто флаг, который можно установить. Если этот флаг установлен, то при перезапуске OpenVPN вызываются оба сценария: `down` и `up` (в указанном порядке).

### **--tls-verify**

Всякий раз, когда клиент подключается к серверу, первым сценарием, который будет выполняться как на клиенте, так и на сервере, является сценарий `tls-verify`. Этот сценарий вызывается несколько раз, по одному разу для каждого сертификата, который сервер представляет клиенту. На этом этапе удаленный узел все еще считается ненадежным. Его можно использовать для проверки информации о сертификате клиента или сервера до проверки подлинности. Если скрипт `tls-verify` возвращает ненулевой код выхода, клиентское соединение отклоняется.

### **--ipchange**

На стороне клиента `ipchange` выполняется сразу после скрипта `tls-verify`, а также при

любом изменении удаленного (также известного как доверенный) IP-адреса. Его можно использовать для обновления правил брандмауэра или прокси-сервера перед открытием адаптера TUN/TAP.

#### **--up**

Сценарий `up` является первым сценарием, выполняющимся после завершения аутентификации клиента. После того, как сервер успешно аутентифицировал клиента, клиенту отправляется набор параметров конфигурации. Эти параметры включают IP-адрес VPN для использования, а также любые другие параметры, передаваемые клиенту. Некоторые люди используют сценарии `up` для инициализации прокси-серверов и/или правил брандмауэра.

#### **--route-up**

После проверки подлинности и установления маршрутов выполняется сценарий `route-up`. При желании он может быть задержан на определенное количество секунд с помощью опции `route-delay`.

#### **--route-pre-down**

Когда началось отключение туннеля, запускается сценарий `route-pre-down`. Он может быть использован для закрытия соединений с удаленными прокси-серверами, другими туннелями (SSH) или исправления записей DNS-сервера.

#### **--down**

В противоположность опции `up`, опция `down` запускает команду после закрытия устройства TUN/TAP. Эта опция принимает команду или скрипт в качестве аргумента, а дополнительные аргументы передаются скрипту или команде. Если вам нужно выполнить команду до закрытия устройства TUN/TAP – вы должны использовать опцию `down-pre`.

#### **Заметка**

Нет способа запустить скрипт как до, так и после закрытия устройства TUN/TAP.

## **Примеры серверных скриптов**

Скрипты сервера могут быть использованы для значительного расширения развертывания OpenVPN. Они могут использоваться для аутентификации, авторизации, регистрации и многое другое. В сочетании с опциями `client-config`, сценарии могут быть далее использованы для генерации директив конфигурации клиента на лету. Например, аутентификация может происходить через LDAP, а правила авторизации могут быть динамическими через тот же каталог LDAP. Правила межсетевого экрана могут быть сгенерированы и применены и маршруты могут быть переданы клиенту. В этом разделе будут продемонстрированы некоторые из этих вещей, чтобы помочь вам в применении данных методов.

Наиболее распространенными серверными сценариями являются сценарии `--client-connect` и `--client-disconnect`. Эти сценарии могут использоваться для многих целей, включая открытие наборов правил брандмауэра, монтирование файловых систем и даже создание файлов конфигурации клиента на лету.

В сочетании с планировщиком задач другие сценарии могут быть запущены вне прямого контекста OpenVPN, но могут по-прежнему работать с подключенными клиентами через интерфейс управления OpenVPN. Например, администратор может предоставить выделенное время конечным пользователям и отключить пользователей после того, как это выделенное время было исчерпано.

## Сценарии подключения клиента

Давайте теперь посмотрим на сценарии подключения клиента.

### Аутентификация клиента

Аутентификация – это определение того, кто может подключиться. Она не определяет что могут делать эти пользователи, просто им разрешено подключаться к VPN или нет. На самом базовом уровне вполне возможно разрешить пользователю подключаться, но не позволено этому пользователю фактически что-либо делать. Одним из применений для авторизации может быть сценарий мониторинга, который просто хочет убедиться, что ваш VPN-сервер работает и аутентифицирует пользователей. Этот псевдопользователь не должен иметь возможность маршрутизировать трафик, поскольку в этом нет необходимости.

Многие переменные и права доступа можно проверить с помощью сценариев аутентификации, включая смарт-карты, серверы LDAP или RADIUS, информацию о сертификатах, списках отзыва сертификатов и многом другом. Если вы читали эту книгу с самого начала, в [Главе 5, Расширенные сценарии развертывания в tun-режиме](#), вы должны иметь встроенные конфигурации сервера VPN, который уже использует LDAP и другие движки. С помощью сценариев вы можете расширить поддержку этих бекендов без текущего плагина или запросить дополнительные источники.

Наиболее распространенным сценарием сервера, как правило, является `--client-connect`. Этот скрипт выполняется после того, как вся проверка TLS прошла. В тех случаях, когда сценарий подключения к клиенту имеет много общего, это идеально для предотвращения атаки типа **Denial of Server (DoS)**, вызванной вашими собственными сценариями. Перед его запуском клиент был проверен на наличие надлежащего ключа `tls-auth` и действительного сертификата. Сценарий `client-connect` может использоваться как своего рода предварительная аутентификация, поскольку он выполняется перед сценарием `--auth-user-pass-verify`. Он также может быть использован для динамического создания конфигурации клиента.

При использовании сценария убедитесь, что для параметра `--script-security` установлено значение 2 или 3 (см. определения ранее в этой главе). Если не указать эту опцию – клиенту будет возвращено сообщение `AUTH_FAIL`. В качестве примера мы создали очень простой скрипт, который печатает среду оболочки и завершает работу с нулем (0), что указывает на успех запуска демона OpenVPN. В нашей конфигурации сервера мы добавили следующие две директивы:

```
script-security 2
client-connect /usr/local/etc/openvpn/cc.sh
```

#### Заметка

Существует переменная окружения `script_type`, определяющая тип вызываемого скрипта. Используя эту переменную, можно иметь один монолитный скрипт для обработки всех вызовов скриптов OpenVPN.

Вот наш пример скрипта `client-connect`:

```
#!/bin/sh
printenv > /tmp/movpn
exit 0
```

Код выхода важен, поскольку все, кроме нуля (0), приведет к отключению клиента. С помощью предыдущего скрипта и конфигурации, при появлении нового клиентского соединения наш скрипт будет выполнен. Он разрешает соединение, но печатает среду оболочки во временный файл. Содержимое этого файла интересно и будет применяться ко всем остальным скриптам:

```
daemon_start_time=1425344172
```

```

daemon_pid=4004
local_1=SERVER_IP
trusted_ip=CLIENT_IP
redirect_gateway=0
untrusted_port=1194
tun_mtu=1500
X509_0_ST=Enlightenment
X509_0_CN=client1
X509_0_emailAddress=root@example.org
time_ascii=Mon Mar 2 18:56:17 2015
proto_1=udp
X509_1_emailAddress=root@example.org
tls_id_0=C=ZA, ST=Enlightenment, O=Mastering OpenVPN, CN=client1,
emailAddress=root@example.org
tls_id_1=C=ZA, ST=Enlightenment, L=Overall, O=Mastering OpenVPN,
CN=Mastering OpenVPN, emailAddress=root@example.org
ifconfig_ipv6_local=2001:db8:100::1
untrusted_ip=CLIENT_IP
daemon=1
tls_serial_hex_0=02
trusted_port=1194
dev_type=tun
tls_serial_hex_1=d2:93:32:f0:8e:bc:58:ee
X509_1_ST=Enlightenment
X509_1_CN=Mastering OpenVPN
script_context=init
tls_serial_0=2
PWD=usrlocaletcopenvpn
daemon_log_redirect=1
tls_serial_1=15173527578309581038
ifconfig_local=10.200.0.1
dev=tun0
local_port_1=1194
time_unix=1425344177
link_mtu=1541
remote_port_1=1194
X509_0_C=ZA
tls_digest_0=1b:27:a6:b4:5f:7a:9c:3f:17:fb:ff:33:05:61:3f:2a:56:89:
16:d3
tls_digest_1=e4:f1:43:37:34:51:de:99:7a:dc:e3:6d:f2:4c:5b:84:34:4b:
f3:64
script_type=client-connect
X509_1_C=ZA
ifconfig_broadcast=10.200.0.255
ifconfig_pool_remote_ip=10.200.0.2
ifconfig_ipv6_remote=2001:db8:100::2
ifconfig_ipv6_netbits=64
ifconfig_netmask=255.255.255.0
config=usrlocaletcopenvpn/openvpn.conf
ifconfig_pool_netmask=255.255.255.0
X509_0_O=Mastering OpenVPN
X509_1_L=Overall
verb=4
common_name=client1
X509_1_O=Mastering OpenVPN

```

Используя эти переменные среды, опытный администратор OpenVPN может настроить конфигурацию в на первый взгляд простой настройке сервера.

## Авторизация клиента

Авторизация может происходить в нескольких местах, включая файл `client-config-dir` или

дополнения могут быть сделаны через скрипт `client-connect`.

### Заметка

Аутентификация доказывает, кто вы есть. Авторизация определяет что вам разрешено делать.

Первым аргументом, передаваемым сценарию `client-connect` - будет путь к временному файлу, который скрипт может использовать для передачи параметров конфигурации подключающегося клиента демону OpenVPN. Этот скрипт проверяет переменную среды `common_name` и, если это `client1` - устанавливает `disable` в конфигурации клиента:

```
#!/bin/sh
if ["$common_name" = "client1"];
then
 echo "disable" >> $1
fi

exit 0
```

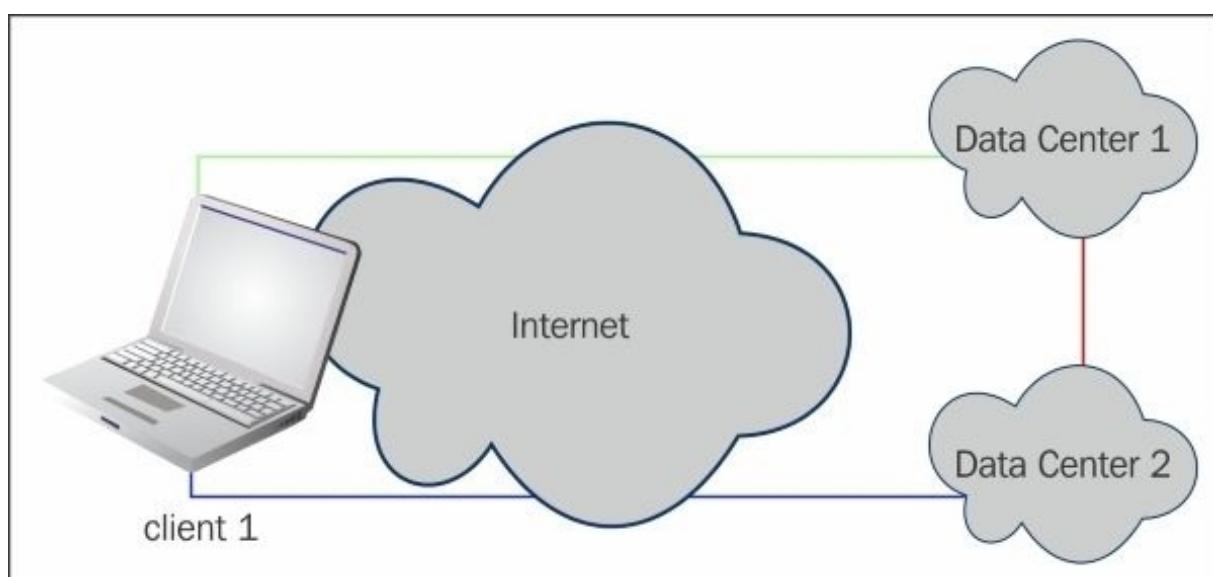
Когда `client1` подключается, опция отключения будет передана на сервер, предотвращая продолжение соединения. Могут быть переданы другие параметры, такие как `ifconfig` для статических IP-адресов, передача различных маршрутов и многое другое.

### Пример 1 - выбранные клиентом маршруты

Рассмотрим сетевого администратора с двумя центрами обработки данных, каждый со своей парой серверов OpenVPN. Был случай, когда необходимо было работать с одним центром обработки данных, при этом гарантируя, что если другой стал недоступен, службы и системы все еще были бы доступны в работающем центре обработки данных.

Чтобы помочь в тестировании, мы создали несколько сценариев (`client-connect` и `auth-user-pass-verify`), чтобы выбрать, какие маршруты были переданы клиенту.

Следующая диаграмма должна дать приблизительное представление о концепции. У VPN-клиента есть три варианта конфигурации: полные маршруты (full), маршруты для **Data Center 1** (dc1) и для **Data Center 2** (dc2). Кроме того, клиент может подключиться к любому центру обработки данных и получить только необходимые маршруты для любого из этих трех.



В нашей схеме `client1` подключается к центру обработки данных и запрашивает их имя пользователя и пароль. В действительности мы игнорируем пароль и читаем имя пользователя, чтобы определить желаемые маршруты.

Чтобы зафиксировать выбор маршрута, мы использовали скрипт `auth-user-pass-verify`.

OpenVPN принимает два аргумента: путь к сценарию и `via-file` или `via-env`, чтобы определить как передавать учетные данные в скрипт. В этом примере мы выбрали `via-file`.

Сценарий считывает учетные данные из скрипта и переписывает их в файл, который будет доступен сценарию подключения клиента:

```
#!/bin/sh
echo 'head -n1 $1' > \
/tmp/openvpn-${untrusted_ip}-${untrusted_port}.tmp
exit 0
```

Затем запускается наш скрипт подключения к клиенту, который читает созданный ранее файл `.tmp`. На основе прочитанного аргумента он записывает выбор маршрута в файл, переданный `client-connect` для аргументов конфигурации:

```
#!/bin/sh
creds="tmpopenvpn-${untrusted_ip}-${untrusted_port}.tmp"
if [-f "$creds"];
then
 selected='head -n1 $creds'
 if ["$selected" = "dc1"];
 then
 cat >> $1 <<- EOF
 push "route 10.10.0.0 255.255.255.0"
 push "route 10.10.1.0 255.255.255.0"
 EOF
 elif ["$selected" = "dc2"];
 then
 cat >> $1 <<- EOF
 push "route 10.20.0.0 255.255.255.0"
 push "route 10.20.1.0 255.255.255.0"
 EOF
 else
 cat >> $1 <<- EOF
 push "route 10.10.0.0 255.255.255.0"
 push "route 10.10.1.0 255.255.255.0"
 push "route 10.20.0.0 255.255.255.0"
 push "route 10.20.1.0 255.255.255.0"
 EOF
 fi
fi
exit 0
```

Хотя это и не идеально и на это можно нападать разными способами, но оно позволило нам создать одну конфигурацию OpenVPN для каждого из системных администраторов и обеспечить им некоторый уровень динамической маршрутизации в зависимости от их задач.

## Пример 2 - отслеживать статистику клиентских подключений

Используя скрипты `client-connect` и `client-disconnect` возможно записывать статистику клиентских подключений в базу данных или другое место. В этом примере мы просто хотим отслеживать, когда пользователь подключился и сколько времени он провёл, подключившись к VPN-серверу.

Мы предполагаем, что вы, по крайней мере, знакомы с SQL и поэтому не будем фокусироваться на семантике. SQLite 3 используется в следующем примере для хранения информации о сеансе. Схема для нашего примера базы данных выглядит следующим образом:

```
CREATE TABLE vpn_session (
session_id integer primary key autoincrement not null,
cn test not null,
connect_time timestamp default CURRENT_TIMESTAMP,
```

```

disconnect_time timestamp default null,
vpn_ip4 char(15),
vpn_ip6 char(40),
remote_ip4 char(40),
connection_time integer default 0
);

```

Схема может быть загружена в базу данных с помощью следующей команды:

```
ecrist@example:~-> sqlite3 movpn.sqlite3 < file.schema
```

### Подсказка

Каталог и файл, который вы используете для SQLite, должны быть доступны для чтения и записи пользователю OpenVPN. Если в вашем файле конфигурации определены `--user` или `--group` этому пользователю понадобится этот доступ. Без него, ваши скрипты `client-connect` и `client-disconnect` не смогут обновить базу данных.

На этот раз, мы собираемся создать скрипт, вызываемый как для `client-connect`, так и для `client-disconnect`. Мы будем обнаруживать и обрабатывать тип сценария в коде. Для типа `client-connect` мы собираемся вставить новую запись для нового сеанса. В случае отключения клиента мы обновим эту запись, чтобы обеспечить дальнейший учет.

Код выглядит следующим образом:

```

#!/bin/sh

DBFILE="/var/openvpn/movpn.sqlite3"
DBBUFFER="/var/openvpn/buffer.sql"

db_query (){
 SQL="$1"
 /usr/local/bin/sqlite3 $DBFILE "$SQL"
 if [$? -ne 0];
 then
 # There was an error, write the SQL out to a buffer file
 echo "$SQL" | tr -d "\t" | tr -d "\n" | tee -a $DBBUFFER
 echo ";" | tee -a $DBBUFFER
 fi
}
logger "OpenVPN Type: $script_type"
case "$script_type" in
 client-connect)
 # do record insert
 logger "OpenVPN: client-connect"
 SQL="
 INSERT INTO vpn_session (
 cn, connect_time, vpn_ip4,
 vpn_ip6, remote_ip4
) VALUES (
 '$common_name', '$time_unix',
 '$ifconfig_pool_remote_ip',
 '$ipconfig_ipv6_remote',
 '$untrusted_ip'
)
 "
 db_query "$SQL"
 ;;
 client-disconnect)
 # update the record, if it's found
 logger "OpenVPN: client-disconnect"
 SQL="

```

```

UPDATE
 vpn_session
SET
 disconnect_time = '$time_unix'
WHERE
 cn = '$common_name'
 AND disconnect_time IS NULL
 AND session_id =
 (
 SELECT MAX(session_id)
 FROM vpn_session
 WHERE cn = '$common_name'
)
"
db_query "$SQL"
;;
esac

exit 0

```

Этот скрипт использует функцию переключения регистра, чтобы определить тип сценария и вести себя соответственно. При новом клиентском соединении он обновит таблицу базы данных с информацией о соединении и обновит эту запись базы данных, когда клиент отключится. Схема, которую мы здесь использовали, довольно проста и может быть легко расширена для поддержки отслеживания использования полосы пропускания и нескольких клиентских подключений.

Используя команду `sqlite3`, мы можем получить три самые последние записи в базе данных:

```

ecrist@example:/usr/local/etc/openvpn-> sqlite3
/var/openvpn/movpn.sqlite3 "SELECT * FROM vpn_session ORDER BY session_id
DESC LIMIT 3"
10|client1|1426430834||10.200.0.2||CLIENT_IP|
9|client1|1426430759|1426430759|10.200.0.2||CLIENT_IP|0
8|client1|1426429888|1426429888|10.200.0.2||CLIENT_IP|0

```

### Пример 3 - отключить пользователя через X минут

Есть несколько способов справиться с этим сценарием. Если вы предоставляете пользователям короткий доступ, скажем, 30 минут за раз, сценарий `client-connect` с простым режимом ожидания может выполнить отключение и блокировку учетной записи. Допустим, вы продаете короткие одноразовые сеансы VPN продолжительностью до 30 минут. В этом случае после того, как пользователь достиг 30 минут использования, задание `cron` отключит пользователя и заблокирует его учетную запись, используя запись в каталоге CCD.

Этот пример будет основываться на предыдущем примере с использованием базы данных SQLite, которую мы создали для отслеживания используемого времени. Наш скрипт будет иметь несколько задач:

- Рассчитать время подключения VPN
- Блокировка пользователя по истечении выделенного времени подключения
- Отключение клиента, если в он данный момент подключен

Для выполнения предыдущих задач мы напишем небольшой скрипт оболочки, который будет вызываться демоном `cron`. Здесь мы проверим информацию о соединении, запросив порт управления и базу данных, которую мы создали в предыдущем примере. Альтернативой запросу порта управления может быть опрос файла журнала состояния OpenVPN и работа с этими данными в режиме реального времени. Один серьезный недостаток для опроса интерфейса управления заключается в том, что он является однопоточным и допускает только одно

соединение за раз. Если скрипт зависает или кто-то подключается к интерфейсу, последовательные опросы также будут зависать. Код выглядит следующим образом:

```
#!/bin/sh
#
Определяет, был ли пользователь ($1) подключен более $2 секунд
if [$# -lt 2];
then
 echo "usage: $0 <user> <time_in_seconds>"
 exit 1
fi

USER=$1
T0=$2

DB секунды

SQL="SELECT SUM(connection_time) FROM vpn_session WHERE cn='\$USER'"
DBTIME='/usr/local/bin/sqlite3 /var/openvpn/movpn.sqlite3 "$SQL"'

if ["\$DBTIME" = ""];
then
 DBTIME=0
fi

Проверка порта управления
CTIME='echo "status 2" | nc -N localhost 1194 | grep -E
"CLIENT_LIST.*\$USER" | cut -f8 -d,
if ["\$CTIME" != ""];
then
 # у нас есть активное соединение
 D='date +"%s"'
 CTIME='expr "\$D - \$CTIME"'
else
 CTIME=0
fi

UTIME='expr \$DBTIME + \$CTIME'
if [\$UTIME -gt \$T0];
then
 logger "Disconnecting \$USER, activity time exceeded (\$UTIME/\$T0)."
 echo "disable" >> /usr/local/etc/openvpn/ccd/\$USER
 echo "kill \$USER" | nc localhost 1194
fi
```

Теперь, когда у нас есть скрипт, мы можем вызвать его:

```
root @ example:~-> timeout.sh client1 1800
```

Он подсчитает сколько секунд client1 был подключен к VPN. Если время превышает 1800 (или любое другое число, указанное вами), он отключит эту конфигурацию через директорию client-config-directory и уничтожит все активные сеансы, используя интерфейс управления.

### Подсказка

Интерфейс управления OpenVPN допускает только одно соединение за раз. Убедитесь, что ваш скрипт правильно обрабатывает это ограничение.

## **Примеры клиентских скриптов**

Многие сторонние клиентские пакеты OpenVPN интенсивно используют клиентские сценарии для обеспечения надежной интеграции с различными операционными системами. Tunnelblick, первоначально написанный Анджело Лауб, использует клиентский сценарий для интеграции настроек DNS сервера OpenVPN с операционной системой Mac OS X.

Название "client" для клиентских сценариев может быть немного неправильным. Во многих случаях клиент OpenVPN может фактически быть другим сервером. Возможно, у вас есть несколько разрозненных офисов и вы используете OpenVPN для их подключения. Клиентские сценарии могут использоваться для запуска демона, процесса резервного копирования или других служб, для которых локальная сеть зависит от сеанса OpenVPN.

Клиентские сценарии написаны аналогично серверным и имеют почти идентичный список доступных переменных среды.

### **Пример 4 - монтирование общего ресурса NFS**

Обычной задачей подключения клиента является предоставление удаленных общих ресурсов после подключения к корпоративной сети. Давайте рассмотрим веб-разработчика, которому необходимо подключить веб-каталог после подключения к VPN.

Первая задача состоит в том, чтобы написать на стороне клиента сценарии up и down. Сценарий up подключит сетевой ресурс, а down – удалит его. Сценарий up в этом случае довольно прост – монтирует директорию webroot через NFS от 10.200.0.53. Этот пример написан для системы Mac OS X, использующей osascript для предоставления графических всплывающих окон для уведомления пользователя, когда смонтирован общий ресурс NFS. Код выглядит следующим образом:

```
#!/bin/sh

создать webroot в home, если он не существует
mkdir -p ~/remote_shares/webroot
if [$? -eq 0];
then
 mount 192.168.19.53:/webroot ~/remote_shares/webroot
 if [$? -eq 0];
 then
 osascript -e 'tell app "System Events" to display dialog
"~/remote_shares/webroot is mounted"'
 else
 osascript -e 'tell app "System Events" to display dialog
"Unable to mount webroot directory."'
 fi
else
 osascript -e 'tell app "System Events" to display dialog "Unable to
create remote share path."'
fi
```

Следующий скрипт довольно прост:

```
#!/bin/sh
umount -f ~/remote_shares/webroot
```

После создания сценариев необходимо обновить конфигурацию клиента, чтобы разрешить внешние сценарии, добавив директиву script-security вместе с директивой up:

```
script-security 2
up /путь/к/скрипту/up.sh
```

## **Подсказка**

Tunnelblick выполняет довольно много собственных сценариев и переопределяет оба вызова сценариев - up и down. Чтобы обойти это, следуйте инструкциям для именования и расположения сценариев. Эти инструкции можно найти по адресу <https://tunnelblick.net/cUsingScripts.html>.

### Пример 5 - использование всех скриптов одновременно

В следующем примере мы будем использовать все сценарии – клиентские и серверные одновременно. Хотя это не похоже на ситуацию в реальной жизни, но дает некоторое хорошее представление о порядке выполнения скриптов, а также об аргументах и переменных среды для каждого из них.

Мы начнем со следующего файла конфигурации сервера:

```
tls-server
proto udp
port 1194
dev tun

server 10.200.0.0 255.255.255.0
server-ipv6 FD00::200:0/112

ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key
dh /etc/openvpn/movpn/dh2048.pem
tls-auth /etc/openvpn/movpn/ta.key 0

persist-key
persist-tun
keepalive 10 60

topology subnet

user nobody
group nobody

daemon
log-append /var/log/openvpn.log

route 10.100.0.0 255.255.0.0

route 192.168.0.0 255.255.255.0

ask-pass /etc/openvpn/movpn/secret
script-security 3
cd /etc/openvpn/movpn
setenv MASTERING_OPENVPN server
push "setenv-safe SPECIAL hack"
up ./movpn-07-01-script.sh
tls-verify ./movpn-07-01-script.sh
auth-user-pass-verify ./movpn-07-01-script.sh via-env
client-connect ./movpn-07-01-script.sh
route-up ./movpn-07-01-script.sh
client-disconnect ./movpn-07-01-script.sh
learn-address ./movpn-07-01-script.sh
route-pre-down ./movpn-07-01-script.sh
down ./movpn-07-01-script.sh
```

Мы сохраним его как movpn-07-01-server.conf. Вот некоторые заметки об этом файле

конфигурации:

- Маршруты, указанные в конфигурации сервера, предназначены только для демонстрационных целей, как мы увидим позже.
- Чтобы обойти ошибку в OpenVPN 2.3.7, мы добавили следующую строку:  
`ask-pass /etc/openvpn/movpn/secret`
- В этом файле `secret` (парольная фраза для расшифровки закрытого ключа сервера) хранится в виде открытого текста.
- В этой серверной конфигурации мы также использовали следующую опцию, чтобы перейти в каталог, где расположен скрипт:  
`cd /etc/openvpn/movpn`

Это делает серверную конфигурацию короче и проще для чтения.

### Заметка

С опцией `cd` было бы возможно указать опции `ca`, `cert`, `key`, `dh` и `tls-auth`, используя более короткий путь, например:  
`ca ./movpn-ca.crt`

Тем не менее, рекомендуется всегда использовать абсолютные имена путей или параметр `--cd` и относительные пути для элементов, связанных с безопасностью во избежание путаницы.

- Мы также установили переменную среды на стороне сервера `MASTERING_OPENVPN` со значением `server`, используя следующую команду:  
`setenv MASTERING_OPENVPN server`
- Мы передаем безопасную переменную среды всем клиентам, используя следующую команду:  
`push "setenv-safe SPECIAL hack"`
- Внутри клиентских скриптов и плагинов эта переменная должна отображаться как `OPENVPN_SPECIAL`.

