УДК 004.658 DOI: 10.17587/prin.13.124-131

Р. Э. Асратян, канд. техн. наук, вед. науч. сотр., rubezas@yandex.ru, Институт проблем управления им. В. А. Трапезникова Российской академии наук, Москва

Защищенный сетевой канал для web-сервисов на основе SSL/TLS в среде Linux

Рассмотрен подход к организации безопасного взаимодействия в распределенных системах через общедоступную сеть, использующий организацию защищенных каналов связи на основе протокола SSL/TLS. В отличие от технологии VPN, описываемый подход строго ориентирован на поддержку HTTP-взаимодействий, что позволяет подключить средства аутентификации и разграничения прав доступа к информационным ресурсам на основе сертификатов открытого ключа клиента в качестве готовых технических решений. Описана реализация подхода в среде Linux и результаты экспериментального исследования.

Ключевые слова: *распределенные системы, информационная безопасность, web-сервисы, технология SSL/TLS, сертификаты открытого ключа, Linux*

Введение

Внимание к средствам информационной безопасности в Интернете постоянно растет и приобретает почти всеобщий характер [1-4]. Эту тенденцию легко проследить: все последние годы число сайтов, использующих защищенный протокол HTTPS, непрерывно увеличивалось, и сегодня они заняли главенствующее положение в Интернете. Пожалуй, единственной областью, где классический HTTP еще сохраняет свои позиции, остаются распределенные системы (РС), построенные на основе web-сервисов [5, 6], но и здесь ситуация постепенно меняется. По крайней мере в среде Windows технология WCF (*Windows Communication Foundation*) [7—9] завоевывает все большее число сторонников среди разработчиков РС, которые все чаще делают выбор в пользу HTTPS в качестве сетевого "транспорта" для взаимодействий клиента и сервиса.

В среде Linux [10] ситуация складывается несколько иначе. Это во многом связано с тем, что поддержка WCF в среде программирования MonoDevelop — в основном средстве разработки РС на основе архитектуры .NET — пока еще находится в стадии становления с не вполне ясными перспективами. Тем не менее поддержка SOAP-сервисов в Linux хорошо зарекомендовала себя уже в течение ряда лет (что стало одним из факторов повышения интереса к среде Linux, которое можно наблюдать в последние годы).

Одной из важных задач, которые приходится решать разработчикам РС в Linux, является задача организации безопасного взаимодействия как с вновь создаваемыми, так и с уже созданными web-сервисами, ориентированными на протокол HTTP/SOAP. Общепринятый подход к решению этой задачи заключается в использовании VPN [11, 12], т. е. технологии создания защищенных каналов взаимодействия с ограниченным доступом через общедоступную сеть. Главным преимуществом этого подхода является его высокая универсальность: так как средства защиты в VPN подключаются на нижних уровнях иерархии протоколов OSI (как правило, не выше сетевого), то его "бенефициарами" оказываются все протоколы вышестоящих уровней. Альтернативный подход основан на применении протокола SSL/TLS для построения высокоуровневых защищенных сетевых туннелей [13, 14]. В отличие от VPN, средства защиты подключаются в этом случае на транспортном уровне иерархии протоколов, что также обеспечивает весьма высокую универсальность подхода: защищенный канал может быть использован для взаимодействия по любому протоколу уровня приложения.

Однако высокая универсальность рассмотренных подходов имеет и отрицательную сторону. Она не позволяет продвинуться в направлении создания готовых полезных решений в области защиты информации в той же степени, которая доступна более специализированным технологиям, сфокусированным на поддержку строго определенных протоколов приложения. Например, в области авторизации и тонкого разграничения прав доступа к данным на уровне не только отдельных web-сервисов, но и отдельных сервисных функций (методов). Известные универсальные средства организации защищенных сетевых каналов не предоставляют такой возможности (во всяком случае, не предоставляют в форме готового к использованию решения).

В данной статье описана технология построения защищенных каналов взаимодействия для РС, использующих технологию web-сервисов, в среде Linux. Основная идея подхода основана на использовании SSL/TLS для надстройки защищенного канала над уже открытым TCP-соединением [13, 15]. Однако

в отличие от известных подходов, строгая ориентация на поддержку web-сервисов позволяет включить в защищенный канал средства анализа HTTP/SOAP-траффика и готовые средства авторизации и тонкого разграничения прав доступа к данным, основанные на разборе HTTP/SOAP-заголовков в информационных запросах и на значениях реквизитов клиентов, содержащихся в предъявленных сертификатах открытого ключа [1]. Другими словами, предлагаемый подход представляет собой попытку получить выигрыш в функциональных возможностях канала за счет сознательного проигрыша в универсальности. Важно отметить, что подобное повышение функциональных возможностей канала, по-видимому, не имеет альтернативы в ситуации, когда требуется добавить средства тонкого разграничения прав доступа в РС без доработки уже существующих клиентских и серверных компонентов.

