РЕАЛИЗАЦИЯ ВИРТУАЛЬНОЙ ЧАСТНОЙ СЕТИ УДАЛЕННОГО ДОСТУПА

Мисюль В.С., студент

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Ярмолик В.Н. – д-р техн. наук, профессор

В работе рассматривается реализация VPN удаленного доступа на базе WireGuard для корпоративной среды. Предлагается расширить протокол механизмом crypto agility, позволяющим заменять криптографические алгоритмы через плагины без изменения основной логики туннеля. Дополнительно исследуются способы повышения пропускной способности, включая распараллеливание обработки трафика и распределение входящих UDP-пакетов между несколькими сокетами с использованием. Цель работы – создать защищенный, производительный VPN, который можно адаптировать под требования конкретной организации.

Удаленный доступ в корпоративной сети требует одновременно высокой скорости, устойчивой криптографической защиты и гибкости в выборе используемых алгоритмов. WireGuard хорошо подходит для такой задачи благодаря простой модели peer-to-peer, работе на уровне сетевого слоя и небольшой поверхности атаки.

Для работы WireGuard создается виртуальный сетевой интерфейс, на который настраивается маршрутизация трафика: пакеты, адресованные определённым IP-префиксам, направляются в этот интерфейс для последующей передачи через VPN. Пользовательское приложение взаимодействует с этим интерфейсом, получая исходящие IP-пакеты и передавая обратно входящие после их обработки.

В основе протокола лежит связка публичного ключа узла и допустимых IP-адресов, что упрощает контроль трафика и делает поведение туннеля предсказуемым. Для каждого узла заранее задается его публичный ключ и список IP-префиксов, которые считаются за ним закреплёнными.

При отправке пакета система сначала определяет, какому узлу он должен быть доставлен, на основе адреса назначения. Затем выбирается соответствующий публичный ключ, и пакет инкапсулируется и шифруется с использованием сессионных ключей, ассоциированных с ним.

При приеме пакета выполняется обратная проверка. После успешной дешифрации определяется, какому публичному ключу соответствует данный пакет. Далее извлекается внутренний IP-пакет, и его исходный адрес проверяется на принадлежность списку IP-адресов закрепленных для этого узла. Если адрес не входит в разрешённый диапазон, пакет отбрасывается. Таким образом предотвращается подмена адреса внутри туннеля.

Передача трафика осуществляется поверх UDP. Полученный из виртуального интерфейса IP-пакет шифруется с использованием сессионных ключей, после чего инкапсулируется в UDP-дейтаграмму и отправляется на сетевой адрес удалённого узла, указанный в конфигурации. На принимающей стороне выполняется обратная операция: UDP-пакет принимается, из него извлекается зашифрованная полезная нагрузка, производится дешифрация, и восстановленный IP-пакет передаётся обратно в виртуальный интерфейс.

Использование UDP в качестве транспортного протокола обусловлено рядом архитектурных причин. В отличие от TCP, UDP не реализует механизмы установления соединения, подтверждения доставки, упорядочивания пакетов и контроля перегрузки. Это позволяет не дублировать функциональность, уже реализованную в вышележащих протоколах (в первую очередь TCP), трафик которых передаётся через VPN. В противном случае возникала бы проблема TCP-over-TCP, приводящая к ухудшению производительности из-за конкурирующих механизмов контроля перегрузки и повторной передачи.

Отсутствие состояния соединения на уровне транспортного протокола упрощает обработку пакетов. Каждая UDP-дейтаграмма обрабатывается независимо, что снижает требования к хранению состояния, уменьшает накладные расходы и позволяет минимизировать задержки. Это особенно важно для VPN, где каждый дополнительный этап обработки напрямую влияет на общую латентность.

UDP также лучше подходит для работы в условиях NAT. Поскольку большинство сетей используют трансляцию адресов, важно поддерживать актуальность соответствий (NAT bindings). В случае UDP это достигается за счёт периодической отправки keepalive-пакетов, что позволяет сохранять открытым маршрут между узлами без необходимости поддержания полноценного соединения. Кроме того, отсутствие жёсткой привязки к состоянию соединения упрощает смену внешнего адреса или порта (например, при перемещении клиента между сетями).

Наконец, модель UDP упрощает реализацию параллельной обработки. Так как пакеты независимы друг от друга, их можно обрабатывать в нескольких потоках без сложной синхронизации состояния соединения.

WireGuard изначально не поддерживает выбор криптоалгоритмов на лету, поскольку сознательно делает ставку на фиксированный набор примитивов ради простоты и надежности.

В рамках данной работы предлагается реализовать VPN на базе WireGuard с расширением crypto agility. Суть подхода состоит в том, чтобы вынести криптографические примитивы в абстрактный слой и подключать их через унифицированные интерфейсы. Это позволит заменять алгоритмы шифрования, хэширования и обмена ключами без переработки всего протокола. Такой подход особенно полезен для крупных организаций, где могут действовать собственные требования к допустимым алгоритмам, длинам ключей и срокам миграции на новые стандарты.