Далее мы создаем следующий скрипт:

```
#!/bin/bash

exec >> /tmp/movpn-07-01.log 2>&1
date +"%H:%M:%S: START $script_type script ==="
echo "argv = $0 $@"
echo "user = 'id -un'`id -gn`"
env | sort | sed 's^/ /'
date +"%H:%M:%S: END $script_type script ==="
```

Мы сохраним его как `movpn-07-01-script.sh` в каталоге `/etc/openvpn/movpn`. Убедитесь, что скрипт исполняемый, создайте пустой файл журнала и запустите `openvpn`:

```
chmod 0755 /etc/openvpn/movpn/movpn-07-01-script.sh
touch /tmp/movpn-07-01.log
chown nobody /tmp/movpn-07-01.log
openvpn --config /etc/openvpn/movpn/movpn-07-01-server.conf
```

Прежде чем мы продолжим, взглянем на файл `/tmp/movpn-07-01.log`. Когда OpenVPN запускается, некоторые скрипты уже были выполнены:

```

15:46:57: START up script ===
argv = ./movpn-07-01-script.sh tun0 1500 1541 10.200.0.1
 255.255.255.0 init
user = root/root
[...]
15:46:57: END up script ===
15:46:57: START route-up script ===
argv = ./movpn-07-01-script.sh
user = root/root

```

Сценарии `up` и `route-up` были выполнены. Параметры, переданные в скрипт `up` представили имя устройства TUN/TAP (`tun0`), `tun-mtu` (1500), и значения `link-mtu` (1541), IP-адрес VPN и маску сети (10.200.0.1/255.255.255.0) и тип вызова (возможные значения: `init` или `restart`).

Обратите внимание, что оба сценария были выполнены с привилегиями «`root`». Мы кратко рассмотрим вывод файла журнала каждого скрипта.

На стороне клиента мы настроили аналогичную конфигурацию. Сначала создайте следующий файл конфигурации:

```

client
proto udp

remote openvpnserver.example.com
port 1194
dev tun
nobind

remote-cert-tls server
tls-auth /etc/openvpn/movpn/ta.key 1
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/client1.crt
key /etc/openvpn/movpn/client1.key

persist-tun
persist-key

explicit-exit-notify 3
auth-user-pass

script-security 3
cd /etc/openvpn/movpn
setenv MASTERING_OPENVPN client
tls-verify ./movpn-07-01-script.sh
ipchange ./movpn-07-01-script.sh
up ./movpn-07-01-script.sh
up-restart
route-up ./movpn-07-01-script.sh
route-pre-down ./movpn-07-01-script.sh
down ./movpn-07-01-script.sh

```

Сохраните его как `movpn-07-01-client.conf` и заново создайте или скопируйте файл `movpn-07-01-script.sh` с сервера.

Опять же, мы гарантируем, что скрипт является исполняемым, создаем пустой файл журнала и запускаем `openvpn`:

```

chmod 0755 /etc/openvpn/movpn/movpn-07-01-script.sh
openvpn --config /etc/openvpn/movpn/movpn-07-01-client.conf
OpenVPN 2.3.7 x86_64-redhat-linux-gnu [SSL (OpenSSL)] [LZO] [EPOLL]

```

```

[PKCS11] [MH] [IPv6] built on Jun 9 2015
library versions: OpenSSL 1.0.1e-fips 11 Feb 2013, LZO 2.08
Enter Auth Username: *****
введите "movpn"
Enter Auth Password: *****
введите "secret"
NOTE: the current --script-security setting may allow this configuration to
call user-defined scripts
Control Channel Authentication: using 'etcopenvpn/movpn/ta.key' as a OpenVPN
static key file
UDPV4 link local: [undef]
UDPV4 link remote: [AF_INET]<IP>:1194
WARNING: this configuration may cache passwords in memory -- use
the auth-nocache option to prevent this
[Mastering OpenVPN Server] Peer Connection Initiated with [AF_INET]
<IP>:1194
TUN/TAP device tun0 opened
do_ifconfig, tt->ipv6=1, tt->did_ifconfig_ipv6_setup=1
usrsbin/ip link set dev tun0 up mtu 1500
usrsbin/ip addr add dev tun0 10.200.0.2/24 broadcast 10.200.0.255
usrsbin/ip -6 addr add fd00::200:1000/112 dev tun0
./movpn-07-01-script.sh tun0 1500 1541 10.200.0.2 255.255.255.0
init
Initialization Sequence Completed

```

Вы можете заполнить как захотите для Auth username и Auth password, так как серверный скрипт всегда будет возвращать успех.

После установления соединения мы проверяем, что VPN-клиент и сервер могут связаться друг с другом, используя ping и ping6.

Затем мы перезапускаем VPN-соединение, отправляя специальный сигнал OpenVPN клиенту:

```
killall -USR1 openvpn
```

Это вызовет «программный сброс» клиента OpenVPN. После восстановления соединения мы еще раз проверяем, что VPN полностью функционирует. Программные сбросы происходят в реальных ситуациях, в основном, когда опция persist-tun используется на стороне клиента, а клиент OpenVPN находится в мобильной сети с роумингом, или когда сеть между клиентом и сервером не очень стабильна. В этом сценарии серверные сценарии вызываются с немного другими параметрами, как мы увидим ниже.

Наконец, закройте VPN-соединение, завершив работу клиента. Теперь мы пройдемся по журналам сценариев на сервере и клиенте.

## **Журнилизование серверных сценариев**

Журнал серверных сценариев может легко вырасти до тысяч строк, но, к счастью, в нем есть некоторая структура. Давайте сначала проверим порядок, в котором сценарии вызываются:

```

15:46:57: START up script ===
argv = ./movpn-07-01-script.sh tun0 1500 1541 10.200.0.1
255.255.255.0 init
15:46:57: START route-up script ===
argv = ./movpn-07-01-script.sh
15:47:15: START tls-verify script ===
argv = ./movpn-07-01-script.sh 1 C=ZA, ST=Enlightenment,
L=Overall, O=Mastering OpenVPN, CN=Mastering OpenVPN,
emailAddress=root@example.org
15:47:15: START tls-verify script ===
argv = ./movpn-07-01-script.sh 0 C=ZA, ST=Enlightenment,
O=Mastering OpenVPN, CN=client1,

```

```

 emailAddress=root@example.org
15:47:15: START user-pass-verify script ===
argv = ./movpn-07-01-script.sh
15:47:15: START client-connect script ===
argv = ./movpn-07-01-script.sh
 /tmp/openvpn_cc_5b1f0d25ac0f71c98c44ec128e5c21d6.tmp
15:47:15: START learn-address script ===
argv = ./movpn-07-01-script.sh add 10.200.0.2 client1
15:47:15: START learn-address script ===
argv = ./movpn-07-01-script.sh add fd00::200:1000 client1
17:37:18: START tls-verify script ===
argv = ./movpn-07-01-script.sh 1 C=ZA, ST=Enlightenment,
 L=Overall, O=Mastering OpenVPN, CN=Mastering OpenVPN,
 emailAddress=root@example.org
17:37:18: START tls-verify script ===
argv = ./movpn-07-01-script.sh 0 C=ZA, ST=Enlightenment,
 O=Mastering OpenVPN, CN=client1,
 emailAddress=root@example.org
17:37:18: START user-pass-verify script ===
argv = ./movpn-07-01-script.sh
17:37:18: START client-disconnect script ===
argv = ./movpn-07-01-script.sh
17:37:18: START client-connect script ===
argv = ./movpn-07-01-script.sh
 /tmp/openvpn_cc_8528c57f838033a03f38ddb72b57ae30.tmp
17:37:18: START learn-address script ===
argv = ./movpn-07-01-script.sh update 10.200.0.2 client1
17:37:18: START learn-address script ===
argv = ./movpn-07-01-script.sh update fd00::200:1000 client1
17:38:50: START client-disconnect script ===
argv = ./movpn-07-01-script.sh
17:38:50: START learn-address script ===
argv = ./movpn-07-01-script.sh delete fd00::200:1000
17:38:50: START learn-address script ===
argv = ./movpn-07-01-script.sh delete 10.200.0.2
17:39:08: START route-pre-down script ===
argv = ./movpn-07-01-script.sh tun0 1500 1541 10.200.0.1
255.255.255.0 init
17:39:09: START down script ===
argv = ./movpn-07-01-script.sh tun0 1500 1541 10.200.0.1
255.255.255.0 init

```

Сначала вызываются сценарии up и route-up, как мы уже видели.

Когда первый клиент подключается, сценарий **tls-verify** вызывается дважды: сначала для сертификата CA, который использовался для подписи сертификата клиента, а затем для самого сертификата клиента.

Затем выполняется скрипт auth-user-pass-verify . Когда этот сценарий возвращает успех (код выхода 0), клиент аутентифицируется и считается доверенным.

Следующий скрипт - это скрипт **client-connect**,ываемый с временным файлом. Этот сценарий часто используется для установки специальных параметров для конкретного клиента или для регистрации активности клиента в базе данных. Этот сценарий может по-прежнему влиять на IP-адрес, назначенный клиенту, распечатывая параметр для временного файла, например:

```
echo "ifconfig-push 10.200.0.88 255.255.255.0"> $1
```

Последний скрипт, который вызывается при подключении клиента – это **learn-address**. Он вызывается дважды, один раз с адресом IPv4 и другой - с IPv6. Этот сценарий на самом деле

лучше всего подходит для обновления правил брандмауэра, но большинство людей склонны использовать для этого сценарий `client-connect`. Сценарий `learn-address` необходим, особенно при настройке на основе TAP в сочетании с внешним сервером DHCP.

### Заметка

В настройке на основе TAP вторым параметром в сценарии с `learn-address` является MAC-адрес клиентского адаптера TAP. Клиентский IP-адрес VPN доступен в качестве переменной среды.

Из файла журнала видно, что клиент OpenVPN получил триггер `soft-restart` в 17:37:18. Порядок выполнения скриптов может показаться странным, но это можно объяснить:

- Обнаружено новое входящее соединение. Для этого выполняются сценарии `tls-verify` и `auth-user-pass-verify` чтобы определить, является ли он действительным клиентом.
- Как только определено что это действительный клиент и этот конкретный клиент уже подключен, старый экземпляр клиента сначала отключается. Таким образом вызывается сценарий `client-disconnect`.
- Затем вызывается сценарий `client-connect` для нового экземпляра клиента. Обратите внимание, что новый экземпляр клиента может подключаться с нового удаленного IP-адреса. Наконец, скрипт `learn-address` вызывается дважды с действием, установленным для `update`, один раз для адреса IPv4 и один раз для адреса IPv6. Если необходимо изменить какие-либо правила брандмауэра, то это будет лучшим местом для этого.

В 17:38:50 клиент был отключен. Поскольку мы указали `explicit-exit-notify` в конфигурации клиента, сервер немедленно уведомляется об этом и выполняется сценарий `client-disconnect`.

Обратите внимание, что сценарии `learn-address` теперь выполняются с действием, установленным на `delete`.

В 17:39:08 сам процесс сервера OpenVPN останавливается и вызываются сценарии `route-pre-down` и `down`. Обратите внимание, что в этот сценарий передаются те же параметры, что и в сценарий `up`.

### ***Переменные среды, установленные в серверных сценариях***

Теперь, когда мы понимаем порядок, в котором вызываются скрипты, пришло время более внимательно посмотреть на переменные среды, доступные для скриптов.

#### **--up**

Переменные среды, доступные для скрипта `up`, следующие:

```
MASTERING_OPENVPN=server
PWD=/etc/openvpn/movpn
SHLVL=1
=/bin/env
config=movpn-07-01-server.conf
daemon=1
daemon_log_redirect=1
daemon_pid=8070
daemon_start_time=1437659216
dev=tun0
dev_type=tun
ifconfig_broadcast=10.200.0.255
```

```
ifconfig_ipv6_local=fd00::200:1
ifconfig_ipv6_netbits=112
ifconfig_ipv6_remote=fd00::200:2
ifconfig_local=10.200.0.1
ifconfig_netmask=255.255.255.0
link_mtu=1541
local_port_1=1194
proto_1=udp
remote_port_1=1194
route_gateway_1=10.200.0.2
route_gateway_2=10.200.0.2
route_net_gateway=<SERVER-IP>
route_netmask_1=255.255.0.0
route_netmask_2=255.255.255.0
route_network_1=10.100.0.0
route_network_2=192.168.0.0
route_vpn_gateway=10.200.0.2
script_context=init
script_type=up
tun_mtu=1500
verb=1
```

Большинство параметров, передаваемых в сценарий `up`, также представлены в виде переменных среды. Информация о маршрутизации на стороне сервера также уже доступна здесь, но лучше всего иметь дело с этими переменными в скрипте `route-up`.

### --route-up

В сценарии `route-up` доступна та же среда, что и в предыдущем коде, с одним добавлением:

```
script_type=route-up
redirect_gateway=0
```

Эта переменная среды имеет значение 1, если шлюз по умолчанию также необходимо перенаправить. Все операторы маршрута, перечисленные в файле конфигурации сервера, также представлены как переменные среды. Для каждого маршрута доступны `route_network`, `route_netmask` и `route_gateway`. Также обратите внимание, что ключевые слова OpenVPN `net_gateway` и `vpn_gateway` представлены здесь как `route_net_gateway` и `route_vpn_gateway`.

### --tls-verify

Сценарий `tls-verify` вызывается с полным именем сертификата, которое также называется **Distinguished Name (DN)**. Еще больше информации о сертификате доступны как переменные среды:

```
X509_0_C=ZA
X509_0_CN=client1
X509_0_O=Mastering OpenVPN
X509_0_ST=Enlightenment
X509_0_emailAddress=root@example.org
X509_1_C=ZA
X509_1_CN=Mastering OpenVPN
X509_1_L=Overall
X509_1_O=Mastering OpenVPN
X509_1_ST=Enlightenment
X509_1_emailAddress=root@example.org
script_type=tls-verify
tls_digest_0=1b:27:a6:b4:5f:7a:9c:3f:17:fb:ff:33:05:61:3f:2a:56:89:16:d3
```

```
tls_digest_1=e4:f1:43:37:34:51:de:99:7a:dc:e3:6d:f2:4c:5b:84:34:4b:f3:64
tls_id_0=C=ZA, ST=Enlightenment, O=Mastering OpenVPN, CN=client1,
emailAddress=root@example.org
tls_id_1=C=ZA, ST=Enlightenment, L=Overall, O=Mastering OpenVPN,
CN=Mastering OpenVPN, emailAddress=root@example.org
tls_serial_0=2
tls_serial_1=15173527578309581038
tls_serial_hex_0=02
tls_serial_hex_1=d2:93:32:f0:8e:bc:58:ee
untrusted_ip=<CLIENT-IP>
untrusted_port=46171
```

Все это можно использовать для определения того, действительно ли этот конкретный сертификат клиента является доверенным. Обратите внимание, что есть две переменные среды: `untrusted_ip = <CLIENT-IP>` и `untrusted_port = 46171`, обозначающие еще недоверенный адрес клиента.

### --auth-user-pass-verify

Сценарий `auth-user-pass-verify` имеет ту же среду, что и предыдущий сценарий, с добавлением трех новых переменных:

```
common_name=client1
username=movpn
password=secret
```

`common_name` устанавливается после успешного завершения сценария `tls-verify`. `username` и `password` передаются как переменные среды, потому что мы установили `script-security` на 3, и добавили параметр `via-env` к опции `auth-user-pass-verify` в файле конфигурации.

### --client-connect

Наиболее часто используемый скрипт `client-connect` имеет почти ту же среду, но переменная `password` была удалена. Кроме того, поскольку процесс аутентификации теперь завершен, есть две новые переменные – `trusted_ip=<CLIENT-IP>` и `trusted_port=<port>`, с точно такими же значениями, что и у их ненадежных аналогов. Обратите внимание, что ненадежные версии также все еще доступны.

### --learn-address

Сценарий `learn-address` имеет ту же среду, что и скрипт `client-connect`. Для этого требуется команда и несколько необязательных аргументов:

- операция: добавить, обновить или удалить
- адрес: адрес, который изучается или не изучается
- общее имя: общее имя сертификата клиента для связи с адресом

В настройке на основе TAP вторым параметром, передаваемым сценарию, является MAC-адрес клиентского адаптера TAP. IP-адрес VPN, который был назначен клиенту сервером (если настроен таким образом), доступен в переменных среды `ifconfig_pool_remote_ip` и `ifconfig_pool_netmask`, соответственно, как показано здесь:

```
ifconfig_pool_remote_ip=10.200.0.2
ifconfig_pool_netmask=255.255.255.0
```

### --client-disconnect

Сценарий `client-disconnect` имеет ту же самую среду, что и `client-connect`, но он также возвращает некоторую статистику аккаунта:

```
bytes_received=7553
bytes_sent=8105
```

Эта информация в основном интересна для нужд аккаунта.

### --route-pre-down и --down

И, наконец, сценарии `route-pre-down` и `down` вызываются с одинаковыми параметрами в качестве скрипта `up`. Когда вызывается сценарий `route-pre-down`, системные маршруты все еще присутствуют. При вызове сценария `down` системные маршруты будут удалены при условии, что у OpenVPN были для этого полномочия. Переменная среды `signal=sigint` предоставляет информацию о типе сигнала, который вызвал отключение OpenVPN.

### Журнал клиентского скрипта

Журнал клиентских скриптов имеет очень похожую структуру и поток, как и журнал на стороне сервера. Опять же, давайте сначала проверим порядок вызова скриптов:

```
15:47:15: START tls-verify script ===
argv = ./movpn-07-01-script.sh 1 C=ZA, ST=Enlightenment,
 L=Overall, O=Mastering OpenVPN, CN=Mastering OpenVPN,
 emailAddress=root@example.org
15:47:15: START tls-verify script ===
argv = ./movpn-07-01-script.sh 0 C=ZA, ST=Enlightenment,
 O=Mastering OpenVPN, CN=Mastering OpenVPN Server,
 emailAddress=root@example.org
15:47:15: START ipchange script ===
argv = ./movpn-07-01-script.sh [AF_INET]<SERVER-IP> [AF_INET]1194
15:47:17: START up script ===
argv = ./movpn-07-01-script.sh tun0 1500 1541 10.200.0.2
255.255.255.0 init
15:47:17: START route-up script ===
argv = ./movpn-07-01-script.sh
17:37:16: START down script ===
argv = ./movpn-07-01-script.sh tun0 1500 1541 10.200.0.2
255.255.255.0 restart
17:37:18: START tls-verify script ===
argv = ./movpn-07-01-script.sh 1 C=ZA, ST=Enlightenment,
 L=Overall, O=Mastering OpenVPN, CN=Mastering OpenVPN,
 emailAddress=root@example.org
17:37:18: START tls-verify script ===
argv = ./movpn-07-01-script.sh 0 C=ZA, ST=Enlightenment,
 O=Mastering OpenVPN, CN=Mastering OpenVPN Server,
 emailAddress=root@example.org
17:37:18: START ipchange script ===
argv = ./movpn-07-01-script.sh [AF_INET]<SERVER-IP> [AF_INET]1194
17:37:20: START up script ===
argv = ./movpn-07-01-script.sh tun0 1500 1541 10.200.0.2
255.255.255.0 restart
17:38:53: START route-pre-down script ===
argv = ./movpn-07-01-script.sh tun0 1500 1541 10.200.0.2
255.255.255.0 init
17:38:53: START down script ===
```

```
argv = ./movpn-07-01-script.sh tun0 1500 1541 10.200.0.2
255.255.255.0 init
```

Когда клиент впервые подключается - сценарий `tls-verify` вызывается дважды, сначала для сертификата CA, используемый для подписи сертификата сервера, а затем для самого сертификата сервера. Таким образом, клиент может проверить, что он подключается к доверенному серверу.

После этого вызывается малоизвестный скрипт `ipchange`. Этот сценарий еще не знает, какому IP-адресу клиента он будет назначен. Он используется главным образом для настройки параметров брандмауэра на клиенте или для уведомления другого приложения о том, что выполняется настройка VPN-подключения.

Как только клиент проходит проверку подлинности на сервере, блок информации передается с сервера на клиент. Затем этот блок анализируется локально как параметры конфигурации, после чего вызываются сценарии `up` и `route`.

Когда мы отправили сигнал `USR1` клиенту OpenVPN – он заставил OpenVPN выполнить `soft-restart`. Это также видно в журнале выполнения скрипта:

- Сначала вызывается сценарий запуска с последним параметром, установленным на `restart` вместо `init`.
- Затем вызываются скрипты `tls-verify` и `ipchange`, так как нам необходимо повторно пройти аутентификацию на сервере.
- Наконец, сценарий `up` вызывается еще раз для настройки IP-адреса VPN. Здесь последний параметр скрипта также устанавливается на `reboot` вместо `init`.

Обратите внимание, что скрипт `route-up` в этом случае не вызывается. Это связано с тем, что мы включили `persist-tun` в конфигурацию клиента. Поскольку интерфейс TUN/TAP не был закрыт или отключен, вся маршрутизация на стороне клиента остается в силе, и, следовательно, скрипт `route-up` не выполняется.

Когда клиент отключается, сценарии `route-pre-down` и `down` вызываются еще раз, на этот раз с параметром, установленным в `init`.

### **Переменные среды, установленные в клиентских скриптах**

Теперь, когда мы понимаем порядок, в котором вызываются сценарии, снова пришло время взглянуть на переменные среды.

Большинство переменных среды на стороне клиента напоминают переменные на серверной стороне. Некоторые переменные являются зеркальными, например `common_name`, содержащая общее имя сертификата на стороне сервера, что можно найти в средах сценариев `up` и `ipchange`:

```
common_name=Mastering OpenVPN Server
```

Также в среде `up script` присутствует переменная, которая была передана с сервера на клиент:

```
OPENVPN_SPECIAL=hack
```

Клиент OpenVPN получил безопасную переменную `SPECIAL` и создал для нее переменную среды, добавив к ней `OPENVPN_`.

### **Заметка**

Нет способа отправить информацию или переменные обратно с клиента на сервер. В версии 2.4 некоторая системная информация будет отправлена обратно, но это не может быть настроено на клиенте.

Когда клиент OpenVPN перезапускается с использованием сигнала USR1, сценарии down и up вызываются с последним параметром, установленным на reboot вместо init. Это также отражается в переменных среды обоих сценариев:

```
script_context=restart
script_type=down ## or up
signal=sigusr1
```

Когда OpenVPN завершается – эти переменные среды содержат следующее:

```
script_context=init
script_type=down
signal=exit-with-notification
```

## Плагины

Из-за простоты написания сценариев интерфейс плагина OpenVPN является относительно недостаточно используемым инструментом, доступным администраторам серверов OpenVPN. OpenVPN по умолчанию поставляется с парой плагинов, один - для аутентификации PAM, а другой для выполнения сценариев - -down с привилегиями root, независимо от того, снижает ли администратор привилегии.

### down-root

Это хорошая идея, чтобы отбросить привилегии в OpenVPN, и плагин down-root позволяет вам это делать. Такие приложения, как брандмауэры, требуют повышения привилегий для добавления и удаления правил брандмауэра. Используя плагин down-root, администратор может предоставить новые правила брандмауэра при подключении клиента, а также возможность удаления этих правил после отключения клиента.

Сценарий использования может быть одним экземпляром OpenVPN, поддерживающим весь персонал компании. Административному и офисному персоналу, как правило, не нужен доступ к интерфейсам управления отключением и другим подобным системам в сети компании. С добавлением правил брандмауэра OpenVPN может ввести разрешенный доступ для определенных клиентских подключений на основе CN или других переменных среды. Как только этот техник отключается, удаление правил брандмауэра не позволяет другому нетехническому сотруднику получить такой доступ, даже если он находится в той же подсети или получает тот же IP, что и ранее подключенный сотрудник.

### Плагин auth-pam

Второй плагин, с которым поставляется OpenVPN, это плагин auth-pam . Он взаимодействует со стеком подключаемых модулей **аутентификации операционной системы (PAM)**. Используя PAM, администратор может использовать любой бэкэнд, который также может взаимодействовать таким образом. LDAP - это пример использования auth-pam.

Многие системы Unix и Linux имеют возможность аутентификации с помощью LDAP. В случае OpenLDAP существует несколько общих объектов, таких как pam-ldap и nss-ldap от PADL Software. После их настройки и добавления в системный стек PAM OpenVPN можно связать с несколькими простыми параметрами конфигурации.

#### Подсказка

Плагин auth-pam не может быть использован в системах Windows из-за отсутствия поддержки PAM.

В простейшей форме к конфигурации сервера можно добавить следующее:

```
plugin openvpn-auth-pam.so login
```

Это активирует плагин и дает ему команду использовать сервис PAM для входа в систему. Обратите внимание, что это может быть любая служба PAM, включая настройки OpenVPN, определенные администратором. Третий параметр в предыдущем коде идентифицирует услугу. Более сложная конфигурация может включать передаваемые параметры:

```
plugin openvpn-auth-pam.so login login USERNAME password PASSWORD
```

В этом случае мы все еще используем сервис PAM для входа в систему, но для этого требуются два параметра: логин и пароль. OpenVPN потенциально передаст три параметра, заменив ключевые слова PASSWORD, USERNAME и COMMONNAME их очевидными аналогами. USERNAME и PASSWORD требуют чтобы auth-user-pass был установлен в конфигурации клиента.

Чтобы определить запросы, сделанные службой PAM, или отладить сам модуль auth-pam, установите для OpenVPN детальность ведения журнала уровня 7 или выше. Запуск auth-pam вместе с OpenVPN 2.3.6 дает следующий вывод в файле журнала:

```
AUTH-PAM: BACKGROUND: received command code: 0
AUTH-PAM: BACKGROUND: USER: ecrist
AUTH-PAM: BACKGROUND: my_conv[0] query='Login:' style=2
AUTH-PAM: BACKGROUND: name match found, query/match-string
['Login:', 'login'] = 'USERNAME'
AUTH-PAM: BACKGROUND: my_conv[0] query='Password:' style=1
AUTH-PAM: BACKGROUND: name match found, query/match-string
['Password:', 'password'] = 'PASSWORD'
```

Глядя на строки с my\_conv, мы видим два значения запроса Login: и Password:. Те успешно частично совпадают с login и password. Чтобы продемонстрировать частичное совпадение, я изменил строку конфигурации следующим образом:

```
plugin openvpn-auth-pam.so login log USERNAME password PASSWORD
```

В журнале видно, что модуль смог успешно сопоставить log с Login: and password to Password::.

```
AUTH-PAM: BACKGROUND: received command code: 0
AUTH-PAM: BACKGROUND: USER: ecrist
AUTH-PAM: BACKGROUND: my_conv[0] query='Login:' style=2
AUTH-PAM: BACKGROUND: name match found, query/match-string
['Login:', 'log'] = 'USERNAME'
AUTH-PAM: BACKGROUND: my_conv[0] query='Password:' style=1
AUTH-PAM: BACKGROUND: name match found, query/match-string
['Password:', 'password'] = 'PASSWORD'
```

Как бы ни было полезно это слабое соответствие – Вам нужно быть осторожным, поскольку есть вероятность возникновения коллизий в зависимости от используемого вами плагина PAM.

Список проектов, предоставляющих различные подключаемые модули аутентификации, а также различные интерфейсы и менеджеры сертификатов, можно найти по адресу <https://community.openvpn.net/openvpn/wiki/RelatedProjects>.

## Резюме

Скрипты и плагины являются мощными инструментами для расширения OpenVPN. Они позволяют администратору лучше интегрировать OpenVPN в существующую инфраструктуру, например, путем включения аутентификации в отдельной бэкэнд-системе или путем записи статистики использования клиента.

Написание скриптов OpenVPN может быть непростым делом, так как нужно соблюдать особую осторожность в отношении сроков написания скриптов. Текущая версия OpenVPN является монолитной и однопоточной – это означает что длинный или некорректно работающий серверный скрипт может заблокировать весь VPN для всех пользователей.

Также важно понимать последовательность и порядок вызова сценариев. В этой главе мы рассмотрели как работает этот порядок и какие переменные среды присутствуют в каждом из серверных и клиентских сценариев.

В следующей главе мы увидим больше графических пользовательских интерфейсов, поскольку углубимся в использование OpenVPN на смартфонах, планшетах и других мобильных устройствах.

# Глава 8. Использование OpenVPN на мобильных устройствах и домашних маршрутизаторах

В настоящее время OpenVPN доступен не только на традиционных платформах ПК, но и на смартфонах и планшетах под управлением Android или Apple iOS, а также на встроенном оборудовании и домашних маршрутизаторах. Для Android существует два приложения:

**OpenVPN для Android**, являющийся полностью OpenSource, и **OpenVPN Connect** для Android, которое является официальным приложением от OpenVPN Technologies Inc. Мы рассмотрим оба этих приложения для Android, так как есть некоторые тонкие различия в их использовании.

В этой главе мы сначала рассмотрим, как использовать OpenVPN на Android и iOS и как наилучшим образом интегрировать использование смартфона в существующую настройку OpenVPN.

Далее мы рассмотрим, как можно использовать OpenVPN на небольшом оборудовании, например на домашних маршрутизаторах с популярной прошивкой DD-WRT на основе Linux. Мы покажем, как использовать домашний маршрутизатор в качестве клиента OpenVPN и сервера OpenVPN.

Важным примечанием является то, что Android и iOS поддерживают только режим tun. Это ограничение операционной системы, а не используемого приложения OpenVPN. К счастью, большинство развертываний OpenVPN основаны на tun и невозможно построить VPN в режиме Ethernet или подключиться к мостовой сети с помощью iOS или Android.

В этой главе будут рассмотрены следующие темы:

- Использование приложения OpenVPN для Android
- Использование приложения OpenVPN Connect для Android
- Использование приложения OpenVPN Connect для iOS
- Интеграция смартфонов в существующую настройку VPN
- Настройка DD-WRT с поддержкой OpenVPN
- Использование домашнего роутера в качестве VPN-клиента
- Использование домашнего роутера в качестве VPN-сервера

## Использование OpenVPN для приложения Android

Приложение OpenVPN для Android является полностью open source и основано на последней ветке кода OpenVPN (Git-master). Это означает, что в этой версии OpenVPN доступны определенные функции, которые еще не вошли в обычную продакшен-версию OpenVPN.

Для этого примера мы установили OpenVPN для Android из Google Play на планшет Samsung Galaxy Note 10.1 2014 под управлением Android 4.3.

И для OpenVPN для Android и для OpenVPN Connect удобно настроить специальный профиль конфигурации. Этот профиль затем может быть импортирован в приложение OpenVPN одним кликом. Это относится как к версиям приложений для Android, так и для iOS как мы увидим позже в этой главе.

### Подсказка

Обратите внимание, что есть два отдельных приложения, написанных известными разработчиками OpenVPN. OpenVPN Connect - продукт от OpenVPN Technologies Inc.,

написанный Джеймсом Йонаном. OpenVPN для Android написана Арне Швабе.

## Создание профиля приложения OpenVPN

Чтобы создать профиль приложения OpenVPN, мы будем следовать приведенным здесь шагам:

1. Мы начнем с файла конфигурации `basic-udp-client.conf` и заменим все ссылки на внешние файлы (`tls-auth`, `ca`, `cert` и `key`) ключевым словом `[inline]`. Затем мы добавляем встроенные BLOB-объекты для этих файлов, копируя и вставляя содержимое файлов `ta.key`, `ca.crt`, `client1.crt` и `client1.key` соответственно.

### Заметка

Встроенные конфигурации требуются для приложения OpenVPN connect (даже для iOS) из-за необходимости хранить конфигурацию и сертификаты вместе. Эти приложения поддерживают несколько конфигураций OpenVPN, что исключает необходимость уникального именования файлов сертификатов.

2. Результирующий профиль конфигурации будет выглядеть так:

```
client
proto udp
remote openvpnserver.example.com
port 1194
dev tun
nobind
remote-cert-tls server
tls-auth [inline] 1
ca [inline]
cert [inline]
key [inline]

<ca>
-----BEGIN CERTIFICATE-----
MIIEwTCCA6mgAwIBAgIJANKTMyC0v...
...
-----END CERTIFICATE-----
</ca>

<cert>
-----BEGIN CERTIFICATE-----
MIIDeTCCAmECAQQwDQYJKoZIhvcNAQE...
...
-----END CERTIFICATE-----
</cert>

<key>
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA3vzLCSqR3fQF...
...
-----END RSA PRIVATE KEY-----
</key>

<tls-auth>
-----BEGIN OpenVPN Static key V1-----
5f5b2bfff373961654089871b40a39eb
...
-----END OpenVPN Static key V1-----
</tls-auth>
```

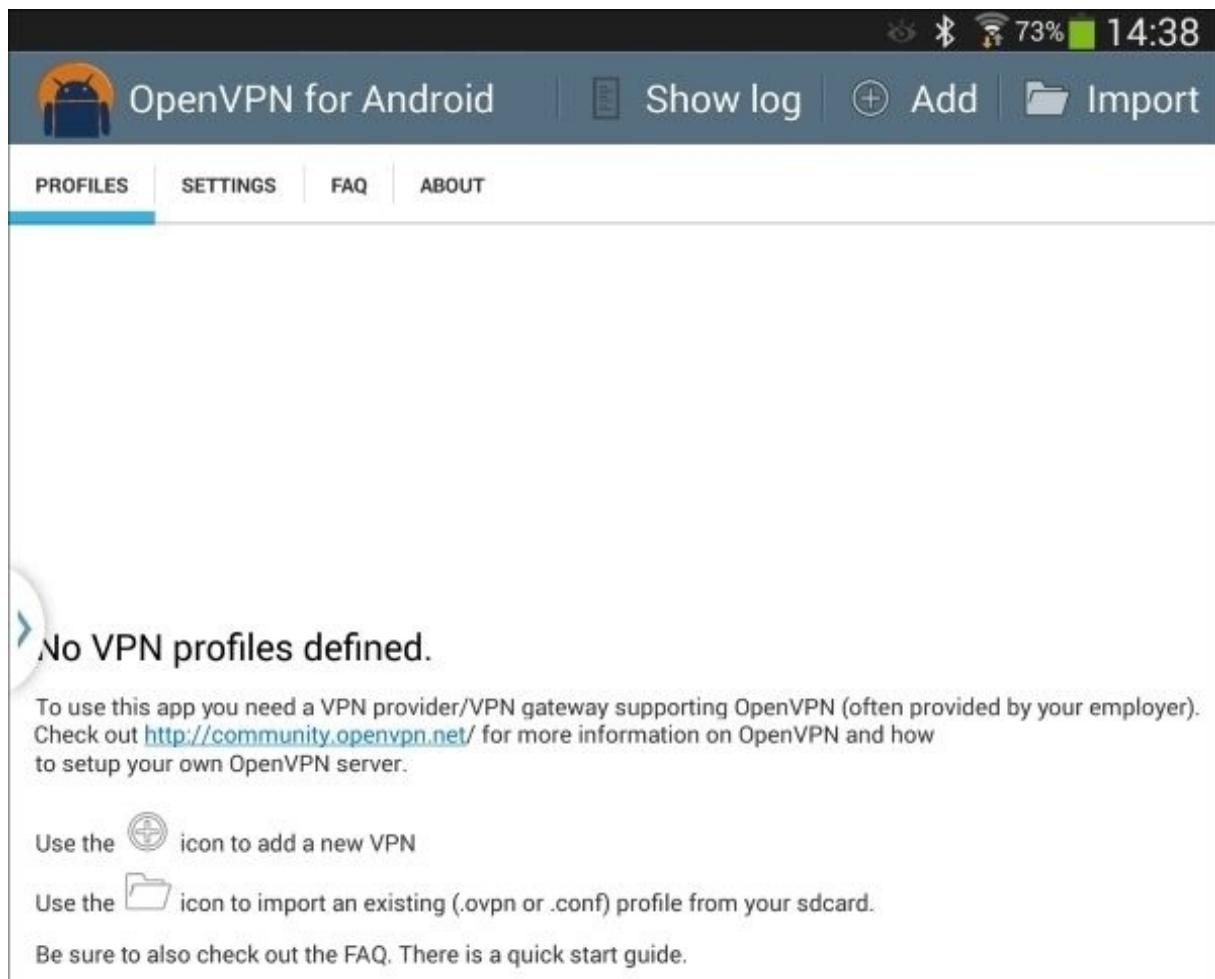
3. Сохраните его как `basic-udp-inline.ovpn`.

4. Сделайте файл доступным на устройстве Android, передав или отправив его по почте.

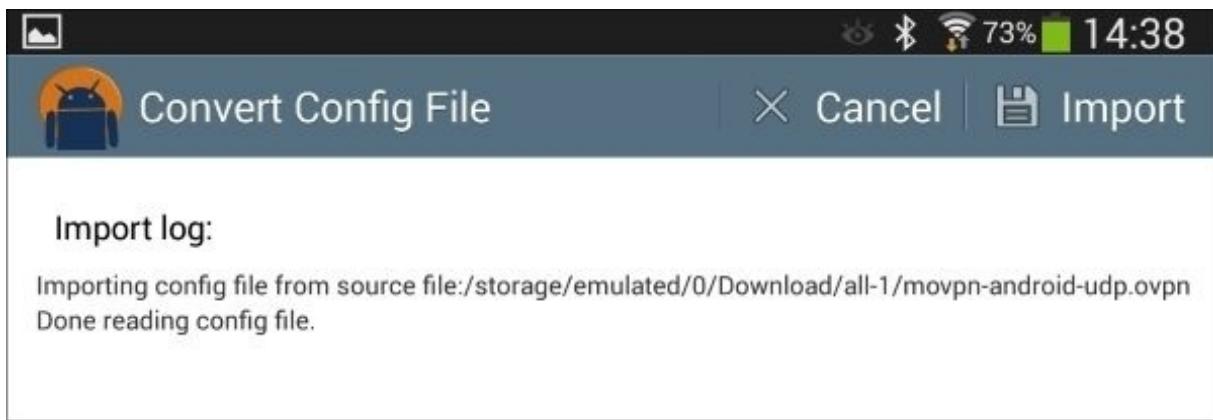
### Заметка

Если вы загружаете файл конфигурации на веб-сервер – очень важно, чтобы тип и расширение файла оставались неизменными. Если планшет или телефон распознает профиль OpenVPN как простой текстовый файл, то он обычно автоматически обрабатывает его как текстовый файл. В некоторых случаях может быть желательно сохранить файл .ovpn внутри ZIP-файла (.zip), чтобы избежать такого искажения типа файла.

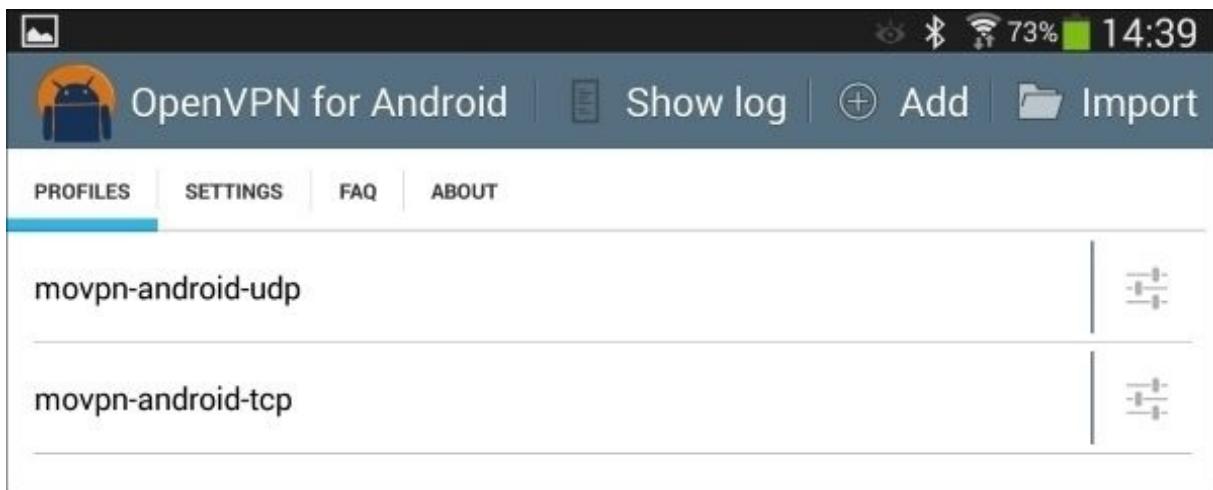
5. На стороне сервера OpenVPN запустите сервер, используя профиль `ipv6-udp-server.conf`.
6. В оставшейся части этого примера мы используем устройство Android.
7. Загрузите и установите бесплатное приложение из Google Play на устройстве.
8. Убедитесь, что файл конфигурации .ovpn также доступен на устройстве.
9. Запустите приложение в самый первый раз. Появится пустой список профилей, как показано здесь:



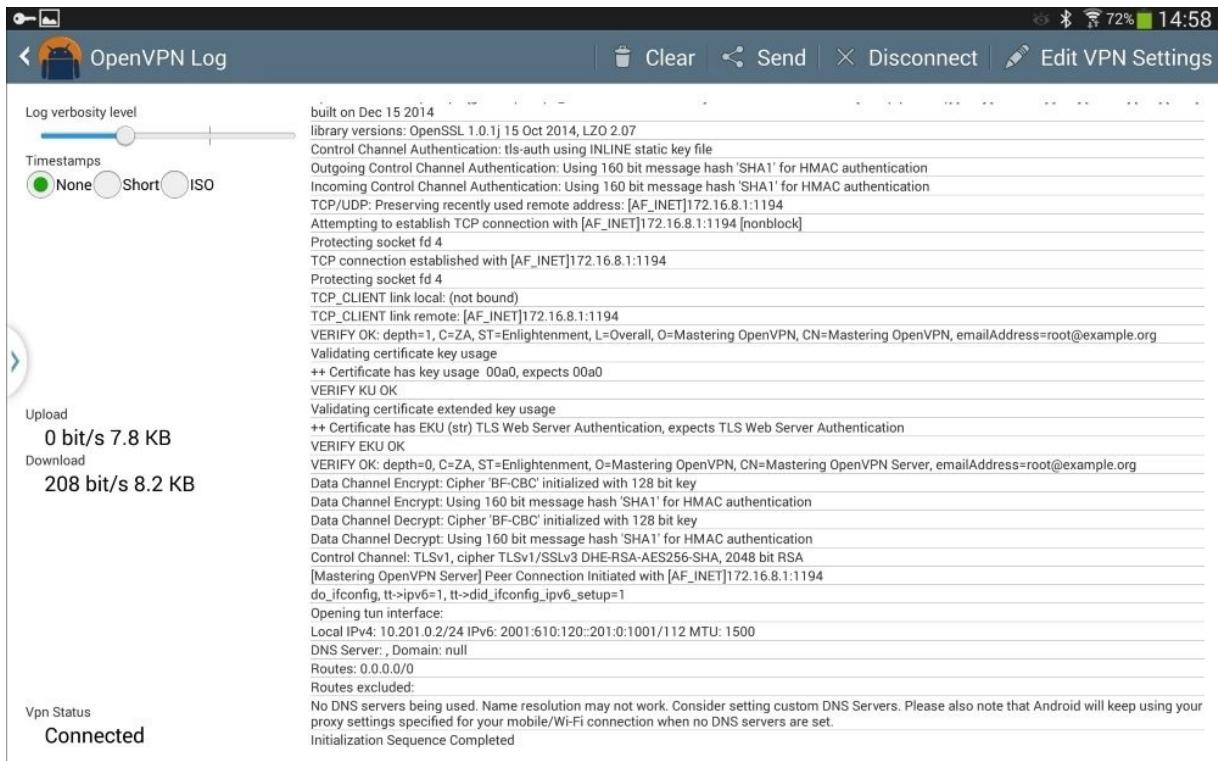
10. Нажмите на значок папки в нижней части экрана, чтобы импортировать файл .ovpn. Найдите расположение файла .ovpn, выберите его и нажмите **Import**. После успешного импорта журнал покажет:



11. Вернитесь к главному экрану приложения OpenVPN. На этом экране вы теперь можете посмотреть список доступных профилей:



12. На устройстве Android один раз нажмите на профиль **movpn-android-udp** чтобы установить соединение OpenVPN. Приложение OpenVPN for Android теперь попытается установить соединение. Если для ведения журнала задана детальность, то на главном экране будет показан журнал OpenVPN, очень похожий на настольный клиент OpenVPN, без отметок времени:



13. OpenVPN подключится. После того, как соединение было успешно установлено, **Vpn Status** в левом нижнем углу будет отображаться как **Connected**.

### Заметка

Из файла журнала обратите внимание, что OpenVPN для Android также поддерживает адресацию IPv6.

14. Далее мы проверяем, что VPN-соединение действительно функционирует.

15. Используя приложение Android Ping, мы проверяем IP-адрес VPN-сервера. Запустите приложение Ping и введите VPN-адрес сервера. Поскольку более новые версии Android больше не поддерживают сообщения проверки связи ICMP, в этом примере используется проверка связи на основе TCP с портом назначения 80. Запустите крошечный веб-сервер на сервере VPN и убедитесь, что входящий трафик TCP через порт 80 разрешен, прежде чем пытаться пропинговать сервер VPN.



16. Конечно, вы также можете использовать обычный ICMP-пинг с сервера на VPN-IP-адрес клиента, чтобы убедиться, что VPN-соединение функционирует:

```
[server]$ ping 10.200.0.2
PING 10.200.0.2 (10.200.0.2) 56(84) bytes of data.
64 bytes from 10.200.0.2: icmp_seq=1 ttl=64 time=14.5 ms
64 bytes from 10.200.0.2: icmp_seq=2 ttl=64 time=13.2 ms
```

### Заметка

Приложение Android для OpenVPN можно использовать на смартфонах и планшетах на основе ядра ARM или Intel Atom. Приложение OpenVPN Connect, о котором пойдет речь в следующем разделе, доступно только для устройств на базе чипа ARM.

## Использование файла PKCS#12

Приложение OpenVPN для Android также может использовать внешнюю пару открытый сертификат/закрытый ключ в так называемом формате PKCS#12.

Вы можете преобразовать существующий файл публичного сертификата (.crt) и приватного ключа (.key) в файл PKCS#12 (.p12), используя следующие команды:

```
$ openssl pkcs12 -export -out client1.p12 \
 -in client1.crt -inkey client1.key -CAfile movpn-ca.crt
Enter Export Password:
Verifying - Enter Export Password:
```

Убедитесь, что вы также включили файл сертификата CA; в противном случае файл са должен быть включен в конфигурацию клиента.

Соответствующий профиль конфигурации клиента представлен следующим образом:

```
client
proto udp
remote openvpnserver.example.com
port 1194
dev tun
nobind
remote-cert-tls server
tls-auth [inline] 1
pkcs12 client1.p12
<tls-auth>
-----BEGIN OpenVPN Static key V1-----
5f5b2bffff373961654089871b40a39eb
...
-----END OpenVPN Static key V1-----
</tls-auth>
```

Преимущество этого метода заключается в том, что конфигурация OpenVPN хранится отдельно от файлов аутентификации клиента. Недостатком является то, что этот метод не работает с приложением OpenVPN Connect.

## Использование приложения OpenVPN Connect для Android

Приложение OpenVPN Connect является официальным приложением от OpenVPN Technologies Inc. Я скачал и установил бесплатное приложение из Google Play на тот же планшет Samsung Galaxy Note 10.1 2014, что и в предыдущем примере.

Приложение OpenVPN Connect можно использовать только с профилями, которые используют [inline] сертификат и пары ключей. Для этого мы используем профиль конфигурации OpenVPN, созданный в предыдущем примере, используя следующие шаги:

1. На стороне VPN-сервера мы запускаем OpenVPN, используя стандартный файл конфигурации `ipv6-udp-server.conf`.
2. После загрузки и установки запустите приложение и импортируйте профиль:



3. Далее выберите нужный профиль и нажмите **Connect**:



4. После того, как соединение OpenVPN установлено, клиент сообщит **OpenVPN: Connected**.

#### Заметка

Обратите внимание, что клиент OpenVPN Connect для Android также поддерживает адресацию IPv6, как видно на предыдущем снимке экрана.

5. Проверка того, что VPN-клиент может быть доступен с сервера и наоборот, остается в качестве упражнения для читателя.

## Использование приложения OpenVPN Connect для iOS

В этом примере я установил приложение OpenVPN Connect из магазина приложений Apple на Apple iPad под управлением iOS 8.1.2, а также на iPhone под управлением iOS 8.

#### Заметка

Эта версия OpenVPN не является open source. Специальное соглашение с Apple было необходимо, чтобы получить представление о сетевом стеке Apple iOS, для возможности портировать OpenVPN на iOS. Это устраняет необходимость во взломанном устройстве.

Как и в версии Android, приложение OpenVPN Connect лучше всего использовать с профилями, использующими [inline] сертификат и пары ключей. Поэтому мы снова используем профиль конфигурации OpenVPN, созданный в первом примере этой главы.

На стороне VPN-сервера мы запускаем OpenVPN, используя стандартный файл конфигурации `ipv6-udp-server.conf`.

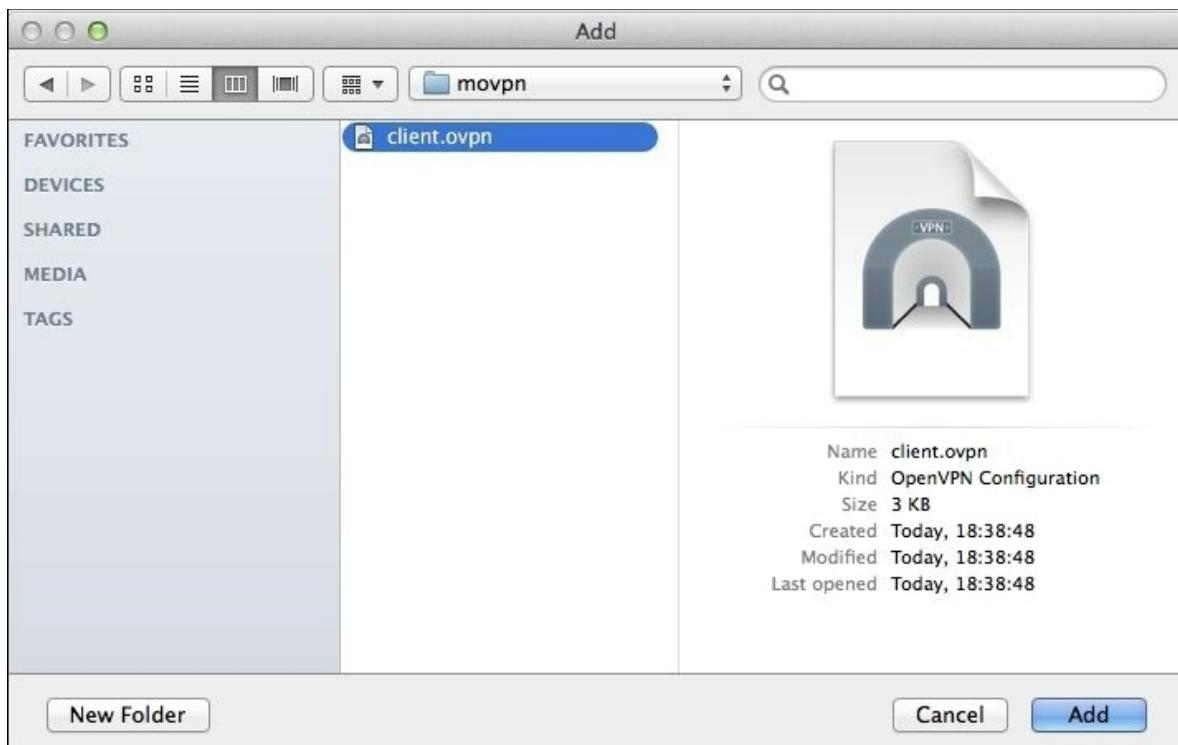
Прежде чем мы сможем использовать профиль OpenVPN на iOS, необходимо перенести его на устройство. Это можно сделать по электронной почте или с помощью iTunes. Это хорошая идея, чтобы убедиться в использовании безопасного метода передачи. Соединение iTunes является безопасным, но также можно использовать электронную почту с шифрованием TLS или другой способ передачи, например Dropbox или Google Drive.

Вот пример, приведенный для OpenVPN в следующих шагах:

1. После передачи профиля `.ovpn` запустите приложение OpenVPN и прокрутите вниз до раздела **File Sharing**. Нажмите на **OpenVPN** и увидите следующий экран:



2. Нажмите кнопку **Add...** чтобы добавить новый профиль. Вы также можете использовать перетаскивание конфигурации OpenVPN (файл в режиме inline обычно с расширением `.ovpn`) из папки или рабочего стола в это окно.
3. Выберите импортированный профиль и снова нажмите **Add**:



4. Файл `client.ovpn` теперь доступен в качестве профиля OpenVPN Connect:



5. Когда мы запустим приложение OpenVPN Connect на чистом iPhone или iPad, мы увидим следующий интерфейс:

[About](#) [OpenVPN](#) [Help](#)

WELCOME TO OPENVPN

OpenVPN requires a profile (.ovpn file) to connect to a server. Please use one of the following apps to import a profile:

 Import your Private Tunnel profile. [Go](#)

 If you are importing a profile from an OpenVPN Access Server, log into the server using Safari and click on "user-locked" or "autologin" profile.  
 [Go](#)

 Using iTunes Sync, select your device, go to OpenVPN under the "apps" tab, and drop your .ovpn and related cert/key files into the file sharing window.

 If you receive the profile as a .ovpn attachment in the Mail app, you can open it in OpenVPN (Note: this method is less secure).

 [More Help...](#) >

MORE FROM OPENVPN TECHNOLOGIES...

 [Private Tunnel -- Your Secure and Private Path to the Internet](#) >

 [OpenVPN Access Server -- VPN Solution for your Business](#) >

OpenVPN is a registered trademark of OpenVPN Technologies, Inc.

- После того, как профиль был импортирован, на экране появляется дополнительный раздел, в котором перечислены все доступные профили. Поначалу интерфейс OpenVPN Connect может быть немного сложнее. Например, когда импортируется неверная конфигурация, отображается следующий экран:

NEW PROFILES ARE AVAILABLE...

 1 new OpenVPN profile is available for import. 

 Error loading profile: client.ovpn  
option\_error: remote option not specified

- Приложение сообщает, что профиль доступен для импорта, но мы можем видеть предупреждающие сообщения о том, что при загрузке этого профиля произошла ошибка. Единственный доступный вариант - удалить профиль, нажав на красную метку X. Когда импортируется правильный файл конфигурации, также доступна зеленая кнопка +. Это показано на следующем снимке экрана:

NEW PROFILES ARE AVAILABLE...



1 new OpenVPN profile is available for import.