Отдельное внимание уделяется производительности. При разработке учитываются подходы к снижению накладных расходов, включая минимизацию копирования данных и ускорение обработки трафика. Для увеличения пропускной способности предполагается распараллелить чтение из виртуального сетевого интерфейса и UDP-сокета на несколько рабочих потоков. Теоретически возможно выполнять чтение через один файловый дескриптор, однако при увеличении числа потоков это приводит к росту задержек, поскольку операция чтения является блокирующей, и в каждый момент времени только один поток может читать из дескриптора. Это ограничивает масштабируемость и приводит к возникновению узкого места.

Для устранения данного ограничения предлагается использовать отдельный файловый дескриптор для каждого потока. Это достигается за счёт многократного создания сокетов с опцией SO_REUSEPORT, а также виртуальных сетевых интерфейсов (IFF_MULTI_QUEUE). Такой подход позволяет каждому потоку независимо принимать и обрабатывать пакеты, что обеспечивает более равномерную загрузку процессора и снижает задержки при высокой интенсивности трафика.

Однако стандартный механизм балансировки для файловых дескрипторов UDP-сокетов и TUN-интерфейсов имеет ограничение: распределение входящих пакетов выполняется на основе хэширования полей заголовка пакета, по которому определяется конкретный файловый дескриптор, принимающий пакет. В результате все пакеты, относящиеся к одному потоку или источнику, направляются в один и тот же дескриптор. Это приводит к неравномерной загрузке: один поток может быть перегружен, тогда как остальные остаются недоиспользованными. В контексте VPN это особенно критично, поскольку трафик от множества клиентов проходит через одну и ту же точку тоннеля.

Для решения данной проблемы предлагается использовать eBPF-программу, подключаемую к механизму распределения пакетов между файловыми дескрипторами. Такой подход позволяет внедрить пользовательскую логику непосредственно в ядро и переопределить стратегию выбора сокета для каждого входящего пакета, устраняя зависимость от стандартного хэширования и обеспечивая более равномерное распределение нагрузки между потоками.

В результате пакеты от одного и того же источника могут обрабатываться разными потоками, что устраняет узкие места и позволяет более равномерно задействовать вычислительные ресурсы. Это снижает задержки обработки, уменьшает вероятность переполнения очередей и повышает общую пропускную способность системы. Особенно заметный эффект достигается при высокой нагрузке и большом числе одновременно обрабатываемых пакетов, что характерно для корпоративных VPN-сервисов.

Таким образом, работа сочетает практическую реализацию VPN удаленного доступа с двумя направлениями развития: гибкой криптографической архитектурой и повышением пропускной способности за счет системных оптимизаций. Полученное решение ориентировано на корпоративное применение, где важны одновременно безопасность, управляемость и высокая производительность.

Список использованных источников:

1. UDP for IPv4 and IPv6 [Электронный ресурс] / Linux man-pages project. – Режим доступа: https://man7.org/linux/man-pages/man7/udp.7.html.

2. sendto(2) – Linux manual page [Электронный ресурс] / Linux man-pages project. – Режим доступа: https://man7.org/linux/man-pages/man2/sendto.2.html.

3. recvfrom(3) – POSIX manual page [Электронный ресурс] / Linux man-pages project – Режим доступа: https://man7.org/linux/man-pages/man3/recvfrom.3p.html.

4. socket(7) – Linux manual page [Электронный ресурс] / Linux man-pages project. – Режим доступа: https://man7.org/linux/man-pages/man7/socket.7.html.

5. bind(2) – Linux manual page [Электронный ресурс] / Linux man-pages project. – Режим доступа: https://man7.org/linux/man-pages/man2/bind.2.html.

6. getsockopt(2) – Linux manual page [Электронный ресурс] / Linux man-pages project. – Режим доступа: https://man7.org/linux/man-pages/man2/setsockopt.2.html.

7. WireGuard: Next Generation Kernel Network Tunnel [Электронный ресурс] / J. A. Donenfeld. – Режим доступа: https://www.wireguard.com/papers/wireguard.pdf.

8. WireGuard Quick Start Guide [Электронный ресурс] / WireGuard. – Режим доступа: https://www.wireguard.com/quickstart/.

9. Universal TUN/TAP device driver [Электронный ресурс] / Linux Kernel Documentation. – Режим доступа: https://docs.kernel.org/networking/tuntap.html.

10. Таненбаум Э. Современные операционные системы. – СПб.: Питер, 2015. – С. 784-802.

11. Таненбаум Э., Уэзеролл Д. Компьютерные сети. – СПб.: Питер, 2017. – С. 611-623.