192.168.5.7/autologin  
Autologin profile



8. Нажмите на значок +, чтобы открыть профиль подключения OpenVPN:

**OpenVPN Connect**

<b>Profile</b>	192.168.5.7/autologin Autologin profile	>
<b>Status</b>	Disconnected	>
<b>Connection</b>	<input type="checkbox"/>	

9. Наконец, используйте ползунок с записью **Connection**, чтобы запустить соединение OpenVPN.

10. После того, как VPN-соединение было успешно установлено, мы видим что в приложении OpenVPN Connect поддерживаются как IPv4, так и IPv6:

**OpenVPN Connect**

<b>Profile</b>	192.168.5.7/autologin Autologin profile	>
<b>Status</b>	Connected	>
<b>Connection</b>	<input checked="" type="checkbox"/>	

**CONNECTION DETAILS**

<b>Duration</b>	0:00:02	<b>Last packet received</b>	2 seconds ago
<b>Bytes In</b>	3.40 KB	<b>Bytes Out</b>	2.56 KB
<b>VPN IPv6</b>	2001:db8:0:dead:beef::1000	<b>VPN IPv4</b>	172.17.0.2
<b>User</b>	<b>Client IP</b>		
<b>Server</b>	192.168.5.7	<b>Server IP</b>	192.168.5.7
<b>Port</b>	1194	<b>Protocol</b>	UDPv4

11. Вы можете снова использовать ползунок **Connection** для остановки VPN-соединения.

## Интеграция смартфонов в существующую настройку VPN

OpenVPN на смартфонах можно использовать только в качестве VPN-клиента, что в любом случае является обычным режимом использования смартфона. Для Android существует несколько клиентских приложений OpenVPN. Между различными приложениями есть некоторые тонкие различия, но все они поддерживают только настройки режима tun, так как базовая ОС не поддерживает подключенные устройства.

Вопрос о том, какое приложение использовать на устройствах Android, является сложным. Если вы используете сочетание устройств iOS и Android, то приложение OpenVPN Connect является более простым выбором, поскольку пользовательский интерфейс более согласован для всех устройств. Если вы используете коммерческий сервер доступа OpenVPN, OpenVPN Connect является единственным маршрутом из-за динамической конфигурации и некоторых опций сервера. Если вам нужны некоторые специальные функции приложения OpenVPN для Android или если вы хотите использовать OpenVPN на телефонах или планшетах не на основе ARM, тогда OpenVPN для Android - логичный выбор.

Как видно из предыдущего примера, для использования версий приложения OpenVPN требуются некоторые изменения. Обратите внимание, что мы не вносим никаких изменений на стороне сервера для поддержки устройств Android или iOS, но настройка сервера была относительно простой. Как мы видели в предыдущих примерах, версии «приложений» поддерживают как IPv4, так и IPv6, а также большинство других функций программного обеспечения OpenVPN для настольных ПК. Однако, особенно когда речь идет о маршрутизации или совместном использовании файлов, может быть сложно создать единую настройку на стороне сервера для поддержки всех платформ. Также невозможно запустить клиентские сценарии, если они потребуются для настройки VPN-подключения. Конечно, можно использовать файлы ключей, защищенные паролем.

Если для установленных версий «приложения» необходимо сделать исключения, рекомендуется настроить отдельный сервер OpenVPN для обслуживания. Поэтому настройте новый статический IP-адрес и подключитесь.

В интерфейсе DD-WRT снова обновите прошивку, выбрав «большую» версию и еще раз нажмите **Upgrade**. Процесс обновления снова займет несколько минут, но после этого интерфейс DD-WRT должен снова стать доступным.

Теперь ваш маршрутизатор готов к настройке в качестве клиента или сервера OpenVPN.

## Использование домашнего маршрутизатора в качестве VPN-клиента

Вы можете использовать следующую процедуру для настройки маршрутизатора DD-WRT как OpenVPN клиента:

1. В веб-интерфейсе DD-WRT перейдите на вкладку **Services** и нажмите **VPN**.
2. Нажмите кнопку **Enable** рядом с **Start OpenVPN Client**.
3. Введите сведения о соединении и включите дополнительные параметры как показано на следующем снимке экрана:

**OpenVPN Client**

OpenVPN Client

Start OpenVPN Client  Enable  Disable

Server IP/Name

Port  (Default: 1194)

Tunnel Device

Tunnel Protocol

Encryption Cipher

Hash Algorithm

nsCertType verification

Advanced Options  Enable  Disable

TLS Cipher

LZO Compression

NAT  Enable  Disable

Firewall Protection  Enable  Disable

IP Address

Subnet Mask

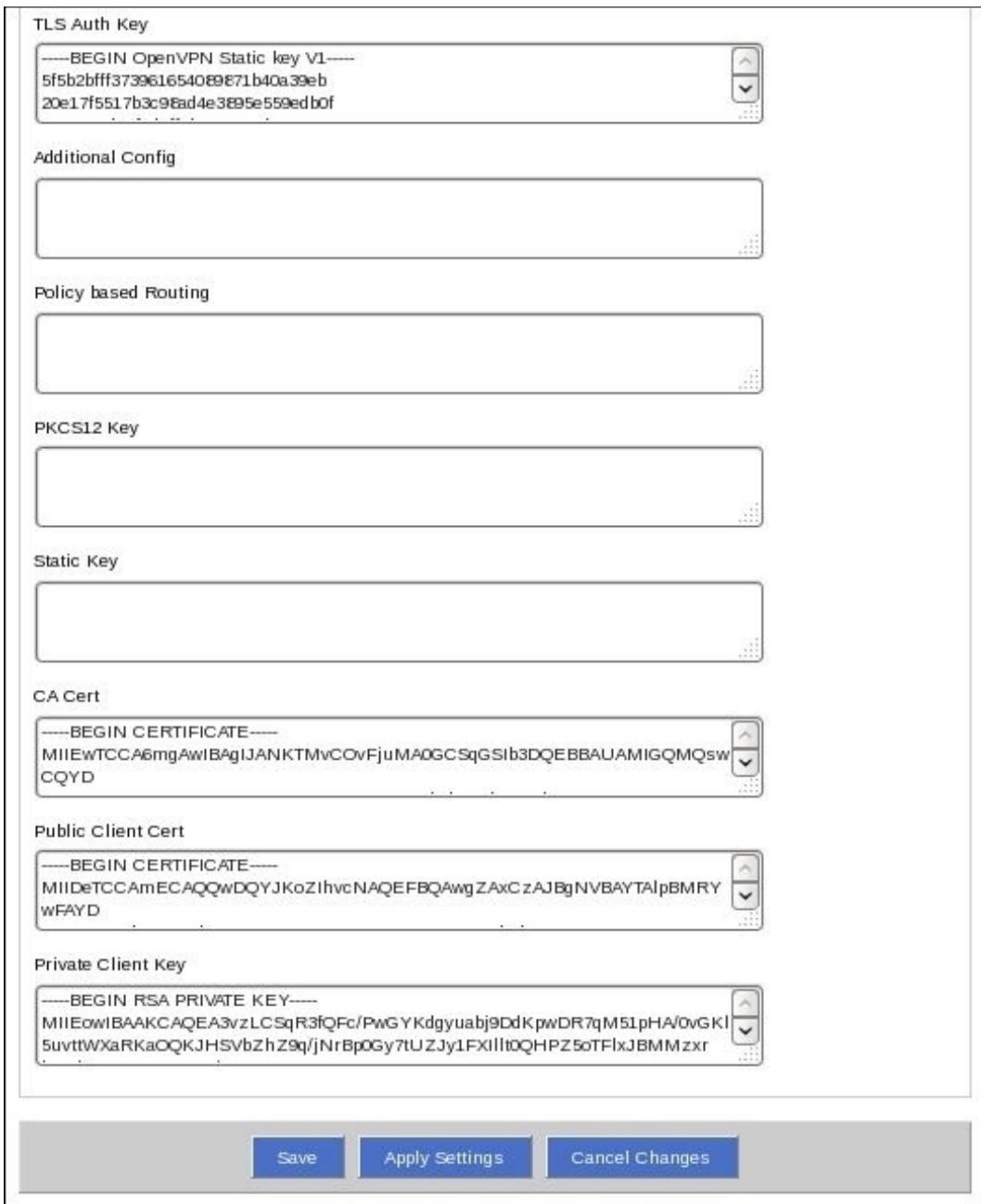
Tunnel MTU setting  (Default: 1500)

Tunnel UDP Fragment

Tunnel UDP MSS-Fix  Enable  Disable

Для большинства настроек можно оставить значения по умолчанию, но отключите **Firewall Protection**, чтобы VPN-сервер мог связаться с клиентом и наоборот, используя следующие шаги:

1. Это длинная веб-форма, поэтому прокрутите вниз и заполните параметры безопасности:
  - Ключ авторизации TLS
  - Сертификат CA
  - Публичный сертификат клиента
  - Приватный ключ клиента
2. Значения этих полей - те же самые значения, которые использовались в профиле конфигурации для клиента Android в начале этой главы:



3. После заполнения всех параметров безопасности нажмите **Save** чтобы сохранить конфигурацию клиента OpenVPN. Обычно, как только введен действительный профиль, клиент OpenVPN DD-WRT пытается установить соединение с сервером. Обратите внимание, что обычно на устройстве DD-WRT может храниться только один профиль из-за ограничений размера памяти устройства NVRAM.
4. После того, как VPN-соединение установлено – мы можем проверить что VPN-клиент может быть доступен с сервера, пропингуя его VPN-IP-адрес:

```
$ ping -c 2 10.200.0.2
PING 10.200.0.2 (10.200.0.2) 56(84) bytes of data.
64 bytes from 10.200.0.2: icmp_seq=1 ttl=64 time=0.591 ms
64 bytes from 10.200.0.2: icmp_seq=2 ttl=64 time=0.659 ms
--- 10.200.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.591/0.625/0.659/0.034 ms
```

Самый простой способ устранения любых проблем с подключением – это просмотреть журналы на стороне сервера; журнал на стороне клиента может быть получен с помощью веб-интерфейса

DD-WRT, но, как мы увидим в следующей главе, журнал на стороне сервера обычно более информативен.

## Использование домашнего маршрутизатора в качестве VPN-сервера

Использование OpenVPN на небольшом беспроводном маршрутизаторе возможно, но это в значительной степени зависит от точного типа используемого беспроводного маршрутизатора. На сайтах DD-WRT и OpenWRT перечислены многие поддерживаемые беспроводные маршрутизаторы, но даже большинство из них имеют недостатки. Требуется минимальный размер флэш-памяти 8 МБ, а также достаточно большое пространство NVRAM.

Однако даже при наличии правильного оборудования производительность OpenVPN на беспроводном маршрутизаторе будет не очень хорошей из-за ограниченной вычислительной мощности таких устройств. Для людей, которые хотят настроить VPN на свой домашний адрес, производительность обычно хорошая, если только ваше домашнее соединение не способно отдавать более 100 Мбит/с в исходящем потоке.

Следующая процедура может использоваться для настройки маршрутизатора DD-WRT как OpenVPN сервера:

1. В веб-интерфейсе DD-WRT перейдите на вкладку **Services** и нажмите **VPN**.
2. Сначала отключите клиент OpenVPN, прокрутив вниз и выбрав **Disable** в разделе **OpenVPN Client**.
3. В разделе **OpenVPN Server/Daemon** включите **OpenVPN**.
4. Введите сведения о соединении и сравните данные со строками, найденными в файле конфигурации `basic-udp-server.conf`:
  1. Выберите **System** как **Start Type**, чтобы демон OpenVPN запускался при загрузке маршрутизатора DD-WRT.
  2. Не совсем ясно, в чем заключается точное различие между **Configure as server** и **Configure as deamon**.
  3. Выберите **Router (TUN)** в качестве **Server Mode**, так как нам нужна настройка OpenVPN в режиме tun.
  4. Заполните **10.200.0.0** и **255.255.255.0** в **Network** и **Netmask**. Это эквивалент строки "server".
  5. **Port**, **Tunnel Protocol**, **Encryption Cipher** и **Hash Algorithm** можно оставить с их значениями по умолчанию.
  6. Нажмите **Enable** рядом с **Advanced Options**, чтобы отобразить все доступные параметры конфигурации OpenVPN. Нам не нужно изменять какие-либо из этих параметров, но полезно посмотреть, какие из них доступны:

**OpenVPN Server/Daemon**

OpenVPN	<input checked="" type="radio"/> Enable <input type="radio"/> Disable
Start Type	<input type="radio"/> WAN Up <input checked="" type="radio"/> System
Config as	<input checked="" type="radio"/> Server <input type="radio"/> Daemon
Server mode	<input checked="" type="radio"/> Router (TUN) <input type="radio"/> Bridge (TAP)
Network	10.200.0.0
Netmask	255.255.255.0
Port	1194 <small>(Default: 1194)</small>
Tunnel Protocol	UDP <small>(Default: UDP)</small>
Encryption Cipher	Blowfish CBC
Hash Algorithm	SHA1
Advanced Options	<input checked="" type="radio"/> Enable <input type="radio"/> Disable
TLS Cipher	None
LZO Compression	Disabled
Redirect default Gateway	<input type="radio"/> Enable <input checked="" type="radio"/> Disable
Allow Client to Client	<input checked="" type="radio"/> Enable <input type="radio"/> Disable
Allow duplicate cn	<input type="radio"/> Enable <input checked="" type="radio"/> Disable
Tunnel MTU setting	1500 <small>(Default: 1500)</small>
Tunnel UDP Fragment	<small>(Default: Disable)</small>
Tunnel UDP MSS-Fix	<input type="radio"/> Enable <input checked="" type="radio"/> Disable

5. Далее мы заполняем сертификат и детали открытого/приватного ключа. Для этого мы используем файлы из стандартного файла конфигурации сервера basic-udp-server.conf.
6. Прокрутите вниз в длинной веб-форме и вставьте содержимое файлов server.crt, ca.crt, server.key и dh2048.pem соответственно:
  - **Public Server Cert:** соответствует строке «cert» (server.crt)
  - **CA Cert:** соответствует строке "ca" (movpn-ca.crt)
  - **Private Server Key:** соответствует строке "key" (server.key)
  - **DH PEM:** соответствует строке "dh" (dh2048.pem)

Эти термины показаны на следующем снимке экрана:

Public Server Cert	<pre>-----BEGIN CERTIFICATE----- MIIFADCCA+igAwIBAgIBATANBgkqhkiG9w0BAQUFADCBlDELMakGA1UEBhMCW kEx</pre>
CA Cert	<pre>-----BEGIN CERTIFICATE----- MIIETCCA6mgAwIBAgIJANKTMvC0vFjuMA0GCSqGSIb3DQEBAUAMIGQMQsw CQYD</pre>
Private Server Key	<pre>-----BEGIN RSA PRIVATE KEY----- MIIEcwIBAAKCAQEAzm0ambZopw+RL0Mm1g00lsuV05nyDQhXU9taE621HucmL 9GV</pre>
DH PEM	<pre>-----BEGIN DH PARAMETERS----- MIIBCAKCAQEA4ZvBl/m5SsITsqQaPwZYvEj0apa/4aI1+uRrKzxAhYz6IWAT55p DK39rKBHsx87rnP4mFVZn2u8ar5roViK75wKRtqUJueAdgLRmRGelAyIDvWSSxk</pre>
Additional Config	<input type="text"/>
TLS Auth Key	<input type="text"/>
Certificate Revoke List	<input type="text"/>

7. Обратите внимание, что мы пропустили поле **TLS Auth Key**. Если это поле также заполнено, то NVRAM устройства DD-WRT исчерпана и маршрутизатор необходимо будет перезагрузить. В результате мы не сможем использовать файл конфигурации клиента `basic-udp-client.ovpn` для подключения к этому серверу.
8. Прокрутите страницу вниз и нажмите кнопку **Save**. Как только конфигурация будет сохранена в NVRAM, процесс сервера OpenVPN будет запущен.

### Заметка

Хранить все конфигурации в NVRAM сложно. Использование файлов может быть альтернативным выбором, но для этого требуется собственный скрипт запуска.

Если конфигурация не вписывается в NVRAM, скорее всего, это приведет к сбою устройства DD-WRT и вам потребуется вручную сбросить его. Маршрутизатор Belkin Playmax является одним из примеров: всю конфигурацию можно было сохранить только в NVRAM, пропустив файл ключа `tls-auth`.

9. Наконец, подключите клиента с помощью файла конфигурации `tnovpn-04-01-client.conf` и убедитесь, что VPN-соединение работает:

```
[client]$ ping -c 4 10.200.0.1
PING 10.200.0.1 (10.200.0.1) 56(84) bytes of data.
64 bytes from 10.200.0.1: icmp_seq=1 ttl=64 time=22.3 ms
64 bytes from 10.200.0.1: icmp_seq=2 ttl=64 time=18.6 ms
64 bytes from 10.200.0.1: icmp_seq=4 ttl=64 time=21.9 ms
64 bytes from 10.200.0.1: icmp_seq=5 ttl=64 time=15.7 ms
```

## Резюме

OpenVPN теперь доступен на многих платформах, включая смартфоны, планшеты и даже некоторые модели (беспроводных) маршрутизаторов. Методы настройки, поддержки и развертывания на разных устройствах различаются и эти различия следует учитывать при выборе поддерживаемых платформ в вашей среде.

В следующей главе мы сосредоточимся на устранении неполадок конфигурации и производительности OpenVPN. Поскольку конфигурация и производительность OpenVPN на смартфонах, а также на беспроводных маршрутизаторах могут быть громоздкими, будет очень полезно узнать о методах устранения неполадок.

# Глава 9. Устранение неполадок и настройка

Обычно довольно легко настроить VPN при использовании OpenVPN. Это одна из самых привлекательных функций OpenVPN по сравнению с другими решениями VPN. Однако иногда необходимо устранить неполадки в нерабочей настройке или настроить существующую настройку для повышения производительности.

Устранение неполадок и настройка OpenVPN часто игнорируются. Файлы журнала OpenVPN на стороне клиента и сервера предоставляют много информации, но вы должны знать как их читать. При настройке файлов конфигурации клиента и сервера также допускается довольно много ошибок. В этой главе вы узнаете как интерпретировать файлы журнала OpenVPN и как обнаруживать и исправлять некоторые из этих распространенных ошибок.

Наконец, существует большая разница между рабочей установкой и установкой, работающей хорошо. Возможно, ваша установка OpenVPN работает правильно, но пользователи все равно могут жаловаться на низкую производительность. Получение максимальной производительности от установки OpenVPN может показаться черной магией. В этой главе вы узнаете немного этой черной магии.

В этой главе будут рассмотрены следующие темы:

- Как читать файлы журнала
- Исправление распространенных ошибок конфигурации
- Устранение проблем с маршрутизацией
- Как оптимизировать производительность с помощью `ping` и `iperf`
- Анализ трафика OpenVPN с использованием `tcpdump`

## Как читать файлы журнала

Поначалу отладка нерабочих настроек может показаться сложной задачей. С чего начать? К счастью, OpenVPN предоставляет отличные средства ведения журналов и отладки. Однако, с увеличением степени детализации журналов становится все труднее читать эти журналы. Уровень детализации журнала OpenVPN по умолчанию равен 1, но рекомендуется установить степень детализации 3. Это часто дает администратору достаточно информации для обнаружения проблем с настройкой, в то же время сводя к минимуму потери производительности.

Установка детальности на 5 или выше рекомендуется только для целей отладки, так как это сильно влияет на производительность.

Каждый пример в этой книге до сих пор включал установку `verb 3`. Во-первых, мы рассмотрим файлы журнала клиента и сервера для рабочей настройки с такой детальностью. Важно понимать и, возможно, даже хранить файлы журналов работающего соединения. При попытке найти ошибку в нерабочей настройке, очень полезно сравнивать файлы журнала нерабочего случая с файлами рабочей настройки.

Запустите сервер, используя файл конфигурации по умолчанию `basic-udp-server.conf`:

```
[root@server]# openvpn --config basic-udp-server.conf
```

Пока не подключайтесь к клиенту. Файл журнала сервера теперь будет содержать следующее:

```
1 14:53:27 OpenVPN 2.3.6 x86_64-redhat-linux-gnu
[SSL (OpenSSL)] [LZO] [EPOLL] [PKCS11] [MH] [IPv6]
built on Dec 2 2014
2 14:53:27 library versions: OpenSSL 1.0.1e-fips 11 Feb 2013,
LZO2.03
3 14:53:27 Diffie-Hellman initialized with 2048 bit key
4 14:53:31 WARNING: this configuration may cache passwords in
memory -- use the auth-nocache option to prevent this
5 14:53:31 Control Channel Authentication: using
'/etc/openvpn/movpn/ta.key' as a OpenVPN static key
file
6 14:53:31 Outgoing Control Channel Authentication: Using 160
bit message hash 'SHA1' for HMAC authentication
7 14:53:31 Incoming Control Channel Authentication: Using 160
bit message hash 'SHA1' for HMAC authentication
8 14:53:31 Socket Buffers: R=[16777216->131072]
S=[16777216->131072]
9 14:53:31 TUN/TAP device tun0 opened
10 14:53:31 TUN/TAP TX queue length set to 100
11 14:53:31 do_ifconfig, tt->ipv6=0, tt-did_ifconfig_ipv6_setup=0
12 14:53:31 /sbin/ip link set dev tun0 up mtu 1500
13 14:53:31 /sbin/ip addr add dev tun0 10.200.0.1/24
broadcast 10.200.0.255
14 14:53:31 GID set to nobody
15 14:53:31 UID set to nobody
16 14:53:31 UDPv4 link local (bound): [undef]
17 14:53:31 UDPv4 link remote: [undef]
18 14:53:31 MULTI: multi_init called, r=256 v=256
19 14:53:31 IFCONFIG POOL: base=10.200.0.2 size=252, ipv6=0
20 14:53:31 Initialization Sequence Completed
```

Метки времени в начале каждой строки были сокращены для ясности.

Давайте посмотрим на этот файл журнала построчно:

- Строки 1 и 2 указывают версию и дату сборки самого OpenVPN, а также библиотеки, от которых зависит OpenVPN.
- Строка 3 говорит нам, что параметры Диффи-Хеллмана сервера были успешно инициализированы. Файл, указанный в строке конфигурации сервера dh /etc/openvpn/movpn/dh2048.pem был использован для этого.
- Строка 4 - это предупреждение, которое печатается почти всегда. Разработчики обсуждали, следует ли удалить эту строку или нет. В конце концов было решено, что по соображениям безопасности лучше всего распечатать это предупреждение. Если вы не слишком озабочены безопасностью, то можете игнорировать эту строку предупреждения.
- Строка 5 указывает, что канал управления защищен с использованием параметра конфигурации tls-auth и что OpenVPN смог успешно прочитать указанный файл.
- Строки 6 и 7 сообщают нам, что два ключа SHA1 получены из файла tls-auth и теперь используются для подписи (хэширования) исходящего трафика и для проверки входящего трафика.
- Строка 8 показывает размер буферов Receive (R) и Send (S), которые использует OpenVPN. Эти параметры полезны только при доработке рабочей настройки, как мы увидим позже в этой главе.
- Строки 9 и 10 показывают что OpenVPN смог успешно открыть устройство tun и установить глубину очереди пакетов для этого устройства равной 100.

- Строки с 11 по 13 показывают настройки IPv4, используемые для этой конфигурации сервера. Они также указывают что не были заданы параметры IPv6. Перечисленные здесь настройки являются переводом строки конфигурации сервера server 10.200.0.0 255.255.255.0.
- Строки 14 и 15 являются результатом указания group nobody и user nobody в файле конфигурации сервера соответственно.
- Строки 16 и 17 показывают что OpenVPN прослушивает трафик UDP и привязан к неопределенному (undefined) интерфейсу (0.0.0.0). Это результат указания proto udp и bind в файле конфигурации сервера.
- Строка 18 говорит нам, что это мультиклиентская установка с реальными и виртуальными размерами таблицы хешей 256.
- В строке 19 указан диапазон адресов пула, доступных клиентам OpenVPN, которые могут подключаться к этому серверу. Это также часть перевода строки конфигурации сервера server 10.200.0.0 255.255.255.0.
- Строка 20 - это волшебная строка, сообщающая нам, что сервер успешно запущен и инициализация завершена. Сервер теперь готов принимать соединения от входящих клиентов.

Далее мы запускаем клиент и смотрим файл журнала на стороне сервера:

```
[root@client]# openvpn --config basic-udp-client.conf
```

После этого мы также рассмотрим файл журнала на стороне клиента:

```
21 15:30:37 <CLIENT-IP>:39086 TLS: Initial packet from
[AF_INET]<CLIENT-IP>:39086, sid=071ba589 7e9ff2a0
22 15:30:37 <CLIENT-IP>:39086 VERIFY OK: depth=1, C=ZA,
ST=Enlightenment, L=Overall, O=Mastering OpenVPN,
CN=Mastering OpenVPN, emailAddress=root@example.org
23 15:30:37 <CLIENT-IP>:39086 VERIFY OK: depth=0, C=ZA,
ST=Enlightenment, O=Mastering OpenVPN, CN=client3,
emailAddress=root@example.org
24 15:30:37 <CLIENT-IP>:39086 Data Channel Encrypt: Cipher
'BF-CBC' initialized with 128 bit key
25 15:30:37 <CLIENT-IP>:39086 Data Channel Encrypt: Using 160 bit
message hash 'SHA1' for HMAC authentication
26 15:30:37 <CLIENT-IP>:39086 Data Channel Decrypt: Cipher
'BF-CBC' initialized with 128 bit key
27 15:30:37 <CLIENT-IP>:39086 Data Channel Decrypt: Using 160 bit
message hash 'SHA1' for HMAC authentication
28 15:30:37 <CLIENT-IP>:39086 Control Channel: TLSv1, cipher
TLSv1/SSLv3 DHE-RSA-AES256-SHA, 2048 bit RSA
29 15:30:37 <CLIENT-IP>:39086 [client3] Peer Connection Initiated
with [AF_INET]<CLIENT-IP>:39086
30 15:30:37 client3/<CLIENT-IP>:39086 MULTI_sva: pool returned
IPv4=10.200.0.2, IPv6=(Not enabled)
31 15:30:37 client3/<CLIENT-IP>:39086 MULTI: Learn: 10.200.0.2 →
client3/<CLIENT-IP>:39086
32 15:30:37 client3/<CLIENT-IP>:39086 MULTI: primary virtual IP
for client3/<CLIENT-IP>:39086: 10.200.0.2
33 15:30:39 client3/<CLIENT-IP>:39086 PUSH: Received control
message: 'PUSH_REQUEST'
34 15:30:39 client3/<CLIENT-IP>:39086 send_push_reply():
safe_cap=940
35 15:30:39 client3/<CLIENT-IP>:39086 SENT CONTROL [client3]:
'PUSH_REPLY,route-gateway 10.200.0.1,topology subnet,
```

```
ping 10,ping-restart 60,
ifconfig 10.200.0.2 255.255.255.0' (status=1)
```

Давайте пройдемся по новым записям журнала:

- Строка 21 указывает, что исходный пакет был получен от клиента с IP-адресом <CLIENT-IP>. Обычно полный адрес IPv4 указан здесь.
- Строки 22 и 23 показывают процесс проверки сертификата, предоставленного клиентом OpenVPN. Важной частью в этих строках журнала является VERIFY-OK.
- Строки с 24 по 27 перечисляют используемый тип шифрования и дешифрования, а также хэши SHA1, используемые для хеширования входящего и исходящего трафика в канале данных. **BF-CBC (Blowfish Cipher Block Chaining)** - текущий шифр по умолчанию для OpenVPN.
- В строке 28 показан шифр TLS, используемый для защиты канала управления OpenVPN. Перечисленный здесь шифр очень похож на код шифрования, используемый защищенным веб-сервером.
- Строка 29 указывает, что клиент client3 с IP-адреса <CLIENT-IP> успешно прошел процесс аутентификации.
- В строках с 30 по 32 указывается адрес пула, который будет назначен этому клиенту.
- Строки с 33 по 34 показывают что клиент запросил информацию о конфигурации (PUSH REQUEST) и ответ от сервера - он отправляет push\_reply.
- Строка 35 показывает содержимое сообщения push\_reply со всей информацией о конфигурации для этого клиента. Эта строка чрезвычайно полезна при отладке установки OpenVPN, поскольку она показывает большую часть информации, которую сервер OpenVPN имеет для конкретного клиента, независимо от используемого файла конфигурации.

Аналогично, вот файл журнала клиента (запишите временные метки и сопоставьте их с временными метками из файла журнала сервера):

```
1 15:30:37 OpenVPN 2.3.6 x86_64-redhat-linux-gnu
 [SSL (OpenSSL)] [LZO] [EPOLL] [PKCS11] [MH] [IPv6]
 built on Dec 2 2014
2 15:30:37 library versions: OpenSSL 1.0.1e-fips 11 Feb 2013,
 LZO 2.03
3 15:30:37 Control Channel Authentication: using
 '/etc/openvpn/movpn/ta.key' as a OpenVPN static key
 file
4 15:30:37 UDPv4 link local: [undef]
5 15:30:37 UDPv4 link remote: [AF_INET]<SERVER-IP>:1194
6 15:30:37 [Mastering OpenVPN Server] Peer Connection Initiated
 with [AF_INET]<SERVER-IP>:1194
7 15:30:39 TUN/TAP device tun0 opened
8 15:30:39 do_ifconfig, tt->ipv6=0, tt-did_ifconfig_ipv6_setup=0
9 15:30:39 /sbin/ip link set dev tun0 up mtu 1500
10 15:30:39 /sbin/ip addr add dev tun0 10.200.0.2/24
 broadcast 10.200.0.255
11 15:30:39 Initialization Sequence Completed
```

Давайте пройдемся по новым записям журнала:

- Строчки 1 и 2 очень похожи на строки из журнала сервера.
- Строка 3 указывает, что канал управления защищен с помощью параметра конфигурации tls-auth и OpenVPN смог успешно прочитать указанный файл.

- Строки 4 и 5 говорят нам что клиент не связывался с локальным IP-адресом и было установлено соединение UDP с сервером по IP-адресу <SERVER-IP> и порту 1194.
- В строке 6 указано, что соединение с сервером OpenVPN, идентифицирующим себя как Mastering OpenVPN Server, было успешно установлено. Имя сервера извлекается из **common name** сертификата на стороне сервера.
- Строка 7 говорит нам, что OpenVPN смог открыть TUN-устройство tun0.
- Строки с 8 по 10 перечисляют информацию IPv4, которую сервер передал к этому клиенту и показывают, что IP-адрес и маска сети задаются с помощью стандартной команды Linux /sbin/ip.
- Строка 11 снова является волшебной строкой, сообщающей нам, что VPN-соединение было успешно установлено и теперь мы можем безопасно общаться с сервером OpenVPN. Однако, как мы увидим позже, сообщения об ошибках могут еще не появиться.

## Обнаружение неработающей установки

Установка OpenVPN может не работать по многим причинам. В следующем разделе мы рассмотрим список распространенных сбоев. Во-первых, давайте посмотрим, что отображается в файлах журналов при неудачной попытке подключения. Сбои могут возникать очень рано при попытке подключения или даже после строки Initialization Sequence Completed.

Если мы используем неправильный файл tls-auth - соединение очень рано оборвется. Это как раз и является причиной использования файла tls-auth, поскольку минимизирует нагрузку на наш сервер OpenVPN, когда мошеннические клиенты пытаются получить доступ. Клиент, который пытается подключиться без указания файла tls-auth, будет отображаться в журналах сервера следующим образом:

```
16:40:31 Authenticate/Decrypt packet error:
 packet HMAC authentication failed
16:40:31 TLS Error: incoming packet authentication failed from
 [AF_INET]<CLIENT-IP>:49956
16:40:33 Authenticate/Decrypt packet error:
 packet HMAC authentication failed
16:40:33 TLS Error: incoming packet authentication failed from
 [AF_INET]<CLIENT-IP>:49956
16:40:37 Authenticate/Decrypt packet error:
 packet HMAC authentication failed
16:40:37 TLS Error: incoming packet authentication failed from
 [AF_INET]<CLIENT-IP>:49956
16:40:45 Authenticate/Decrypt packet error:
 packet HMAC authentication failed
16:40:45 TLS Error: incoming packet authentication failed from
 [AF_INET]<CLIENT-IP>:49956
16:41:01 Authenticate/Decrypt packet error:
 packet HMAC authentication failed
16:41:01 TLS Error: incoming packet authentication failed from
 [AF_INET]<CLIENT-IP>:49956
```

Об этом клиенте больше ничего не сообщается, поскольку сервер OpenVPN немедленно отклоняет попытку подключения. Из временных меток в файле журнала мы видим, что клиент увеличивает время задержки между попытками соединения с каждым неудачным соединением. Если в течение 60 секунд соединение не может быть установлено, клиент прервет:

**TLS Error: TLS key negotiation failed to occur within 60 seconds  
(check your network connectivity)**

## **TLS Error: TLS handshake failed**

Второй сбой соединения станет очевидным только после того, как соединение будет успешно инициализировано. Для этого мы указываем использование другого кодирующего шифра на одной стороне, но забываем сделать это на другой. В файле журнала клиента теперь будет отображаться следующее:

```
16:56:20 /sbin/ip link set dev tun0 up mtu 1500
16:56:20 /sbin/ip addr add dev tun0 10.200.0.2/24 broadcast 10.200.0.255
16:56:20 Initialization Sequence Completed
16:56:30 Authenticate/Decrypt packet error: cipher final failed
16:56:40 Authenticate/Decrypt packet error: cipher final failed
```

Таким образом, сначала соединение, кажется, было успешно установлено (строки с 1 по 3), но через 10 секунд шифрование и дешифрование канала данных не удается.

### **Заметка**

Если бы в этом случае использовался графический интерфейс Windows – значок графического интерфейса стал бы зеленым, но сама VPN не работала бы!

Во время инициализации будет сообщено о большинстве проблем конфигурации на стороне сервера или клиента. О проблемах маршрутизации, встречаемых гораздо чаще, OpenVPN обычно не сообщает. Следовательно, требуются различные методы устранения неполадок.

## **Исправление распространенных ошибок конфигурации**

При настройке конфигурации OpenVPN есть несколько распространенных ошибок, которые легко допустить. Эти ошибки конфигурации можно условно разделить на четыре категории:

- Ошибки и несоответствия сертификатов (PKI)
- Несоответствие опций, таких как `tun` по сравнению с `tap`, шифрование и сжатие
- Недостаточно прав для запуска OpenVPN
- Ошибки маршрутизации

В этом разделе мы рассмотрим первые три из этих категорий. Ошибки маршрутизации будут обсуждаться в этой главе позже.

## **Неправильный сертификат CA в конфигурации клиента**

Файл конфигурации клиента почти всегда будет содержать три строки, подобные этой:

```
ca ca.crt
cert client.crt
key client.key
```

Эти файлы сертификатов и секретных ключей были созданы в [главе 3, PKI и сертификаты](#) и широко используются в последующих главах.

Файл CA, однако, не должен указывать центр сертификации, используемый для подписи файла сертификата клиента. Это должен быть публичный сертификат центра сертификации, который использовался для подписи сертификата сервера. Если сертификат сервера был подписан другим центром сертификации – клиент откажется подключиться к серверу. Это можно увидеть в файле журнала на стороне клиента:

```
UDPV4 link remote: [AF_INET]<SERVER-IP>:1194
VERIFY ERROR: depth=1, error=self signed certificate in certificate
chain: C=ZA, ST=Enlightenment, L=Overall, O=Mastering OpenVPN,
CN=Mastering OpenVPN, emailAddress=root@example.org
TLS_ERROR: BIO read tls_read_plaintext error: error:14090086:SSL
routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed
TLS Error: TLS object -> incoming plaintext read error
```

## **TLS Error: TLS handshake failed**

В этом случае ошибки не регистрируются на стороне сервера, так как сертификат клиента считается действительным на сервере.

Единственное, что зарегистрируется на сервере, это:

```
<CLIENT-IP>:42472 TLS: Initial packet from
[AF_INET]<CLIENT-IP>:42472, sid=9a1e4a84 cdbb6926
<CLIENT-IP>:51441 TLS: Initial packet from
[AF_INET]<CLIENT-IP>:51441, sid=17d3c89b 6999ae97
<CLIENT-IP>:43513 TLS: Initial packet from
[AF_INET]<CLIENT-IP>:43513, sid=4609202f 4c91c23d
```

Это показывает последовательные попытки подключения, которые сделаны клиентом OpenVPN.

### **Как исправить**

Убедитесь, что в файле конфигурации клиента указан правильный файл CA.

## **Сертификат клиента не распознан сервером**

Если сертификат клиента не распознан сервером - сервер откажет в доступе к нему. Это может произойти, если используется неправильный (или мошеннический) клиентский сертификат или если клиентский сертификат был отозван, а опция `crl-verify` указана в файле конфигурации сервера.

Следующие записи будут отображаться в файле журнала сервера, если неизвестный клиент попытается подключиться к серверу OpenVPN:

```
<CLIENT-IP>:57072 TLS: Initial packet from
[AF_INET]<CLIENT-IP>:57072, sid=a175f1be 6faed111
<CLIENT-IP>:57072 VERIFY ERROR: depth=0, error=unable to get
local issuer certificate: C=NL, O=Cookbook, CN=client1,
name=Cookbook Client, emailAddress=janjust@nikhef.nl
<CLIENT-IP>:57072 TLS_ERROR: BIO read tls_read_plaintext error:
error:140890B2:SSL routines:SSL3_GET_CLIENT_CERTIFICATE:
no certificate returned
<CLIENT-IP>:57072 TLS Error: TLS object -> incoming plaintext
read error
<CLIENT-IP>:57072 TLS Error: TLS handshake failed
```

Сервер не может проверить сертификат клиента, так как он не распознает сертификат CA, используемый для его подписи. Поэтому отказывается разрешить этому клиенту подключение.

На стороне клиента никакие сообщения не печатаются в файле журнала в течение 60 секунд, после чего первоначальное рукопожатие прекращается и делается новая попытка подключения:

```
13:24:23 UDPv4 link local: [undef]
13:24:23 UDPv4 link remote: [AF_INET]<SERVER-IP>:1194
13:25:23 TLS Error:
TLS key negotiation failed to occur within
60 seconds (check your network connectivity)
13:25:23 TLS Error: TLS handshake failed
13:25:23 SIGUSR1[soft,tls-error] received, process restarting
13:25:25 Control Channel Authentication: using
'/etc/openvpn/movpn/ta.key' as a OpenVPN static key file
13:25:25 UDPv4 link local: [undef]
13:25:25 UDPv4 link remote: [AF_INET]<SERVER-IP>:1194
```

### **Как исправить**

Убедитесь, что сертификат клиента распознается сервером. Это можно сделать либо указав

соответствующий сертификат СА в файле конфигурации сервера, либо добавив сертификат СА в составленный файл сертификата СА в файле конфигурации сервера:

```
cat CA1.crt CA2.crt > /etc/openvpn/movpn/ca-stack.pem
```

Далее используйте следующее в конфигурации сервера:

```
ca /etc/openvpn/movpn/ca-stack.pem
```

Таким образом, клиентские сертификаты, подписанные CA1.crt или CA2.crt будут приняты сервером.

Конечно, если это мошенник, пытающийся подключиться, то более подходящим решением может быть черный список IP-адресов, с которых подключается клиент.

## Несоответствие сертификата клиента и закрытого ключа

Если сертификат и закрытый ключ на клиенте не совпадают, то OpenVPN даже не будет пытаться подключиться к серверу. Вместо этого в файле журнала будет напечатана следующая ошибка:

```
Cannot load private key file /etc/openvpn/movpn/client1.key:
error:0B080074:x509 certificate routines:
 X509_check_private_key:key values mismatch
Error: private key password verification failed
Exiting due to fatal error
```

Эта проблема может возникнуть, особенно, когда сертификат и закрытый ключ обновляются; Распространенной ошибкой является использование старого закрытого ключа с новым сертификатом.

### Как исправить

Убедитесь, что сертификат клиента и приватный ключ совпадают. Удивительно, но для этого не существует простого в использовании инструмента. Чтобы выяснить, принадлежат ли сертификат и закрытый ключ друг другу - мы можем использовать следующие команды и искать разделы modulus:

```
$ openssl x509 -text -noout -in client1.crt
[...]
Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
Modulus:
00:b2:17:bd:31:6d:56:d9:eb:c9:09:98:e2:c1:48:
c9:6a:e4:4a:6b:54:52:ea:1e:60:94:6b:cb:5e:d5:
a1:ef:83:05:f8:cf:a4:06:df:06:ee:d6:c8:75:65:
de:a7:96:68:a1:41:d1:9d:f0:2c:84:3f:ca:b9:d2:
e8:07:af:37:48:24:69:57:4e:09:70:66:47:6c:47:
36:4d:c9:29:13:eb:ed:c1:aa:cd:36:84:3c:55:18:
bc:ce:01:34:b5:89:04:dc:09:c5:ea:f2:57:9f:c2:
f5:c1:05:dd:66:4d:11:13:05:47:46:26:1a:55:18:
51:bd:89:65:ba:0d:89:bd:ea:03:58:5e:d3:d9:96:
a5:5e:2f:5f:b9:c8:88:fc:48:95:cb:4a:b2:12:3b:
b5:ed:4c:40:4c:50:8d:1d:eb:a5:c9:c0:e6:2c:ec:
01:0a:56:ac:db:9e:e7:56:f0:06:f7:ba:b6:ac:de:
41:d4:fb:b3:d6:f5:fe:13:b4:03:81:d9:f7:7c:2e:
60:2f:9c:5a:81:eb:2e:3a:e1:c4:8b:f8:b6:8d:2d:
f7:ec:7a:f6:2c:ff:af:1c:d2:7b:58:ca:9e:d1:f4:
ed:8a:7a:35:00:97:a3:35:dd:79:02:b4:79:9a:66:
3c:5e:c8:4d:87:eb:68:5d:45:29:73:70:7f:61:28:
67:b1
```

```
$ openssl rsa -text -noout -in client1.key
```

```
Private-Key: (2048 bit)
modulus:
00:b2:17:bd:31:6d:56:d9:eb:c9:09:98:e2:c1:48:
c9:6a:e4:4a:6b:54:52:ea:1e:60:94:6b:cb:5e:d5:
a1:ef:83:05:f8:cf:a4:06:df:06:ee:d6:c8:75:65:
de:a7:96:68:a1:41:d1:9d:f0:2c:84:3f:ca:b9:d2:
e8:07:af:37:48:24:69:57:4e:09:70:66:47:6c:47:
36:4d:c9:29:13:eb:ed:c1:aa:cd:36:84:3c:55:18:
bc:ce:01:34:b5:89:04:dc:09:c5:ea:f2:57:9f:c2:
f5:c1:05:dd:66:4d:11:13:05:47:46:26:1a:55:18:
51:bd:89:65:ba:0d:89:bd:ea:03:58:5e:d3:d9:96:
a5:5e:2f:5f:b9:c8:88:fc:48:95:cb:4a:b2:12:3b:
b5:ed:4c:40:4c:50:8d:1d:eb:a5:c9:c0:e6:2c:ec:
01:0a:56:ac:db:9e:e7:56:f0:06:f7:ba:b6:ac:de:
41:d4:fb:b3:d6:f5:fe:13:b4:03:81:d9:f7:7c:2e:
60:2f:9c:5a:81:eb:2e:3a:e1:c4:8b:f8:b6:8d:2d:
f7:ec:7a:f6:2c:ff:af:1c:d2:7b:58:ca:9e:d1:f4:
ed:8a:7a:35:00:97:a3:35:dd:79:02:b4:79:9a:66:
3c:5e:c8:4d:87:eb:68:5d:45:29:73:70:7f:61:28:
67:b1
```

[...]

Если мы посмотрим на модуль с открытого ключа (сертификата) и приватного ключа, то увидим что они одинаковы. Таким образом, этот сертификат и приватный ключ принадлежат друг другу.

### Подсказка

При сравнении модулей часто достаточно сравнить первые несколько байтов, а затем последние несколько байтов.

## Несоответствие ключей auth и tls-auth

Параметры `auth` и `tls-auth` используются для аутентификации пакетов канала управления и канала данных с использованием алгоритма подписи HMAC. Значением по умолчанию для алгоритма аутентификации HMAC является SHA1, в котором используются 160-битные ключи. Для опции `tls-auth` нет значения по умолчанию, так как оно не требуется. Однако этот вариант рекомендуется, поскольку он обеспечивает дополнительный уровень защиты от DDoS-атак.

Если алгоритм аутентификации, указанный в конфигурации клиента и сервера, не совпадает, то сервер не позволит клиенту начать квитирование безопасности TLS. Аналогичным образом, если файлы `tls-auth` на клиенте и сервере не совпадают или если с обеих сторон указан неверный параметр `direction` - сервер также не позволит клиенту начать квитирование безопасности TLS.

Обычно в файле конфигурации сервера указывается следующая опция:

```
tls-auth /etc/openvpn/movpn/ta.key 0
```

Соответственно, на клиенте у нас есть следующая опция:

```
tls-auth /etc/openvpn/movpn/ta.key 1
```

Здесь второй параметр определяет `direction` из `tls-auth` для используемых ключей. Этот параметр необязателен, но он позволяет OpenVPN использовать разные ключи хеширования (или HMAC) для входящего и исходящего трафика. Ключ, используемый на клиенте для подписи исходящего трафика, должен совпадать с ключом, используемым на сервере для проверки входящего трафика, и наоборот.

Если используется неверный файл ключей `tls-auth` или если направление опущено или указано неверно – в журнале сервера появятся следующие записи:

```
Authenticate/Decrypt packet error: packet HMAC
```

```
authentication failed
TLS Error: incoming packet authentication failed from
[AF_INET]<CLIENT-IP>:54377
```

В то же время, клиент просто попытается подключиться в течение 60 секунд, прежде чем произойдет тайм-аут.

## Как исправить

Убедитесь, что используется один и тот же файл `tls-auth` в файлах конфигурации клиента и сервера. Также убедитесь, что параметр `direction` указан правильно на обоих концах (если используется вообще).

Если вы все еще не уверены, какие ключи HMAC используются для входящих и исходящих соединений, то можете увеличить детализацию файла журнала, чтобы увидеть фактические ключи, используемые как клиентом, так и сервером. Давайте добавим следующее в файлы конфигурации клиента и сервера:

```
verb 7
```

Теперь обе стороны будут печатать большое количество информации о регистрации при запуске.

Строки для поиска в файле журнала на стороне сервера:

```
Outgoing Control Channel Authentication:
 Using 160 bit message hash 'SHA1' for HMAC authentication
Outgoing Control Channel Authentication:
 HMAC KEY: 4660a714 7f4d33f9 d2f7c61a 9f1d5743 4bf9411e
Outgoing Control Channel Authentication:
 HMAC size=20 block_size=20
Incoming Control Channel Authentication:
 Using 160 bit message hash 'SHA1' for HMAC authentication
Incoming Control Channel Authentication:
 HMAC KEY: cd1f6d9c 88db5ec7 d7977322 e01d14f1 26ee4e22
Incoming Control Channel Authentication:
 HMAC size=20 block_size=20
```

Строка `HMAC size = 20` соответствует тому, что используется 160-битовое хеширование с помощью SHA1, так как 160 соответствуют 20 байтам.

Если на стороне клиента используются правильный файл `tls-auth` и параметр `direction`, мы найдем следующее:

```
Outgoing Control Channel Authentication:
 Using 160 bit message hash 'SHA1' for HMAC authentication
Outgoing Control Channel Authentication:
 HMAC KEY: cd1f6d9c 88db5ec7 d7977322 e01d14f1 26ee4e22
Outgoing Control Channel Authentication:
 HMAC size=20 block_size=20
Incoming Control Channel Authentication:
 Using 160 bit message hash 'SHA1' for HMAC authentication
Incoming Control Channel Authentication:
 HMAC KEY: 4660a714 7f4d33f9 d2f7c61a 9f1d5743 4bf9411e
Incoming Control Channel Authentication:
 HMAC size=20 block_size=20
```

Ключи аутентификации входящего и исходящего каналов управления зеркально отображаются на клиенте и на сервере, обеспечивая надлежащую аутентификацию TLS.

## Несоответствие размера MTU

OpenVPN использует два размера **максимальной единицы передачи (MTU)**:

- **tun-mtu**: указывает настройку MTU адаптера tun и максимальный размер каждого пакета внутри VPN-туннеля.
- **link-mtu**: указывает максимальный размер каждого пакета вне туннеля. Он включает в себя все биты заполнения, шифрования и аутентификации, но это не фактический размер пакета при передаче по сети. Фактический размер пакета не может быть определен заранее, так как размер каждого пакета может отличаться из-за алгоритмов сжатия и шифрования.

Значение по умолчанию для параметра **tun-mtu** составляет 1500 байт, что также является размером MTU по умолчанию для адаптера Ethernet. При нормальных обстоятельствах мы можем использовать следующую формулу для вычисления размера **link-mtu** из размера **tun-mtu**:

**link-mtu = tun-mtu + constant**

Здесь **constant** зависит от используемых параметров конфигурации. Среди параметров конфигурации, которые влияют на эту константу, мы имеем следующие:

- Варианты сжатия, такие как **comp-lzo** и **comp-noadapt**
- Размер **вектора инициализации (IV)** параметра шифрования опции **cipher**
- Опция **fragment**, добавляющая дополнительный байт
- Опция **no-replay**, которая удаляет байт.

Если мы видим несоответствие предупреждений **link-mtu** - это обычно указывает на неправильную конфигурацию в других местах наших файлов конфигурации клиента и сервера. Как вы увидите в последующих примерах, несоответствие в **link-mtu** между клиентом и сервером может происходить довольно часто. Обычно VPN-соединение не будет работать правильно, если имеется несоответствие **link-mtu**.

### Подсказка

Не поддавайтесь искушению исправить само предупреждение **link-mtu** явно установив его. Сначала исправьте другие предупреждения, которые могли вызвать появление предупреждения **link-mtu**.

Параметр **link-mtu** также имеет большое значение при настройке VPN-соединения.

Чтобы получить максимальную производительность от VPN-соединения - нам нужно убедиться, что пакеты не фрагментируются операционной системой, поскольку это сильно повлияет на производительность. В частности, для спутниковых каналов это может привести к снижению производительности почти до полной остановки.

Если на стороне сервера указан другой размер MTU, по сравнению со стороной клиента, в файлах журнала появится следующее предупреждение:

```
WARNING: 'link-mtu' is used inconsistently,
local='link-mtu 1441', remote='link-mtu 1541'
WARNING: 'tun-mtu' is used inconsistently,
local='tun-mtu 1400', remote='tun-mtu 1500'
```

Это показывает, что для конфигурации **default**, издержки **link-mtu** на самом деле составляют 41 байт. Здесь мы добавили в файл конфигурации клиента:

**tun-mtu 1400**

На этом этапе VPN-соединение будет функционировать. Однако производительность будет ограничена, так как пакеты должны быть фрагментированы и повторно собраны. При такой настройке можно вызвать ошибку, отправив большие пакеты ICMP с установленным флагом

not fragment. В Linux/FreeBSD это можно сделать с помощью следующей команды:

```
$ ping -M do -s 1450 10.200.0.2
```

В Windows мы используем следующее:

```
C:\> ping -f -l 1450 10.200.0.2
```

Это приведет к 100-процентной потере пакета для команды ping, а также будет отображаться в файле журнала:

```
Authenticate/Decrypt packet error:
packet HMAC authentication failed
```

Это сообщение об ошибке может сначала показаться странным, но оно вызвано тем, что отправляющая сторона создала и подписала пакет, размер которого превышает 1400 байт. Клиент получает только первые 1400 байтов этого пакета и проверяет подпись, которая терпит неудачу. Затем он отклоняет пакет и распечатывает ошибку.

### Как исправить

Убедитесь, что, если вы хотите использовать опцию tun-mtu - она указана в файлах конфигурации клиента и сервера.

## Несоответствие шифрования

Алгоритм шифрования, используемый для канала данных OpenVPN, можно указать, используя следующую опцию со значением по умолчанию BF-CBC:

```
cipher aes-256-cbc
```

Если в файле конфигурации клиента указан другой шифр, чем в файле конфигурации сервера, то в файлах журнала с обеих сторон будет предупреждающее сообщение, но VPN-соединение будет установлено. Однако, как только любой трафик пойдет по нему – его невозможно будет расшифровать. Мы можем видеть это в следующем фрагменте из файла журнала на стороне клиента:

```
WARNING: 'link-mtu' is used inconsistently,
local='link-mtu 1557', remote='link-mtu 1541'
WARNING: 'cipher' is used inconsistently,
local='cipher AES-256-CBC', remote='cipher BF-CBC'
WARNING: 'keysize' is used inconsistently,
local='keysize 256', remote='keysize 128'
[Mastering OpenVPN Server] Peer Connection Initiated
with [AF_INET]<SERVER-IP>:1194
TUN/TAP device tun0 opened
do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
/sbin/ip link set dev tun0 up mtu 1500
/sbin/ip addr add dev tun0 10.200.0.2/24 broadcast 10.200.0.255
Initialization Sequence Completed
Authenticate/Decrypt packet error: cipher final failed
```

Три напечатанных предупреждения изначально показывают как другой тип, так и другой размер используемого шифра. Шифр Blowfish по умолчанию использует 128-битную стойкость, тогда как AES-256 - 256-битную, что приводит к немного большему зашифрованному пакету (link-mtu 1541 байт для Blowfish по сравнению с link-mtu 1557 байт для AES-256).

### Как исправить

Убедитесь, что в файлах конфигурации клиента и сервера указан один и тот же шифр. Поскольку файлы журналов клиента и сервера выводят ожидаемый шифр, исправить эту ошибку довольно просто.

### **Заметка**

В настоящее время невозможно передать шифр с сервера на клиент. Это в списке пожеланий разработчиков OpenVPN, но оно оказывает существенное влияние на код. И не будет добавлено в OpenVPN до версии 2.4 или даже 2.5.

## **Несоответствие сжатия**

OpenVPN имеет возможность сжимать весь VPN-трафик на лету. Для определенных типов трафика, таких как обычный веб-трафик, это может повысить производительность VPN, но добавляет дополнительные издержки к протоколу VPN. Для несжимаемого трафика эта опция фактически немного снижает производительность.

Параметр, используемый для указания сжатия в настоящее время, выглядит следующим образом:

```
comp-lzo [no|yes|adaptive]
```

Обратите внимание, что нам не нужно указывать второй параметр. Значение по умолчанию является адаптивным, если используется сжатие.

Как мы узнаем в [Главе 10, Будущие направления](#), этот вариант будет заменен более общим вариантом compression, что позволит различные механизмы сжатия.

Можно передать опцию compression с сервера на клиента, но только если опция сжатия была указана в самом файле конфигурации клиента. Если файл конфигурации клиента не содержит такой опции - VPN-соединение не будет установлено. В файле журнала клиента будет показано следующее:

```
UDPV4 link remote: [AF_INET]<SERVER-IP>:1194
WARNING: 'link-mtu' is used inconsistently,
 local='link-mtu 1541', remote='link-mtu 1542'
WARNING: 'comp-lzo' is present in remote config but
 missing in local config, remote='comp-lzo'
[Mastering OpenVPN Server] Peer Connection Initiated with
 [AF_INET]<SERVER-IP>:1194
TUN/TAP device tun0 opened
do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
/sbin/ip link set dev tun0 up mtu 1500
/sbin/ip addr add dev tun0 10.200.0.2/24 broadcast 10.200.0.255
Initialization Sequence Completed
write to TUN/TAP : Invalid argument (code=22)
```

Файл журнала сервера будет содержать те же WARNING, а также будет отображать предупреждения распаковки:

```
client3<CLIENT-IP>:45113 Bad LZO decompression header byte: 42
```

### **Заметка**

Странно, но верно: если мы будем ждать достаточно долго, клиент будет перезагружен из-за ошибок сжатия и попытается восстановить соединение. На этот раз, однако, соединение будет успешным, так как опция comp-lzo все еще находится в памяти.

## **Как исправить**

Убедитесь, что, если вы хотите использовать сжатие, опция comp-lzo указана в файлах конфигурации клиента и сервера. С опцией comp-lzo в файле конфигурации на стороне клиента мы теперь можем контролировать тип сжатия, используемый на стороне сервера, используя опцию push. Используйте следующее:

```
comp-lzo no
push "comp-lzo no"
```

Это отключит сжатие, но, к сожалению, это не то же самое, что вообще не указывать какой-либо метод сжатия. Надеемся, что оно будет решено в будущем выпуске.

## Несоответствие фрагмента

Одним из наиболее часто используемых параметров настройки является опция `fragment`. Подробнее об этой опции вы узнаете в разделе *Как оптимизировать производительность с помощью ping и iperf* далее в этой главе.

Как и параметр `comp-lzo`, параметр `fragment` указывать не нужно ни с одной стороны. Однако мы не можем указать его только с одной стороны; он должен быть настроен на обеих. Если он указан только с одной стороны, то также должен быть указан и с другой. Технически говоря, даже нет необходимости использовать одно и то же значение для параметра `fragment` с обеих сторон, но это рекомендуется.

Если опция `fragment` не указана на стороне клиента, но используется на стороне сервера, то VPN-соединение не будет работать должным образом, как видно из журнала клиента:

```
WARNING: 'link-mtu' is used inconsistently,
 local='link-mtu 1541', remote='link-mtu 1545'
WARNING: 'mtu-dynamic' is present in remote config but
 missing in local config, remote='mtu-dynamic'
[Mastering OpenVPN Server] Peer Connection Initiated with
 [AF_INET]194.171.96.101:1194
TUN/TAP device tun0 opened
do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
/sbin/ip link set dev tun0 up mtu 1500
/sbin/ip addr add dev tun0 10.200.0.2/24 broadcast 10.200.0.255
Initialization Sequence Completed
write to TUN/TAP : Invalid argument (code=22)
```

Опять же, это будет выглядеть так, как будто VPN подключился (*Initialization sequence completed*), но файл журнала заполнится сообщениями об ошибках с `code=22`.

Обратите внимание, что в предупреждении фактически указан `mtu-dynamic`, который является устаревшим названием этой функции.

## Как исправить

Убедитесь, что, если вы хотите использовать параметр `fragment` - он указывается в файлах конфигурации клиента и сервера.

Обратите внимание, что, в отличие от опции `comp-lzo`, эту функцию нельзя передать с сервера на клиент.

## Несоответствие tun и tap

Наиболее распространенный вариант использования сети в режиме `tap` - это мостовая установка, как мы узнали из [Главы 6](#), *Режим клиент-сервер с tap-устройствами*. Однако не все устройства поддерживают сеть в режиме `tap`. В частности, все устройства Android и iOS не имеют этой возможности. Следовательно, если мы подключим такое устройство к серверу OpenVPN в режиме `tap`, в файле журнала сервера будут перечислены предупреждения от этих клиентов:

```
<CLIENT-IP>:39959 WARNING: 'dev-type' is used inconsistently,
 local='dev-type tap', remote='dev-type tun'
<CLIENT-IP>:39959 WARNING: 'link-mtu' is used inconsistently,
 local='link-mtu 1573', remote='link-mtu 1541'
```

```
<CLIENT-IP>:39959 WARNING: 'tun-mtu' is used inconsistently,
local='tun-mtu 1532', remote='tun-mtu 1500'
```

Помимо этих предупреждений сервер не обнаружит ничего о подключающихся клиентах. На клиенте аналогичные предупреждения будут перечислены вместе с этим:

```
WARNING: Since you are using --dev tun with a point-to-point topology, the
second argument to --ifconfig must be an IP address.
You are using something (255.255.255.0) that looks more like a netmask.
(silence this warning with --ifconfig-nowarn)
```

Так как мы не можем передать топологию подсети в настройке режима tap – клиент возвращается к сети по умолчанию в режиме Net30. Этот тип сети по своей сути несовместим с tap-сетью, но, кроме этого, клиент не выдает никаких предупреждений или ошибок.

Даже если бы мы (ошибочно) добавили `topology subnet` для подавления этого предупреждения на клиенте, VPN все равно не работал бы правильно.

## Как исправить

Убедитесь, что с обеих сторон используется один и тот же тип сети (tun или tap). Если вам необходимо использовать устройства Android или iOS - вы должны настроить конфигурацию сервера в режиме tun, так как эти операционные системы не поддерживают режим tap.

## Проблемы с client-config-dir

В [Главе 4](#), Режим клиент-сервер с устройствами tun, мы узнали о CCD-файлах и их использовании в разделе Специфичная для клиента конфигурация - файлы CCD. Файлы CCD обычно используются для подключения клиентской локальной сети к сети сервера с помощью оператора `iroute`.

Опыт работы со списками рассылки и форумами OpenVPN показал, что опция `client-config-dir` подвержена ошибкам и неправильной настройке. Вот три основные причины этого:

- Файл CCD или каталог, в котором он находится, не может быть прочитан OpenVPN после переключения на `safe` пользователя, такого как `nobody`.
- Опция `client-config-dir` указана без абсолютного пути.
- Имя файла CCD указано неверно. Обычно имя файла CCD совпадает с именем из поля `/CN=` сертификата клиента, без части `/CN=` и без какого-либо расширения!

При нормальном уровне журнала OpenVPN не жалуется, если не может найти или прочитать файл CCD. Он просто обрабатывает входящее соединение как стандартное, и, таким образом, требуемый оператор `iroute` никогда не достигается.

Самый простой способ отладки - временно добавить дополнительную опцию в конфигурацию сервера:

```
ccd-exclusive
```

Перезагрузите сервер и клиент попытается восстановить соединение. Если сервер не может прочитать соответствующий файл CCD для подключающегося клиента - он откажет в доступе. Если это произойдет - мы знаем, что файл CCD не был прочитан. Если клиент может подключиться, то возникает другая проблема - скорее всего, проблема маршрутизации.

Другой способ увидеть что сервер OpenVPN делает с файлами CCD - это повысить уровень журнала до 4 и повторно подключить клиента, для которого указан файл CCD. Содержимое этого CCD-файла для клиента с сертификатом `/CN=client1` выглядит следующим образом:

```
ifconfig-push 10.200.0.99 255.255.255.0
iroute 192.168.4.0 255.255.255.0
```

Это дает команду серверу OpenVPN назначить IP-адрес VPN 10.200.0.99 для этого клиента и для маршрутизации подсети 192.168.4.0/24 через него. Файл журнала сервера теперь содержит следующее:

```
<CLIENT-IP>:38876 [client1] Peer Connection Initiated with [AF_INET]<CLIENT-
IP>:38876
client1/<CLIENT-IP>:38876 OPTIONS IMPORT: reading client specific options
from: /etc/openvpn/movpn/clients/client1
client1/<CLIENT-IP>:38876 MULTI: Learn: 10.200.0.99 -> client1/<CLIENT-
IP>:38876
client1/<CLIENT-IP>:38876 MULTI: primary virtual IP for client1/<CLIENT-
IP>:38876: 10.200.0.99
client1/<CLIENT-IP>:38876 MULTI: internal route 192.168.4.0/24 ->
client1/<CLIENT-IP>:38876
client1/<CLIENT-IP>:38876 MULTI: Learn: 192.168.4.0/24 -> client1/<CLIENT-
IP>:38876
```

Если выделенная строка отсутствует, то файл CCD не читается. Также следующие строки, начинающиеся с MULTI: показывают как сервер OpenVPN интерпретирует строки, найденные в файле CCD. Это может быть важно для дальнейшей отладки любых вопросов iroute.

### Как исправить

Если процесс сервера не может прочитать файл CCD - проверьте права доступа к полному пути к файлу, включая все подкаталоги, ведущие к нему. Убедитесь, что пользователь, указанный в параметре user, имеет разрешение на чтение всех каталогов и самого файла CCD.

Убедитесь, что в опции client-config-dir указан абсолютный путь вместо относительного. Кроме того, если мы используем опцию chroot (подробности см. в man), убедитесь, что директория client-config-dir видна внутри chroot-jail.

Используйте опцию ccd-exclusive для быстрого определения возможности OpenVPN прочитать файл CCD. Если это возможно, то увеличьте уровень журнала на стороне сервера, чтобы увидеть, как OpenVPN интерпретирует операторы, найденные в файле CCD.

### Нет доступа к устройству tun в Linux

Если OpenVPN запускается с недостаточными привилегиями или если OpenVPN настроен на удаление привилегий root и переключение на другой userid (например, nobody), то доступ к устройству tun может быть потерян. Это также может произойти, если OpenVPN используется в виртуализированной среде, такой как OpenVZ или **Virtual Private Server (VPS)**.

Если OpenVPN запущен с недостаточными привилегиями - VPN-соединение вообще не будет установлено:

```
UDPV4 link local: [undef]
UDPV4 link remote: [AF_INET]<SERVER-IP>:1194
[Mastering OpenVPN Server] Peer Connection Initiated with
[AF_INET]<SERVER-IP>:1194
ERROR: Cannot ioctl TUNSETIFF tun: Operation not permitted
(errno=1)
Exiting due to fatal error
```

Проверьте userid или используйте sudo для переключения на привилегированного пользователя перед запуском OpenVPN.

Наиболее распространенный сценарий, когда недоступны достаточные привилегии, происходит после автоматического перезапуска VPN-подключения. Рассмотрим следующий файл конфигурации клиента:

```
client
```

```

proto udp
remote openvpnserver.example.com
port 1194
dev tun
nobind

remote-cert-tls server
tls-auth /etc/openvpn/movpn/ta.key 1
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/client3.crt
key /etc/openvpn/movpn/client3.key

user nobody
group nobody

```

Это базовый файл конфигурации с двумя строками внизу. Когда мы запускаем VPN-соединение с помощью этого файла конфигурации, соединение устанавливается правильно, но выводится предупреждение:

```
WARNING: you are using user/group/chroot/setcon without persist-tun -- this
may cause restarts to fail
```

Действительно, если VPN-соединение необходимо перезапустить (например, из-за плохого сетевого соединения), перезапуск будет неудачным:

```
[Mastering OpenVPN Server] Inactivity timeout (--ping-restart), restarting
Mon Jun 1 16:51:50 2015 sbinip addr del dev tun0 10.200.0.2/24
RTNETLINK answers: Operation not permitted
Linux ip addr del failed: external program exited with error status: 2
SIGUSR1[soft,ping-restart] received, process restarting
WARNING: you are using user/group/chroot/setcon without persist-key -- this
may cause restarts to fail
Error: private key password verification failed
Exiting due to fatal error
```

Здесь мы видим, что OpenVPN не удалось перезапустить, так как пользователю `nobody` не разрешили прочитать приватный ключ, используемый для этого соединения. Если бы мы указали пользователя с правами доступа, то увидели бы другую ошибку:

```
ERROR: Cannot ioctl TUNSETIFF tun: Operation not permitted
(errno=1)
Exiting due to fatal error
```

Обратите внимание, что во время перезапуска OpenVPN не может завершить работу существующего устройства `tun` или удалить любые системные маршруты. Это также будет иметь место, если используется `persist-tun`, но в этом случае он будет безвредным.

## Как исправить

Добавьте следующие параметры в файл конфигурации клиента, если вы также используете параметры `user` и/или `group`:

```
persist-tun
persist-key
```

Убедитесь, что вы запускаете OpenVPN с достаточными привилегиями.

Также убедитесь, что OpenVPN имеет правильный контекст безопасности SELinux, или попробуйте запустить OpenVPN с SELinux, установленным в разрешающий или отключенный режим:

```
setenforcing permissive
```

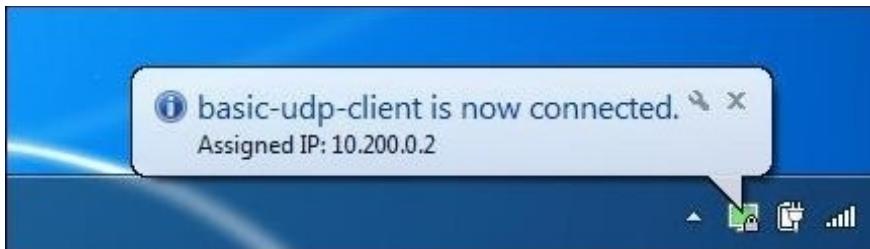
## Отсутствие повышенных привилегий в Windows

В некоторых старых версиях программы установки OpenVPN для Windows не были заданы правильные привилегии для приложения OpenVPN GUI.

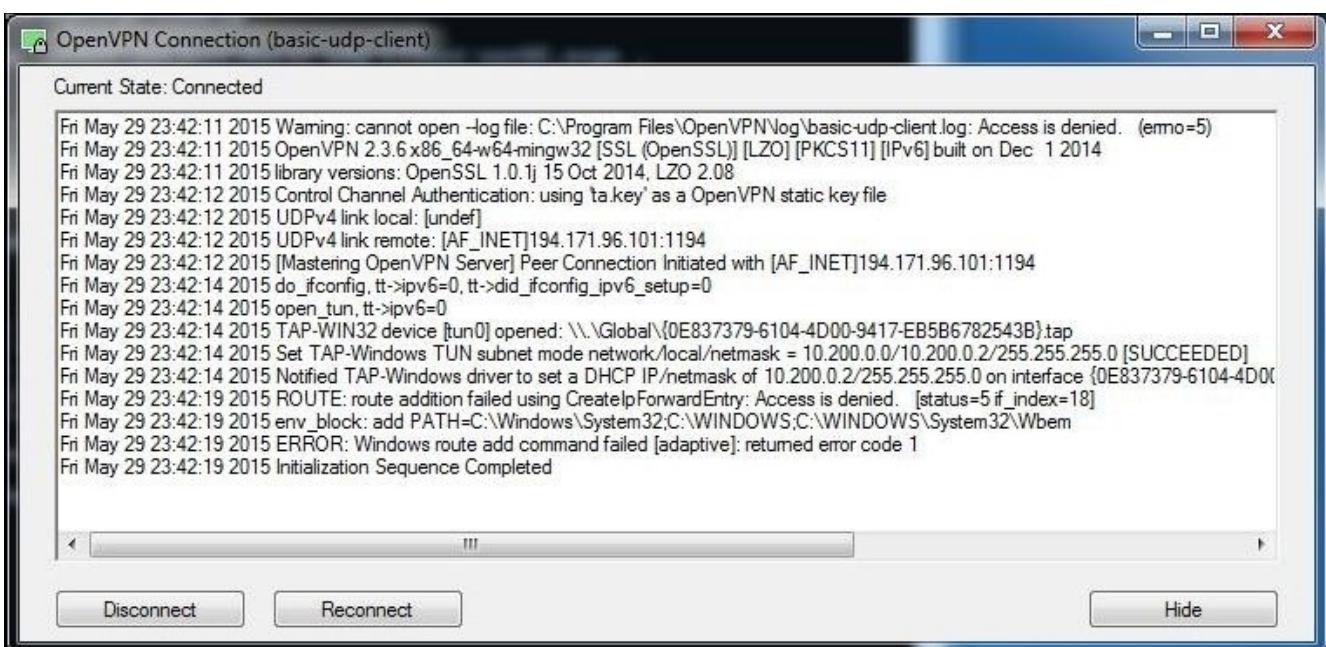
В этом конкретном примере один сервер был передан с сервера OpenVPN всем клиентам:

```
push "route 192.168.122.0 255.255.255.0"
```

В Windows Vista и выше OpenVPN требуются повышенные привилегии чтобы иметь возможность добавлять или удалять системные маршруты. Если эти привилегии отсутствуют – VPN обычно инициализируется правильно, а значок GUI становится зеленым:



Мы даже можем пропинговать сервер OpenVPN по его IP-адресу. Тем не менее, файл журнала в OpenVPN GUI покажет некоторые ошибки:



Первая строка на самом деле хитрая:

**Warning: cannot open -log file: ....: Access is denied**

Сложность в том, что как только мы нажимаем кнопку **Disconnect** – журнал исчезает, так как он не может быть записан на диск! Это вызвано тем, что каталог журналов по умолчанию `C:\Program Files\OpenVPN\log` доступен только пользователю с повышенными привилегиями.

Последние несколько строк в файле журнала говорят нам, что OpenVPN не удалось добавить маршрут, который был передан сервером. Опять же, это связано с тем, что программа OpenVPN была запущена с недостаточными привилегиями.

### Как исправить

После перезапуска OpenVPN GUI с повышенными правами (включите **Запуск от имени администратора**) маршрут будет добавлен правильно. Это видно из таблицы маршрутизации:

on: Console

IPv4 Route Table						
Active Routes:		Network Destination	Netmask	Gateway	Interface	Metric
		0.0.0.0	0.0.0.0	On-link	10.255.255.1	10255
		0.0.0.0	0.0.0.0	192.168.3.1	192.168.3.18	25
		10.200.0.0	255.255.255.0	On-link	10.200.0.2	276
		10.200.0.2	255.255.255.255	On-link	10.200.0.2	276
		10.200.0.255	255.255.255.255	On-link	10.200.0.2	276
		10.255.255.0	255.255.255.0	On-link	10.255.255.1	10255
		10.255.255.1	255.255.255.255	On-link	10.255.255.1	10255
		10.255.255.255	255.255.255.255	On-link	10.255.255.1	10255
		127.0.0.0	255.0.0.0	On-link	127.0.0.1	306
		127.0.0.1	255.255.255.255	On-link	127.0.0.1	306
		127.255.255.255	255.255.255.255	On-link	127.0.0.1	306
		192.168.3.0	255.255.255.0	On-link	192.168.3.18	281
		192.168.3.18	255.255.255.255	On-link	192.168.3.18	281
		192.168.3.255	255.255.255.255	On-link	192.168.3.18	281
		192.168.122.0	255.255.255.0	10.200.0.1	10.200.0.2	20
		224.0.0.0	240.0.0.0	On-link	127.0.0.1	306
		224.0.0.8	240.0.0.0	On-link	10.200.0.2	276
		224.0.0.0	240.0.0.0	On-link	10.255.255.1	10255
		224.0.0.0	240.0.0.0	On-link	192.168.3.18	281
		255.255.255.255	255.255.255.255	On-link	127.0.0.1	306
		255.255.255.255	255.255.255.255	On-link	10.200.0.2	276
		255.255.255.255	255.255.255.255	On-link	10.255.255.1	10255
		255.255.255.255	255.255.255.255	On-link	192.168.3.18	281

Переданный маршрут - 192.168.122.0/24 , теперь присутствует в таблице маршрутизации, используя IP-адрес VPN сервера 10.200.0.1 в качестве шлюза.

## Устранение проблем с маршрутизацией

Большинство вопросов, задаваемых в списках рассылки OpenVPN и форумах пользователей, на самом деле являются вопросами маршрутизации. Настройка VPN-соединения - это одно, а интеграция в существующую сеть - совсем другое. Для новичка трудная часть состоит в том, чтобы видеть, где заканчивается OpenVPN и где начинается маршрутизация. Этот раздел предназначен в качестве пошагового руководства по устранению неполадок маршрутизации в довольно простой настройке OpenVPN.

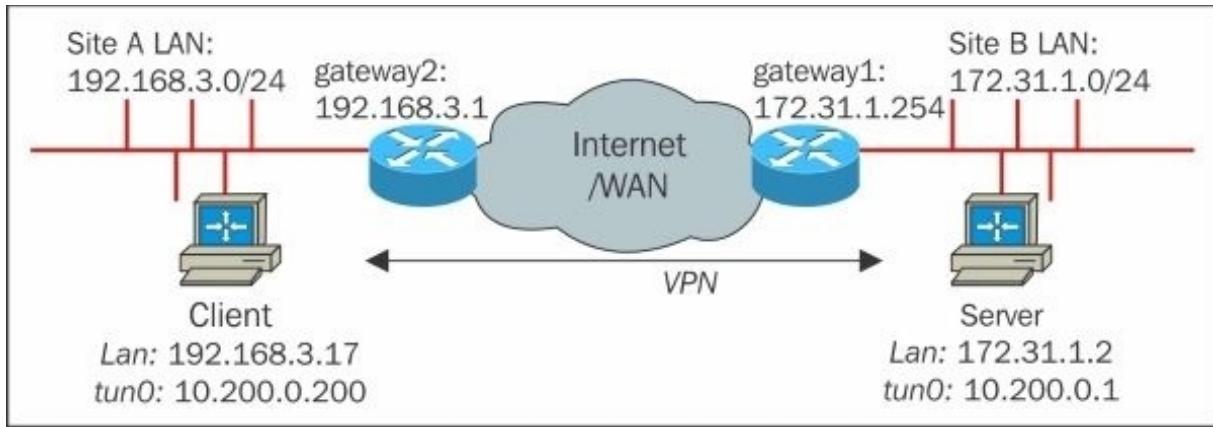
Рассмотрим следующий план сети:

- Сеть в главном офисе должна быть доступна для дополнительного офиса и для людей, работающих из дома
- Серверы в дополнительном офисе должны быть доступны для IT-отдела главного офиса
- Люди, работающие из дома, должны иметь доступ только к компьютерным ресурсам в главном офисе

Для этого в главном офисе устанавливается сервер OpenVPN, сотрудники которого подключаются как обычные клиенты VPN, а дополнительный офис подключается как специальный клиент, раскрывая свою собственную сеть.

## Рисование детальной картины

Перед созданием файлов конфигурации для OpenVPN нарисуйте подробное изображение схемы сети, включая все подсети, IP-адреса, IP-адрес шлюза, имена интерфейсов и многое другое.



Используемые публичные IP-адреса не перечислены на этом рисунке, но это рекомендуется сделать. Кроме того, не включены подключения от людей, работающих из дома, но они будут подключаться к общедоступному IP-адресу gateway1 на предыдущем рисунке.

На gateway1 добавлено правило переадресации портов, поэтому входящий и исходящий трафик UDP через порт 1194 перенаправляется на сервер OpenVPN в 172.31.1.2:1194.

Поскольку нам необходимо раскрыть сеть в дополнительном офисе, нам также потребуется использовать файл `client-config-dir` с соответствующим оператором `iroute`.

Файлы конфигурации сервера и клиента для этой настройки уже перечислены в [Глазе 4, Режим клиент-сервер с устройствами tun](#), с некоторыми незначительными изменениями IP-адреса. Новый набор файлов конфигурации выглядит следующим образом:

```

proto udp
port 1194
dev tun

server 10.200.0.0 255.255.255.0

tls-auth /etc/openvpn/movpn/ta.key 0
dh /etc/openvpn/movpn/dh2048.pem
ca /etc/openvpn/movpn/movpn-ca.crt
cert /etc/openvpn/movpn/server.crt
key /etc/openvpn/movpn/server.key

persist-key
persist-tun
keepalive 10 60

topology subnet

user nobody
group nobody

verb 3
daemon
log-append /var/log/openvpn.log

push "route 172.31.1.0 255.255.255.0"
client-config-dir /etc/openvpn/movpn/clients
route 192.168.3.0 255.255.255.0 10.200.0.1

```

Этот файл сохраняется как `movpn-09-01-server.conf`. Для клиента OpenVPN в дополнительном офисе создается специальный сертификат с именем `/CN=SecondaryOffice`. Соответствующий файл CCD, следовательно, имеет имя

/etc/openvpn/movpn/clients/SecondaryOffice. Его содержание таково:

```
ifconfig-push 10.200.0.200 255.255.255.0
iroute 192.168.3.0 255.255.255.0
```

Для всех клиентов может быть использован конфигурационный файл basic-udp-client.conf или basic-udp-client.ovpn. Это, кстати, показывает гибкость конфигурационных файлов OpenVPN. В большинстве случаев нет необходимости изменять файлы конфигурации клиента, даже если макет сети на стороне сервера был изменен или в VPN была добавлена вторичная сеть.

Затем мы запускаем сервер OpenVPN и клиент вторичного офиса и проверяем что файл CCD подхвачен. VPN-клиент во вторичном офисе может пинговать сервер OpenVPN по его VPN IP-адресу как и тестовый пользователь дома.

### Заметка

На данный момент VPN работает, а маршрутизация - нет.

## Начните с середины и двигайтесь наружу

Наиболее эффективный способ устранения неполадок в этой настройке состоит в том, чтобы рассматривать канал VPN как середину, а затем постепенно выполнять пошаговую работу до тех пор, пока все части сети не будут подключены. Во-первых, есть несколько тестов для выполнения на клиенте OpenVPN в дополнительном офисе. Почти для всех тестов достаточно простой команды ping.

Обратите внимание, что нет смысла переходить ко второму тесту, если первый не пройден, и, аналогично к третьему, если второй еще не работает:

- Может ли клиент достичь IP-адреса VPN сервера?  
Это должно функционировать; в противном случае существует проблема с нашим VPN. Это может быть очень ограниченная настройка брандмауэра/iptables на сервере. IP-адрес VPN-сервера должен быть приватным (как правило RFC1918) и, следовательно, не будет маршрутизуемым через общий Интернет.
- Может ли клиент получить доступ к IP-адресу сервера в локальной сети?  
Если это не работает, то, скорее всего, существует правило брандмауэра или iptables, блокирующее доступ. Проверьте входящие правила или попробуйте отключить правила брандмауэра для отладки.
- Может ли клиент достичь IP-адреса шлюза на стороне сервера?  
Если нет, то проверьте ответы на следующие вопросы:
  - На сервере включено перенаправление IP?
  - Существует ли правило брандмауэра/iptables, блокирующее перенаправление доступа к серверу с определенного диапазона IP-адресов?
  - Существует ли на межсетевом шлюзе правило брандмауэра, блокирующее доступ с IP-адресов, не относящихся к локальной сети? (Это было бы хорошей политикой безопасности.) Если это так, то ее необходимо настроить на разрешение трафика, поступающего с VPN IP 10.200.0.0/24.
  - Есть ли обратный маршрут на шлюзе, куда должны возвращаться пакеты, исходящие из VPN? Пакеты с адресом назначения в диапазоне 10.200.0.0/24 следует пересыпать на сервер OpenVPN по IP 172.31.1.2 на маршрутизатор gateway1. Обратите внимание, что это обычно не так. Фактический синтаксис для добавления такого маршрута к шлюзу зависит от модели и встроенного программного обеспечения

используемого маршрутизатора.

- Может ли клиент связаться с другим сервером в локальной сети на стороне сервера? Если нет, то проверьте ответы на следующие вопросы:
    - Имеет ли этот сервер в локальной сети на стороне сервера правильный шлюз в качестве шлюза по умолчанию?
    - Существует ли на сервере правило брандмауэра, блокирующее доступ с IP-адресов, не относящихся к локальной сети? (На самом деле это будет хорошая политика безопасности!)
- Убедившись, что клиент может получить доступ ко всем машинам в локальной сети на стороне сервера - пришло время убедиться что обратное также верно. Убедитесь, что сервер OpenVPN может получить доступ ко всем машинам в локальной сети за вторичным клиентом. Тесты для выполнения очень похожи:
- Может ли сервер достичь IP-адреса VPN клиента?  
Это должно функционировать; в противном случае существует проблема с нашим VPN. Это может быть очень ограниченная настройка брандмауэра/iptables на клиенте. Тем не менее, на данный момент это вряд ли будет проблемой. Но лучше перестраховаться, чем потом жалеть, так что давайте проверим.
  - Может ли сервер получить доступ к IP-адресу локальной сети клиента?  
Если это не работает, то, скорее всего, существует правило брандмауэра/iptables, блокирующее доступ. Проверьте входящие правила.
  - Может ли сервер достичь IP-адреса шлюза на стороне клиента?  
Если нет, то проверьте ответы на следующие вопросы:
    - Включено ли перенаправление траффика на клиенте вторичного офиса?  
Существует ли правило брандмауэра/iptables, блокирующее перенаправление доступа на клиенте из определенного диапазона IP-адресов?
    - Есть ли на клиентском шлюзе правило брандмауэра, блокирующее доступ с IP-адресов не из локальной сети? (Это было бы хорошей политикой безопасности.) Если это так, то ее необходимо отрегулировать так, чтобы трафик приходил с VPN IP 10.200.0.1.
    - Есть ли обратный маршрут на шлюзе во вторичном офисе, чтобы сообщить ему, куда должны возвращаться пакеты, исходящие из VPN? Пакеты с адресом источника 10.200.0.1 должны быть перенаправлены клиенту OpenVPN по IP 192.168.3.17 на маршрутизатор gateway2. Обратите внимание, что это обычно не так. Фактический синтаксис для добавления такого маршрута к шлюзу зависит от модели и встроенного программного обеспечения используемого маршрутизатора. Также обратите внимание, что мы разрешаем проходить только пакетам с самого сервера OpenVPN, так как все остальные клиенты не требуют доступа к его сети.
  - Может ли сервер OpenVPN подключиться к другому компьютеру в локальной сети на стороне клиента?  
Если нет, то проверьте ответы на следующие вопросы:
    - Имеет ли этот сервер в локальной сети на стороне клиента надлежащий шлюз в качестве шлюза по умолчанию?
    - Существует ли на сервере правило брандмауэра, блокирующее доступ с IP-адресов, не относящихся к локальной сети? (На самом деле это будет хорошая политика безопасности!)

На этом этапе клиент OpenVPN в дополнительном офисе должен иметь доступ ко всем машинам в локальной сети на стороне сервера, а сервер OpenVPN в главном офисе должен иметь доступ ко всем машинам в локальной сети на стороне клиента. Есть ещё только один шаг: убедиться, что серверы в локальной сети на стороне сервера могут обращаться к серверам в локальной сети на стороне клиента и наоборот. Опять же, нужно выполнить четыре теста, начиная с компьютера в локальной сети на стороне сервера:

- Может ли эта машина достичь IP-адреса VPN клиента OpenVPN?  
Это должно работать, так как клиент может достучаться до этой машины в результате четвертого теста. Однако лучше перестраховаться, чем сожалеть, так что давайте проверим.
- Может ли эта машина получить доступ к IP-адресу локальной сети клиента?  
Если это не работает, то, скорее всего, существует правило брандмауэра или iptables, блокирующее доступ. Проверьте входящие правила на клиенте OpenVPN.
- Может ли сервер локальной сети достичь IP-адреса шлюза на стороне клиента?  
Если нет, то проверьте ответы на следующие вопросы:
  - Включено ли перенаправление траффика на клиенте вторичного офиса? Существует ли правило брандмауэра/iptables, блокирующее перенаправление доступа на клиенте для определенного диапазона IP-адресов? Обратите внимание, что пакеты, поступающие с компьютера в локальной сети на стороне сервера, будут иметь адрес источника (172.31.1.X), отличный от адреса самого сервера OpenVPN (10.200.0.1).
  - Есть ли на клиентском шлюзе правило брандмауэра, блокирующее доступ с IP-адресов, не относящихся к локальной сети? (Это было бы хорошей политикой безопасности.) Если это так, то ее необходимо настроить, чтобы разрешить трафик, поступающий из диапазона IP-адресов локальной сети 172.31.1.0/24. Точно так же может потребоваться добавить правило брандмауэра на шлюзе на стороне сервера, для разрешения трафика, поступающего из диапазона IP-адресов локальной сети 192.168.3.0/24 на стороне клиента.
  - Есть ли обратный маршрут на шлюзе во вторичном офисе, для сообщения ему куда должны возвращаться пакеты, исходящие из VPN? Пакеты с адресом источника 172.31.1.0/24 должны быть перенаправлены клиенту OpenVPN по IP 192.168.3.17 на маршрутизатор gateway2. Обратите внимание, что это обычно не так.
- Может ли сервер на стороне сервера подключиться к другому компьютеру на стороне клиента?  
Если нет, то проверьте ответы на следующие вопросы:
  - Имеет ли сервер в локальной сети на стороне клиента надлежащий шлюз в качестве шлюза по умолчанию?
  - Существует ли на клиентском компьютере правило брандмауэра, блокирующее доступ с IP-адресов, не относящихся к локальной сети? (На самом деле это будет хорошей политикой безопасности!)

Методично прорабатывая все эти шаги, мы можем решить практически все проблемы маршрутизации. В некоторых случаях могут потребоваться более продвинутые методы отладки. Это может потребовать от нас временно отключить правила брандмауэра, поэтому перед попыткой сделать это следует соблюдать особую осторожность.

## **Найдите время, чтобы временно отключить брандмауэр**

В списках рассылки OpenVPN было слишком много случаев, когда люди не могли заставить маршрутизацию работать, и это оказалось слишком строгим правилом брандмауэра или iptables. Нет необходимости отключать все правила брандмауэра, но если вы застряли на одном из двенадцати шагов, перечисленных ранее, то попробуйте отключить брандмауэр, связанный с устройством, которое вы не можете достичь или с которого вы отправляете трафик.

### **Заметка**

Если вам нужно использовать настройку NATted, убедитесь, что вы не отключаете правила NATting.

## **Если ничего не помогает – используйте tcpdump**

Низкоуровневый сетевой инструмент tcpdump - отличный инструмент для проверки подключения. Для устранения проблем с маршрутизацией мы можем использовать tcpdump, чтобы увидеть, поступает ли какой-либо трафик в конкретный сетевой интерфейс или покидает его, и мы можем проверить адреса источника и назначения этого трафика. На клиенте или сервере Windows может быть проще запустить Wireshark (<https://www.wireshark.org>), который предоставляет аналогичные функции, включая графический интерфейс.

В двенадцати шагах, перечисленных ранее, могут помочь следующие операторы tcpdump:

1. Запустите `tcpdump -nnel -i tun0` на сервере, чтобы увидеть, поступает ли вообще какой-либо трафик через VPN.
2. Запустите `tcpdump -nnel -i eth0` на сервере (где `eth0` - интерфейс локальной сети используемого сервера), чтобы увидеть, поступает ли вообще какой-либо трафик на интерфейс локальной сети. Если нет, то, скорее всего, правило брандмауэра отбрасывает входящий трафик на туннельном интерфейсе.
3. Запустите `tcpdump -nnel -i eth0` на сервере, чтобы проверить, покидает ли трафик интерфейс LAN с помощью следующего:

```
source address = 10.200.0.200
destination address = 172.31.1.254
```

Также проверьте, видим ли мы обратный трафик от серверного шлюза с обратными адресами отправителя и получателя.

4. Снова запустите `tcpdump -nnel -i eth0` на сервере, чтобы проверить, покидает ли трафик интерфейс локальной сети со следующими заголовками пакетов:

```
source address = 10.200.0.200
destination address = 172.31.1.XXX
```

Здесь `172.31.1.XXX` - это IP-адрес компьютера, к которому мы пытаемся подключиться в локальной сети на стороне сервера. Есть ли обратный трафик?

И так далее и так далее для оставшихся шагов!

## **Как оптимизировать производительность с помощью ping и iperf**

Получить максимальную производительность из установки OpenVPN может быть сложно. В чистой сети Ethernet стандартные настройки OpenVPN довольно хороши. Однако в гигабитных сетях требуется некоторая настройка.

Когда используется ADSL или кабельное модемное соединение, производительность также обычно довольно хорошая. Однако, при определенных обстоятельствах производительность нашего туннеля OpenVPN может значительно отставать от производительности обычной сети.

Эти обстоятельства почти всегда зависят от интернет-провайдера, но, тем не менее, стоит изучить как повысить производительность.

Ключом к оптимизации производительности является наличие хороших инструментов для измерения производительности. Два основных, но бесценных инструмента для измерения производительности сети - это `ping` и `iperf`. Инструмент `iperf` легко доступен в Linux, FreeBSD и Mac OS. Есть порты, доступные для Windows и даже Android.

## Использование `ping`

Используя `ping` мы можем определить оптимальный размер MTU для нашей сети. Большинство сетевых операторов сейчас предоставляют своим клиентам MTU для Ethernet размером 1500 байт. Это приводит к полезной нагрузке пакета в 1472 байта. Остальные 28 байт являются издержками TCP/IP для таких вещей, как адрес источника и назначения.

Однако, если между клиентом и сервером существует сеть с более низким MTU, то это может значительно повысить производительность, уменьшив размер пакетов OpenVPN чуть ниже этого размера. Чтобы определить максимальный размер передачи для нашей сети, мы используем следующее:

```
$ ping -M do -s 1472 www.example.org
```

В Windows мы используем следующее:

```
C:\> ping -f -l 1472 www.example.org
```

Эта команда будет отправлять ICMP-пакеты на удаленный сервер по нашему выбору с установленным флагом `not fragment`, инструктируя сетевые маршрутизаторы не разбивать этот пакет на более мелкие. Если между клиентом и сервером есть сеть с меньшим MTU, то команда `ping` завершится неудачно:

```
$ ping -M do -s 1472 www.example.org
PING www.example.org (IP) 1472(1500) bytes of data.
```

```
ping: local error: Message too long, mtu=1480
```

Это говорит нам о том, что производительность будет, скорее всего, лучше, если мы используем либо фрагмент размером 1480, либо размер MTU 1480 байт вместо значения по умолчанию 1500. Обратите внимание, что это не является гарантией - только измерив фактическую производительность VPN, мы узнаем, каково влияние на самом деле.

## Использование `iperf`

Используя `iperf` мы можем измерить производительность сети как внутри, так и вне VPN-туннеля. Это даст нам ценную информацию о том, сколько пропускной способности мы тратим, используя VPN-туннель.

Прежде чем измерять производительность самого VPN-туннеля, всегда пытайтесь измерить производительность нормальной сети. Будет довольно сложно заставить VPN работать лучше чем базовая сеть.

Сначала запустите `iperf` на сервере с помощью следующей команды:

```
$ iperf -s
```

Затем запустите `iperf` на клиенте с помощью следующей команды:

```
$ iperf -c openvpn.example.org
```

В кабельной сети, которая использовалась для тестирования, результат выглядит следующим образом:

```
Mastering OpenVPN
File Edit View Search Terminal Help
$ iperf -c openvpn.example.org

Client connecting to openvpn.example.org, TCP port 5001
TCP window size: 85.0 KByte (default)

[3] local 192.168.3.17 port 43909 connected with <SERVER-IP> port 5001
[ID] Interval Transfer Bandwidth
[3] 0.0-10.4 sec 5.25 MBytes 4.22 Mbits/sec
```

Это на самом деле скорость загрузки используемого кабельного соединения. Теперь мы можем проверить производительность VPN-туннеля в той же сети:

**[ 3] 0.0-10.8 sec 5.25 MBytes 4.09 Mbits/sec**

Повторение измерения дает очень похожие цифры, поэтому справедливо утверждать, что производительность VPN-туннеля на несколько процентов ниже производительности базовой сети. Это на самом деле имеет смысл, так как использование VPN действительно создает некоторые накладные расходы для инкапсуляции, шифрования и аутентификации (подписи) исходных данных. Дальнейшая оптимизация этой сети будет затруднена.

Аналогично, для скорости загрузки используемого кабельного соединения мы обнаруживаем, что производительность VPN-туннеля на несколько процентов ниже:

Производительность базовой сети показана следующим образом:

**[ 4] 0.0-10.6 sec 51.6 MBytes 40.7 Mbits/sec**

Теперь сравните это с VPN-туннелем:

**[ 4] 0.0-10.7 sec 49.5 MBytes 39.0 Mbits/sec**

Опять же, мы видим снижение производительности на 4,5 процента.

Теперь мы можем использовать параметры `fragment` и `mssfix`, чтобы посмотреть сможем ли мы повысить производительность. Там будет немного проб и ошибок для поиска подходящего места для конкретной установки. Неизвестно, какое именно место будет заранее, но метод его определения всегда один и тот же. Теперь добавьте параметры в файлы конфигурации клиента и сервера:

```
fragment X
mssfix
```

Делая это и изменяя X, мы получаем следующие результаты:

X (bytes)	Download (Mbps)	Upload (Mbps)
1200	37.9	3.94
1300	38.1	4.01
1400	38.4	4.04
1472	38.8	4.06
1500	37.6	3.98
<нет>	39.0	4.09

Мы можем заключить, что настройки OpenVPN по умолчанию - самое приятное место для этой сети. Мы могли бы повторить это упражнение, изменив параметр `tun-mtu`, но получили бы тот же результат. Однако рекомендуется сначала настроить производительность с помощью параметра `fragment`, поскольку этот параметр меньше влияет на пересылку пакетов.

## Гигабитная сеть

Теперь мы выполним ту же процедуру в неиспользуемой сети Gigabit Ethernet.

Производительность iperf базовой сети составляет 950 Мбит/с на прием и на отдачу.

Когда мы запускаем сервер OpenVPN с помощью конфигурации `basic-udp-server.conf` и подключаем к нему клиента с помощью файла конфигурации `basic-udp-client.conf`, мы достигаем следующей производительности iperf:

[ ID]	Interval	Transfer	Bandwidth
[ 5]	0.0-10.0 sec	193 MBytes	161 Mbits/sec
[ 4]	0.0-10.0 sec	242 MBytes	203 Mbits/sec

Сейчас наблюдается явное падение производительности. К сожалению, снижение параметра `fragment` нам здесь не поможет. С `fragment 1200` мы достигаем 149 Мбит/с и 115 Мбит/с соответственно.

В высокоскоростных сетях также имеет смысл поэкспериментировать с шифром кодирования. Оба сервера, используемые в этом примере, способны выполнять быстрые инструкции AES благодаря расширению AES-NI, которое присутствует в процессорах (Xeon E5 2620 с тактовой частотой 2 ГГц и Xeon E5 2643 с тактовой частотой 3,5 ГГц, соответственно). Давайте добавим следующее:

```
cipher aes-256-cbc
```

Теперь мы получаем следующий результат:

[ 5]	0.0-10.0 sec	316 MBytes	265 Mbits/sec
[ 4]	0.0-10.0 sec	266 MBytes	223 Mbits/sec

На способном процессоре шифр оказывает большое влияние на производительность. Поскольку OpenVPN является монолитной программой - большое количество ядер не помогает вообще.

Тактовая частота процессора является доминирующим фактором. Подключив ноутбук Core i7 с тактовой частотой 3,8 ГГц к серверу Xeon E5-2643 с частотой 3,5 ГГц, мы получаем гораздо более высокую пропускную способность, используя точно такую же конфигурацию:

[ 5]	0.0-10.0 sec	707 MBytes	593 Mbits/sec
[ 4]	0.0-10.0 sec	529 MBytes	443 Mbits/sec

Таким образом, если вы хотите настроить туннель OpenVPN через высокоскоростную сеть, то лучший совет - использовать высокопроизводительные ЦП, поддерживающие набор инструкций AES-NI. С такой настройкой можно достичь скорости сети более 500 Мбит/с в обоих направлениях.

## Анализ трафика OpenVPN с помощью tcpdump

Низкоуровневый сетевой инструмент `tcpdump` или его аналог с графическим интерфейсом Wireshark являются последним средством для устранения неполадок и производительности сети. В этом разделе мы рассмотрим процесс захвата и анализа зашифрованного сетевого трафика, создаваемого OpenVPN.

Сначала мы настраиваем нашу стандартную сеть OpenVPN, используя конфигурационные файлы `basic-udp`. На клиенте также работает веб-сервер. Мы будем использовать команду `wget` на стороне сервера, чтобы извлечь файл с веб-сервера, чтобы мы могли посмотреть на полученный сетевой трафик.

Мы запускаем `tcpdump` на интерфейсе Ethernet и собираем сетевой трафик, выполняя `wget` вне туннеля:

```
wget -O /dev/null https://CLIENT-IP/test1
```

Результирующий вывод `tcpdump` выглядит следующим образом (для ясности изменен):

Mastering OpenVPN: tcpdump

```

File Edit View Search Terminal Help
tcpdump -nnel -i eth1 tcp port 80
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
17:28:35.499618 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 66: <SERVER-IP>.40310 > <CLIENT-IP>.80
17:28:35.499672 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 66: <CLIENT-IP>.80 > <SERVER-IP>.40310
17:28:35.499877 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 60: <SERVER-IP>.40310 > <CLIENT-IP>.80
17:28:35.499905 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 164: <SERVER-IP>.40310 > <CLIENT-IP>.80
17:28:35.499937 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 54: <CLIENT-IP>.80 > <SERVER-IP>.40310
17:28:35.500616 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 1514: <CLIENT-IP>.80 > <SERVER-IP>.40310
17:28:35.500636 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 1514: <CLIENT-IP>.80 > <SERVER-IP>.40310
17:28:35.500646 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 1514: <CLIENT-IP>.80 > <SERVER-IP>.40310
17:28:35.500656 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 1353: <CLIENT-IP>.80 > <SERVER-IP>.40310
17:28:35.500744 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 54: <CLIENT-IP>.80 > <SERVER-IP>.40310
17:28:35.500844 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 60: <SERVER-IP>.40310 > <CLIENT-IP>.80
17:28:35.500891 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 60: <SERVER-IP>.40310 > <CLIENT-IP>.80
17:28:35.501037 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 60: <SERVER-IP>.40310 > <CLIENT-IP>.80
17:28:35.501057 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 54: <CLIENT-IP>.80 > <SERVER-IP>.40310
^C
13 packets captured
13 packets received by filter
0 packets dropped by kernel

```

Как мы видим, существует 13 пакетов для передачи текстового файла размером 5 КБ. Большинство из этих пакетов были использованы для установки и разрыва соединения, но есть четыре больших пакета, которые были использованы для фактической передачи данных. Первые три из четырех пакетов имеют размер 1514 байт, что является максимальным размером пакета Ethernet.

Далее мы запускаем ту же команду `wget` внутри туннеля. Теперь мы наблюдаем зашифрованный трафик на адаптере Ethernet:

Mastering OpenVPN: tcpdump

```

File Edit View Search Terminal Help
tcpdump -nnel -i eth1 udp port 1194
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
17:23:44.224950 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 95: <SERVER-IP>.1194 > <CLIENT-IP>.48693: UDP, length 53
17:23:44.225061 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 95: <CLIENT-IP>.48693 > <SERVER-IP>.1194: UDP, length 53

17:23:46.798604 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 135: <SERVER-IP>.1194 > <CLIENT-IP>.48693: UDP, length 93
17:23:46.798792 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 135: <CLIENT-IP>.48693 > <SERVER-IP>.1194: UDP, length 93
17:23:46.799099 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 119: <SERVER-IP>.1194 > <CLIENT-IP>.48693: UDP, length 77
17:23:46.799112 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 231: <SERVER-IP>.1194 > <CLIENT-IP>.48693: UDP, length 189
17:23:46.799238 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 119: <CLIENT-IP>.48693 > <SERVER-IP>.1194: UDP, length 77
17:23:46.800221 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 1487: <CLIENT-IP>.48693 > <SERVER-IP>.1194: UDP, length 1445
17:23:46.800316 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 1487: <CLIENT-IP>.48693 > <SERVER-IP>.1194: UDP, length 1445
17:23:46.800403 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 1487: <CLIENT-IP>.48693 > <SERVER-IP>.1194: UDP, length 1445
17:23:46.800492 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 1487: <CLIENT-IP>.48693 > <SERVER-IP>.1194: UDP, length 1445
17:23:46.800539 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 319: <CLIENT-IP>.48693 > <SERVER-IP>.1194: UDP, length 277
17:23:46.800580 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 119: <CLIENT-IP>.48693 > <SERVER-IP>.1194: UDP, length 77
17:23:46.800864 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 119: <SERVER-IP>.1194 > <CLIENT-IP>.48693: UDP, length 77
17:23:46.800886 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 119: <SERVER-IP>.1194 > <CLIENT-IP>.48693: UDP, length 77
17:23:46.800890 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 119: <SERVER-IP>.1194 > <CLIENT-IP>.48693: UDP, length 77
17:23:46.800895 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 119: <SERVER-IP>.1194 > <CLIENT-IP>.48693: UDP, length 77
17:23:46.800899 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 119: <SERVER-IP>.1194 > <CLIENT-IP>.48693: UDP, length 77
17:23:46.800903 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 119: <SERVER-IP>.1194 > <CLIENT-IP>.48693: UDP, length 77
17:23:46.801122 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 119: <CLIENT-IP>.48693 > <SERVER-IP>.1194: UDP, length 77

17:23:56.410192 2:2:2:2:2:2 > 1:1:1:1:1:1, IPv4, length 95: <CLIENT-IP>.48693 > <SERVER-IP>.1194: UDP, length 53
17:23:56.410540 1:1:1:1:1:1 > 2:2:2:2:2:2, IPv4, length 95: <SERVER-IP>.1194 > <CLIENT-IP>.48693: UDP, length 53
^C
22 packets captured
22 packets received by filter
0 packets dropped by kernel

```

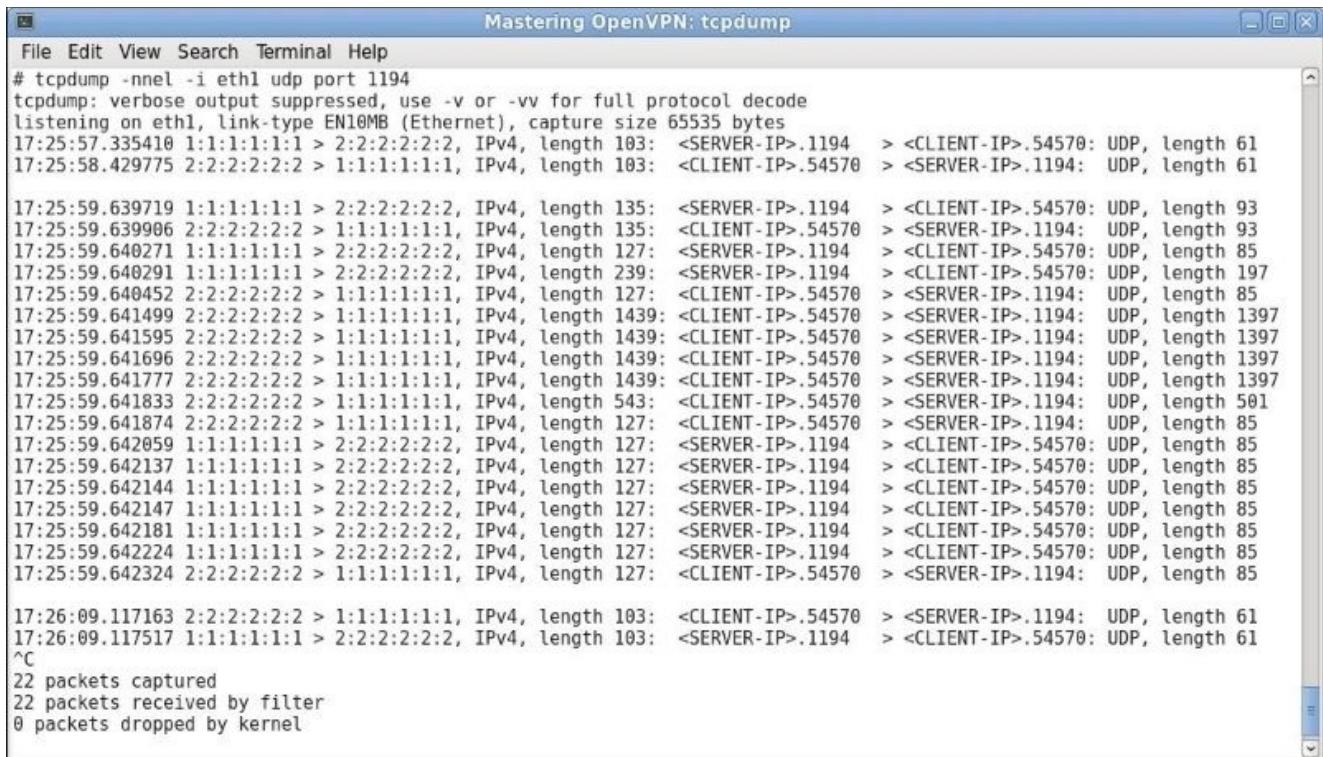
Здесь мы видим 22 захваченных пакета. Первый и последний два пакета являются heartbeat пакетами OpenVPN и могут игнорироваться. Оставшиеся 18 пакетов являются зашифрованным эквивалентом пакетов, показанных в первом выводе `tcpdump`. Как мы видим здесь, длина пакета немного меньше, и особенно payload каждого пакета немного меньше: payload самого большого пакета UDP составляет 1445 байтов. Эти 1445 байт содержат зашифрованные и

подписанные данные из команды `wget`. В нашей настройке мы не указали параметр `fragment` – это означает, что OpenVPN 2.3 по умолчанию будет иметь внутреннюю фрагментацию 1450 байт.

Общий размер каждого пакета никогда не превышает 1487 байтов, что довольно близко к оптимальному: обычно пакеты не должны превышать размер MTU, составляющий 1500 байт.

Этот дамп экрана `tcpdump` также показывает что фрагментации не происходит, кроме как внутри OpenVPN. Это хорошо, поскольку мы хотим избежать фрагментации пакетов операционной системой или сетью для максимальной производительности. Если бы мы видели здесь фрагментацию пакетов, то это было бы отличным признаком того, что нам нужно было добавить дополнительную фрагментацию в нашу конфигурацию OpenVPN.

Давайте посмотрим, что произойдет, если мы добавим `fragment 1400` в нашу настройку. Мы перезапускаем сервер и клиент и снова запускаем команду `wget`:



```
Mastering OpenVPN: tcpdump
File Edit View Search Terminal Help
tcpdump -nnel -i eth1 udp port 1194
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
17:25:57.335410 1:1:1:1:1 > 2:2:2:2:2, IPv4, length 103: <SERVER-IP>.1194 > <CLIENT-IP>.54570: UDP, length 61
17:25:58.429775 2:2:2:2:2 > 1:1:1:1:1, IPv4, length 103: <CLIENT-IP>.54570 > <SERVER-IP>.1194: UDP, length 61

17:25:59.639719 1:1:1:1:1 > 2:2:2:2:2, IPv4, length 135: <SERVER-IP>.1194 > <CLIENT-IP>.54570: UDP, length 93
17:25:59.639906 2:2:2:2:2 > 1:1:1:1:1, IPv4, length 135: <CLIENT-IP>.54570 > <SERVER-IP>.1194: UDP, length 93
17:25:59.640271 1:1:1:1:1 > 2:2:2:2:2, IPv4, length 127: <SERVER-IP>.1194 > <CLIENT-IP>.54570: UDP, length 85
17:25:59.640291 1:1:1:1:1 > 2:2:2:2:2, IPv4, length 239: <SERVER-IP>.1194 > <CLIENT-IP>.54570: UDP, length 197
17:25:59.640452 2:2:2:2:2 > 1:1:1:1:1, IPv4, length 127: <CLIENT-IP>.54570 > <SERVER-IP>.1194: UDP, length 85
17:25:59.641499 2:2:2:2:2 > 1:1:1:1:1, IPv4, length 1439: <CLIENT-IP>.54570 > <SERVER-IP>.1194: UDP, length 1397
17:25:59.641595 2:2:2:2:2 > 1:1:1:1:1, IPv4, length 1439: <CLIENT-IP>.54570 > <SERVER-IP>.1194: UDP, length 1397
17:25:59.641696 2:2:2:2:2 > 1:1:1:1:1, IPv4, length 1439: <CLIENT-IP>.54570 > <SERVER-IP>.1194: UDP, length 1397
17:25:59.641777 2:2:2:2:2 > 1:1:1:1:1, IPv4, length 1439: <CLIENT-IP>.54570 > <SERVER-IP>.1194: UDP, length 1397
17:25:59.641833 2:2:2:2:2 > 1:1:1:1:1, IPv4, length 543: <CLIENT-IP>.54570 > <SERVER-IP>.1194: UDP, length 501
17:25:59.641874 2:2:2:2:2 > 1:1:1:1:1, IPv4, length 127: <CLIENT-IP>.54570 > <SERVER-IP>.1194: UDP, length 85
17:25:59.642059 1:1:1:1:1 > 2:2:2:2:2, IPv4, length 127: <SERVER-IP>.1194 > <CLIENT-IP>.54570: UDP, length 85
17:25:59.642137 1:1:1:1:1 > 2:2:2:2:2, IPv4, length 127: <SERVER-IP>.1194 > <CLIENT-IP>.54570: UDP, length 85
17:25:59.642144 1:1:1:1:1 > 2:2:2:2:2, IPv4, length 127: <SERVER-IP>.1194 > <CLIENT-IP>.54570: UDP, length 85
17:25:59.642147 1:1:1:1:1 > 2:2:2:2:2, IPv4, length 127: <SERVER-IP>.1194 > <CLIENT-IP>.54570: UDP, length 85
17:25:59.642181 1:1:1:1:1 > 2:2:2:2:2, IPv4, length 127: <SERVER-IP>.1194 > <CLIENT-IP>.54570: UDP, length 85
17:25:59.642224 1:1:1:1:1 > 2:2:2:2:2, IPv4, length 127: <SERVER-IP>.1194 > <CLIENT-IP>.54570: UDP, length 85
17:25:59.642324 2:2:2:2:2 > 1:1:1:1:1, IPv4, length 127: <CLIENT-IP>.54570 > <SERVER-IP>.1194: UDP, length 85

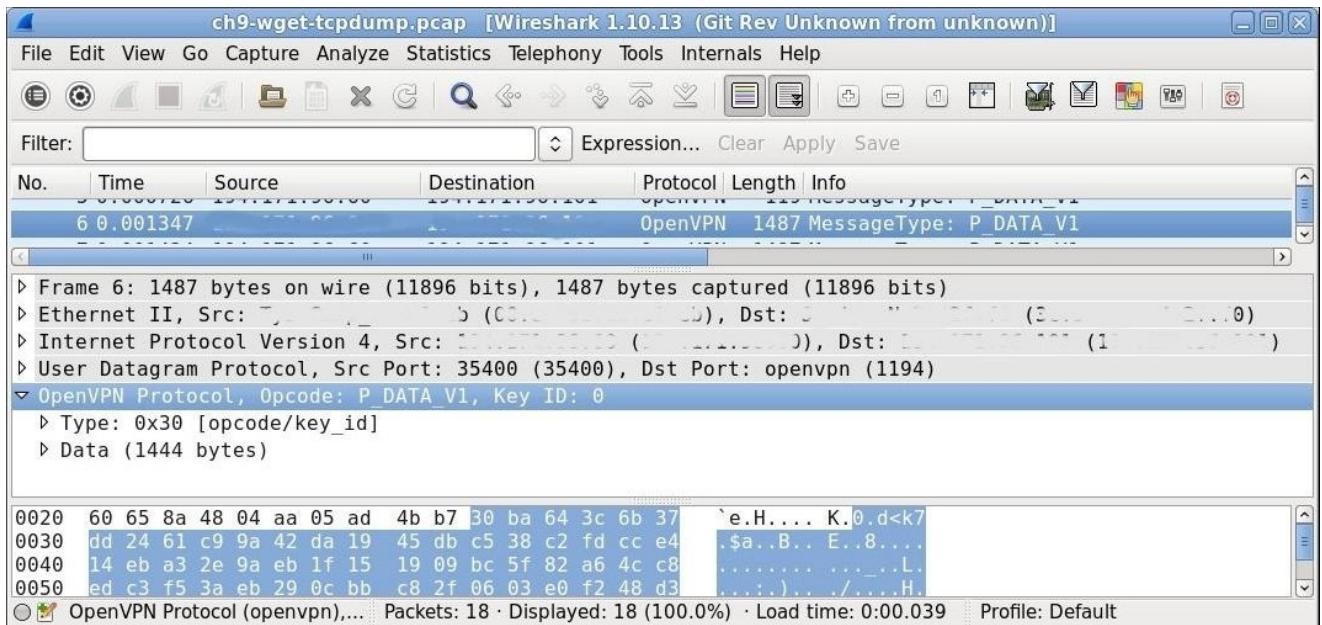
17:26:09.117163 2:2:2:2:2 > 1:1:1:1:1, IPv4, length 103: <CLIENT-IP>.54570 > <SERVER-IP>.1194: UDP, length 61
17:26:09.117517 1:1:1:1:1 > 2:2:2:2:2, IPv4, length 103: <SERVER-IP>.1194 > <CLIENT-IP>.54570: UDP, length 61
^C
22 packets captured
22 packets received by filter
0 packets dropped by kernel
```

С добавленным в нашу настройку `fragment 1400` мы можем видеть в выводе `tcpdump` что полезная нагрузка пакета теперь составляет 1397 байт, что очень близко к пределу 1400. Мы также видим, что теперь требуется больше пакетов для передачи текстового файла размером 5 КБ по туннелю, что означает снижение производительности. Из этого снимка экрана мы можем сделать вывод, что нам следует снова удалить параметр.

Из приведенного и предыдущего скриншота мы также можем сделать вывод, что каждый пакет OpenVPN несет 42-байтовые издержки. Эти издержки частично способствуют накладным расходам, возникающим при использовании любого решения VPN. Он включает в себя всю служебную информацию, поскольку все сетевые пакеты должны содержать служебную информацию об адресе источника, адресе назначения, типе пакета, контрольные суммы, флаги и многое другое.

Наконец, давайте посмотрим на содержимое реального зашифрованного пакета OpenVPN. Для этого удобно использовать инструмент Wireshark (<https://www.wireshark.org>). Wireshark в основном предоставляет графический интерфейс поверх низкоуровневого инструмента `tcpdump`. Он может декодировать содержимое большинства типов сетевого трафика, как мы

можем видеть на следующем скриншоте (он сделан анонимным по соображениям конфиденциальности):



Этот скриншот говорит нам следующее:

- Фактический размер пакета составляет 1487 байт.
- Он содержит заголовки Ethernet и IPv4, как и любой сетевой пакет в сети Ethernet.
- Это пакет OpenVPN с исходным портом 35400 и портом назначения 1194 - это означает, что он перемещается от клиента к серверу. На самом деле это один из зашифрованных пакетов при передаче файла размером 5 КБ с клиента на сервер.
- Полезной нагрузкой пакета является пакет данных OpenVPN (формат версии 1) с размером полезной нагрузки 1487 байт. Обратите внимание, что tcpdump сообщил о 1488 байтах ранее, но Wireshark может декодировать полезную нагрузку и увидеть, что первый байт является кодом операции OpenVPN.

Этот пакет будет получен OpenVPN, проверен на аутентификацию, расшифрован и распакован (если мы указали). Полученный незашифрованный пакет затем перенаправляется в таблицы маршрутизации операционной системы, которые решают, куда направить пакет. В нашем случае пакет останется на сервере и будет передан процессу wget.

## Резюме

В этой главе вы познакомились с некоторыми основными приемами устранения неполадок и настройки OpenVPN. Вы также получили представление о чтении файлов журналов клиента и сервера. Вы узнали как обнаружить и исправить некоторые из наиболее часто совершаемых ошибок. Большинство вопросов в списке рассылки OpenVPN касаются проблем маршрутизации - поэтому мы обсудили обнаружение и устранение проблем маршрутизации. Наконец, существует большая разница между работающей установкой и хорошо работающей установкой, поэтому мы рассмотрели примеры того, как обнаруживать и решать проблемы с производительностью.

Конечно, OpenVPN не идеален и поэтому ваша нерабочая настройка также может быть вызвана ошибкой в самом OpenVPN. Существует несколько каналов для сообщения об ошибках, включая список электронной почты ([openvpn-users@lists.sourceforge.net](mailto:openvpn-users@lists.sourceforge.net)), канал IRC (#openvpn на freenode.net IRC) и веб-сайт форума (<https://forums.openvpn.net>). Вы также можете отправлять запросы на функции или списки пожеланий на эти каналы, некоторые из которых могут

появиться в будущей версии OpenVPN.

В следующей главе вы узнаете, что нового можно ожидать в будущих выпусках OpenVPN. Вы также узнаете, какие в настоящее время известны проблемы с базой кодов OpenVPN, и узнаете о планах по устранению этих проблем.

# Глава 10. Будущие направления

История OpenVPN была ухабистой - от неопытного новичка до широко используемого, почти мертвого и обратно. Разрыв в разработке с 2006 по примерно 2009 год был значительным, но тяжелая работа и преданность разработчиков, таких как Дэвид Соммерсет (dazo), Герт Доеринг (cron2), Штеффан Каргер и Самули Сеппанен, дали проекту недавнее успешное прошлое и светлое будущее.

OpenVPN доступен практически на всех доступных платформах. Snom (производитель IP-телефонов), например, включает в себя версию прошивки своего VOIP-телефона с включенным клиентом OpenVPN. pfSense, OpenWRT и другие WAP/брандмауэры операционных систем включают OpenVPN и (обычно) веб-интерфейс для управления развертыванием.

В последние годы OpenVPN был доступен на многих мобильных телефонах. Первый был доступен для Android, *OpenVPN для Android* от Арне Швабе. При этом использовался перенесенный драйвер tun и он не поддерживал VPN-режим в режиме tap (мост).

Однако приложение OpenVPN для iOS (Apple) появилось гораздо позже. OpenVPN Technologies Inc. потребовался почти год, чтобы договориться с Apple о поддержке внешнего API VPN и предоставлении доступа к этому API для его использования. На Android исходный код OpenVPN может быть перенесен на платформу, с некоторыми упщениями для tap, так как драйвер tap не был доступен на платформе. Из-за среды разработки на iOS клиент должен был быть написан с нуля. Джеймс Йонен написал за несколько месяцев почти полностью функциональный клиент на C++ и *OpenVPN Connect* был опубликован в App Store®.

## Сильные стороны

OpenVPN 2.3 опережает то, на чем застряли в недавнем прошлом. Беглый взгляд на список изменений с 2008 года по настоящее время показывает довольно значительное количество важных обновлений, исправлений и улучшений.

Есть несколько вещей, которые OpenVPN делает хорошо. OpenVPN расширяемый, подключаемый и динамичный. Он имеет функциональную поддержку IPv6, передачу шлюза по умолчанию (даже в неработающих сетях) и переключение по общедоступным IP-адресам с сохранением соединений.

Кроссплатформенная поддержка непревзойденна для различных реализаций VPN.

## Текущие недостатки

Есть заметные недостатки в текущей версии OpenVPN. Во-первых, все приложение написано как одно монолитное. Тот же двоичный файл, используемый для клиентских подключений, также используется в качестве экземпляра сервера. Это не слишком большая проблема, но здесь нет модульности кода, поэтому вся логика должна обрабатываться независимо от контекста, в котором выполняется приложение.

Работая над проблемами монолитного проектирования, разработчикам будет проще реализовать такие функции, как IPv6, дополнительные алгоритмы сжатия и т.д. Кроме того, изменения для улучшения сетевого стека необходимо обновлять во многих местах кода, а не в отдельной библиотеке или компоненте. По этой причине сегодня стеки IPv6 и IPv4 обрабатываются отдельно.

## Масштабирование на гигабитных скоростях и выше

Как правило, на современном оборудовании OpenVPN может поддерживать пару сотен

клиентских подключений до того, как ограничения ядра снизят производительность до неблагоприятных уровней. Этот лимит не был проблемой до недавнего времени, пока не стали доступны высокоскоростные интернет-соединения. В прошлом один сервер OpenVPN с хорошим каналом связи мог легко идти в ногу со многими клиентскими подключениями по обычному домашнему интернет-соединению.

Сегодня, однако, гигабитные соединения в домашних условиях не редкость, и даже там, где они сейчас недоступны, эти высокоскоростные каналы связи будут доступны в самом ближайшем будущем. Благодаря подходящему высокоскоростному процессору OpenVPN способен (почти) насыщать гигабитный канал Ethernet.

Для этого требуются инструкции шифрования AES (известные как AES-NI), имеющиеся в современных процессорах, таких как процессоры Intel Core i7 и Xeon E5, а также в современных процессорах AMD. Также требуется, чтобы метод шифрования был установлен на AES, например, с помощью `--cipher aes-256-cbc` как в конфигурации сервера, так и клиента.

### Заметка

Невозможно передать метод шифрования с сервера на клиент. Это ограничение нынешнего проектирования OpenVPN, и мы надеемся что оно будет исправлено в следующей версии.

Здесь также играют роль операционная система и библиотека шифрования. Большинство настроек сервера используют библиотеки OpenSSL для шифрования и дешифрования.

Поддержка набора команд AES-NI была включена только в OpenSSL 1.0.0. Например, CentOS 5 по-прежнему использует библиотеку OpenSSL (0.9.8e-fips), не поддерживающую эти инструкции. Достаточно легко проверить, используют ли процессор и операционная система инструкции AES-NI. Используя команду `openssl speed` вы можете быстро определить производительность кодирования как для шифра OpenVPN по умолчанию (BlowFish или `bf-cbc`), так и для шифра AES (`aes-256-cbc`):

```
$ openssl speed -evp bf-cbc
[...]
type ... 256 bytes 1024 bytes 8192 bytes
bf-cbc ... 137977.26k 138565.97k 137470.47k

$ openssl speed -evp aes-256-cbc
type ... 256 bytes 1024 bytes 8192 bytes
aes-256-cbc 566760.53k 588199.94k 591250.12k
```

Этот тест проводился на процессоре Intel Core i7-4810MQ под управлением Fedora 20, и очевидно, что AES намного быстрее чем BlowFish. Мы можем с уверенностью заключить, что AES-NI поддерживается как процессором, так и операционной системой. Если мы отключим поддержку OpenSSL для инструкций AES-NI, влияние на производительность будет весьма существенным:

```
$ OPENSSL_ia32cap=0 openssl speed -evp aes-256-cbc
type ... 256 bytes 1024 bytes 8192 bytes
aes-256-cbc 120009.39k 264001.19k 262821.68k
```

Используя процессор, такой как Core i7, можно достичь производительности более 500 Мбит в обоих направлениях, как было показано в [Главе 9, Устранение неполадок и настройка](#).

Дополнительные улучшения могут быть получены с использованием `--mssfix`, `--tun-mtu` и `--fragment`. При совместном использовании может быть достигнуто увеличение скорости до 400 процентов.

Другие факторы могут способствовать проблемам с производительностью вне OpenVPN. Масштабирование сверх гигабитных скоростей потребует обширной модернизации OpenVPN, поскольку требует совершенно другого подхода к обработке таких высоких уровней трафика.

Имейте в виду, что сетевой трафик обычно обрабатывается кусками по 1500 байт (обозначается как **Maximum Transmission Unit (MTU)**). Для гигабитного канала это означает, что ядру операционной системы и процессу OpenVPN необходимо обрабатывать примерно 80 000 пакетов в секунду. На 10-гигабитном канале это число возрастает до 800 000 пакетов, с которыми даже самые современные процессоры не так легко справляются. Увеличение значения MTU с 1500 до 9000 байтов также известного как Jumbo-кадры, уменьшает количество пакетов, не уменьшая при этом пропускную способность. Jumbo-кадры должны поддерживаться всеми узлами в сети, иначе может возникнуть фрагментация пакетов.

## Куда идем

Начиная с 2010 года, разработчики открытого исходного кода начали дискуссии о способах улучшения процесса сервера OpenVPN и повышения эффективности. Был определен ряд областей, которые можно улучшить. К счастью, начало этой работы было завершено переписыванием клиентского кода Джеймса для приложения iOS. Официальные дорожные карты для предстоящих v2.4 и будущих версий v3.0 можно найти в вики сообщества OpenVPN в следующих местах:

- <https://community.openvpn.net/openvpn/wiki/OpenVPN2.4>
- <https://community.openvpn.net/openvpn/wiki/RoadMap>

Более конкретно обсуждалась модульность для плагинов, даже создание модулей поддержки OpenSSL и PolarSSL. Это позволит упростить интеграцию других библиотек по мере их появления, и даже с помощью этого подхода можно добиться поддержки чего-то совершенно отличного от SSL. Также рассматривается улучшение потоков и разгрузка процессов для улучшения объема клиентского соединения и использования полосы пропускания.

Несмотря на большие успехи, которые мы уже сделали, есть много возможностей для улучшения. Ключевой вопрос, решения которого не видно - поддержка команды разработчиков. Есть только очень небольшое количество разработчиков, активных и преданных проекту. Конечным результатом является медленный цикл разработки, а новые функции встречаются редко.

Некоторые элементы, над которыми в настоящее время работают, включают улучшенную поддержку IPv6, правильное разделение привилегий Windows и роуминг TLS.

Полный список текущих ошибок доступен на трекере ошибок сообщества OpenVPN. Патчи всегда приветствуются в списке рассылки, и хорошо написанные и протестированные патчи обязательно получат быстрое одобрение. Ссылка <https://community.openvpn.net/openvpn/report/1> приведет вас непосредственно к открытым отчетам об ошибках.

## Улучшение поддержки сжатия

Начиная с версии 2.4, OpenVPN будет поддерживать различные механизмы сжатия VPN-трафика. В настоящее время поддерживается только сжатие LZ02, но в версии 2.4 вы также можете компилировать с поддержкой алгоритмов сжатия Snappy и LZ4. Это может немного улучшить производительность в зависимости от типа трафика, проходящего через VPN. Обычный веб-трафик значительно повысит производительность, в то время как в трудносжимаемом трафике, таком как изображения или видеофайлы, вероятно, произойдет небольшое снижение производительности из-за дополнительных затрат на сжатие и распаковку каждого пакета.

## Сжатие на клиенте

В версии 2.3 и ниже, если сжатие включено на сервере, оно также должно быть включено на клиенте. В прошлом это было трудно идентифицировать по клиентским журналам, так как

туннель просто не пропускал трафик. В дорожной карте v2.4 предусмотрено согласование сжатия. Это позволило бы сжатие для каждого клиента и даже согласование протокола/алгоритма сжатия.

## Новые криптографические процедуры

В версии 2.4 поддержка алгоритмов аутентификации эллиптической кривой включена впервые. Пока невозможно использовать эллиптические кривые для всего трафика, но это позволяет использовать сертификаты на основе ECDSA.

Надеемся, мы также увидим поддержку шифрования на основе GCM в версии 2.4. Шифрование **GCM (Galois/Counter Mode)** более эффективно и производительно, чем используемые в настоящее время процедуры шифрования **CBC (Cipher Block Chain)**.

**Шифрование с проверкой подлинности с помощью связанных данных (AEAD)** также дебютирует в версии 2.4.

## Смешанная аутентификация сертификата/имени пользователя

В настоящее время OpenVPN поддерживает аутентификацию с использованием сертификатов и/или имени пользователя и пароля, но оба варианта невозможны. Опция `--client-cert-not-required` фактически отключает проверку сертификата.

В версии 2.4 станет возможной поддержка клиентов, подключаемых с использованием либо сертификата, либо имени пользователя и пароля, либо обоих. Это обеспечивает большую гибкость при предоставлении пользователям разных уровней доступа к настройке VPN. Для этого добавлена новая опция.

- `verify-client-cert none` : это фактически то же самое, что `--client-cert-not-required`.
- `verify-client-cert optional` : проверит сертификат, предоставленный клиентом, но не отклонит соединение, если проверка не удалась.
- `verify-client-cert require` : проверит сертификат, предоставленный клиентом и отклонит соединение, если проверка не удалась. Это будет настройка по умолчанию, так как она по умолчанию используется в OpenVPN версии 2.3 и более ранних.

Параметр `--client-cert-not-required` будет объявлен устаревшим в ближайшем будущем и упоминается в дорожной карте v3.0.

## Поддержка IPv6

Сетевой код в OpenVPN использует отдельные функции для путей кода IPv4 и IPv6. Пару лет назад произошел серьезный пересмотр работы с IPv4, но работа над функциями IPv6 так и не была выполнена. OpenVPN 2.3 поддерживает полностью собственный транспорт IPv6, а также инкапсулированный трафик. Использование DNS-серверов IPv6 и получение этой информации от DHCP не поддерживается, но включено в план для v2.4.

`push "redirect-gateway ipv6"` также есть в списке. Вы все еще можете имитировать маршрут по умолчанию с IPv6, передавая специальные маршруты вручную:

```
push "route-ipv6 ::/0 2600:dead:beef::1"
```

## Разделение привилегий Windows

OpenVPN в настоящее время требует административных привилегий на всех клиентских рабочих станциях. Пользователи с автономными рабочими станциями должны иметь

возможность обновлять конфигурацию без прав администратора, а клиентское приложение должно иметь возможность принимать действительные утверждения конфигурации сервера.

Два подхода были представлены для достижения этой цели. Один из них ориентирован на Windows, а другой предлагает принципы, которые могут быть развернуты или реализованы на других платформах.

Первый из двух достигается путем предоставления интерактивной службы OpenVPN. Хайко Хунд впервые предложил такой подход в феврале 2012 года (<http://thread.gmane.org/gmane.network.openvpn.devel/5685/focus=5728>). Эта концепция включает в себя централизованную службу, действующую как оболочка вокруг другого процесса OpenVPN. Будет установлено клиентское соединение от интерактивного пользователя OpenVPN или OpenVPN GUI и этот процесс будет подключаться к этой службе. Затем служба будет принимать аргументы от клиента и создавать актуальный VPN-туннель, устанавливать маршруты и другие параметры.

При правильном внедрении есть требования, которые ранее не были реализованы или не рассматривались. Во-первых, закрытый ключ должен быть должным образом защищен:

*Чтобы быть полным, оболочка [=interactive service] также должна владеть закрытым ключом OpenVPN – в противном случае конфигурация будет копироваться непrivилегированным пользователем, что должно быть предотвращено корпоративной моделью. Защита приватного ключа может быть выполнена путем хранения ключа в системном хранилище сертификатов/ключей и доступа к ключу через API криптографического поставщика, например Crypto API в Windows, PKCS#11 в Linux или Keychain на Mac.*

-- Джеймс Йонан

Во-вторых, интерактивный сервис не должен разрешать доступ другим процессам (не OpenVPN) работать как тот же (текущий) пользователь:

*Канал/сокет для привилегированного процесса [=interactive service] должен быть управляем доступом, чтобы его мог использовать только openvpn. Вы не должны получить уязвимость эскалации привилегий, когда операции, которые обычно были бы привилегированными (например, изменение маршрута по умолчанию), теперь могут быть выполнены любым процессом в пользовательском пространстве, просто используя канал OpenVPN/сокет.*

-- Джеймс Йонан

В-третьих:

*Другие непривилегированные программы могут получить доступ к API-интерфейсам для этих оболочек [=interactive service], например, путем ввода маршрутов в API. Вредоносные программы, которые обычно были бы ограничены пользовательским пространством, теперь могут выполнять привилегированные операции, такие как изменение маршрута по умолчанию. Теперь конечный пользователь может подключиться к любому VPN-серверу по своему выбору (серьезное нарушение корпоративной модели). То, что вы фактически сделали с этой моделью - это ввели уязвимость эскалации привилегий, потому что операции, которые обычно требуют привилегий, такие как добавление маршрутов, теперь могут выполняться непривилегированным пользователем.*

-- Джеймс Йонан

Второй подход использует два или три отдельных объекта СОМ+, как предложено Алоном Бар-

Левом в марте 2012 года (<http://thread.gmane.org/gmane.network.openvpn.devel/5755/focus=5869>). При таком подходе необходимы три компонента: OpenVPN GUI, служба OpenVPN и сетевой обработчик OpenVPN.

OpenVPN GUI не сильно изменится по сравнению с тем, что есть сегодня. Он будет продолжать передавать задачи и обновления в бэкэнд-процесс. Поскольку нет текущего разделения привилегий, текущий GUI не должен обрабатывать какую-либо авторизацию. С использованием COM+ и сетевого модуля OpenVPN графический интерфейс пользователя может быть полностью непривилегированным.

## Резюме

После нескольких лет работы в IRC-канале OpenVPN и на форуме поддержки OpenVPN среди пользователей сервера администрирования возникают некоторые постоянные трудности: базовые сети и маршрутизация, управление сертификатами X.509 и аутентификация пользователей или клиентов. Прочитав эту книгу, вы должны хорошо понимать эти концепции и основные механизмы. Различия между виртуальными сетевыми адаптерами tun и tap также обсуждались.

OpenVPN - очень активный проект с открытым исходным кодом, который постоянно развивается. Методы и примеры в рамках освоения OpenVPN, скорее всего, не устареют в ближайшем будущем. Однако в коде возможны недостатки, поэтому мы настоятельно рекомендуем вам прочитать руководство (man page), доступное по адресу <https://openvpn.net/community-resources/#articles>.

Как и большинству проектов с открытым исходным кодом, OpenVPN нуждается в большей помощи - требуется больше добровольцев для помощи модерирования форума и помощи по IRC, а также дополнительные разработчики, помогающие увеличить скорость разработки. Есть стремление создать систему вознаграждений за помощь в этих усилиях. Сообщество сильное и протокол широко известен.

Предстоит проделать большую работу, но набор функций OpenVPN по сравнению с другими приложениями VPN ставит его в соответствие с ожиданиями. Если вы хотите принять участие в проекте OpenVPN - просмотрите следующие ресурсы чтобы найти интересную работу, и обратитесь к кому-нибудь, чтобы помогли вам начать:

- IRC: [#openvpn и #openvpn-devel](https://freenode.net)
- Веб-форум: <https://forums.openvpn.net>
- Список рассылки: <https://sourceforge.net/p/openvpn/mailman/>
- Трекер ошибок: <https://community.openvpn.net/openvpn/report/1>
- Man page: <https://openvpn.net/community-resources/#articles>