Описываемый защищенный канал основан на использовании специальных шлюзов, обеспечивающих переключение с HTTP на HTTPS на стороне клиента и переключение с HTTPS на HTTP на стороне web-сервера и составляющих защищенный канал взаимодействия (HTTPS-канал) для компонентов РС. Предполагается, что как программы клиента, так и web-серверы размещены в одних защищенных частных сетях (или даже на одних сетевых узлах) с обслуживающими их шлюзами и только взаимодействие между шлюзами осуществляется через общедоступную сеть (рис. 1). Важно подчеркнуть, что эти дополнительные возможности не диктуются потребностям конкретного проекта, они реализованы в форме готовых к использованию "общих" решений.

Основы организации защищенного канала

Функционирование защищенного канала проиллюстрировано на рис. 1. Его работа организована в соответствии с изложенными далее основными положениями.

- Защищенный канал опирается на соединение двух базисных технологий: SSL/TLS и технологии proxy-серверов. По своему статусу в системе и клиентский, и сервисный шлюзы представляют собой proxy-серверы — постоянно активные программы ("демоны"), выполняющие функции посредников между клиентскими и сервисными компонентами РС. В данном случае смысл "посредничества" заключается в криптозащите данных.
- Объектами обработки в канале являются не отдельные IP-пакеты, а электронные документы: информационные запросы к web-сервисам и результаты их обработки (ответы) в формате HTTP/SOAP (рис. 1).
- Защита данных в канале основывается на сертификатах открытого ключа клиентского и серверного шлюзов. Хотя передача сертификата клиента серверу, вообще говоря, является опциональной в современных версиях протоколов SSL/TLS, в данном случае она является обязательной: серверный шлюз немедленно разрывает соединение с клиентским шлюзом, если в результате процедуры "рукопожатия" (*handshaking*) сертификат последнего не был получен. Именно на характеристиках владельцев сертификатов (а не на учетных записях, паролях и т. п.) строятся средства аутентификации и контроля прав доступа к сервисам (разумеется, корректность данных в сертификатах обязательно проверяется на основе сертификата "доверенной" организации).
- Средства защиты опираются на возможности библиотек libssl и libcrypt для ОС Linux и сосредоточены исключительно в рамках защищенного канала. Ни клиентские, ни серверные компоненты РС не должны иметь дело ни с крипто-функциями, ни с сертификатами или закрытыми ключами. Это требование означает, что средства контроля прав доступа к сервисам, основанные на сертификатах, должны быть реализованы в самом канале, а не в клиентских и/или сервисных компонентах. Важно отметить, что в описанном подходе некорректные (нарушающие права доступа) запросы вообще не допускаются до сервера, они отвергаются еще на уровне канального шлюза. Разумеется, как программные модули шлюзов, так и их конфигурационные файлы (см. далее) должны быть доступны по записи только администраторам РС.
- Каждый клиентский шлюз способен взаимодействовать не с одним, а со многими серверными шлюзами поочередно. Выбор серверного шлюза осуществляется на основе интернет-имени адресуемого web-сервера с использованием таблицы маршрутизации, содержащейся в конфигурационном файле клиентского шлюза. Важная особенность описываемого подхода заключается в том, что функции маршрутизации "переплетаются" здесь с функциями защиты: упомянутая таблица содержит не только интернет-имена (или адреса) серверных шлюзов, но и требования к их сертификатам, сформулированные в терминах ограничений на реквизиты владельца (защита от сфальцифицированного серверного шлюза). С точки зрения пользователя самым важным компонентом шлюза является его конфигурационный файл ssltunnel.cfg, который содержит параметры (настройки), управляющие его работой (рис. 1). Параметры задаются в форме

```
ключевое слово = значение
```

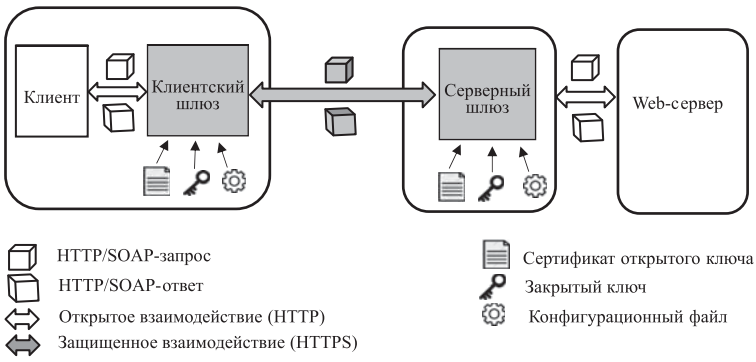

Рис. 1. Основы организации защищенного канала

Основные настройки шлюза защищенного канала

№	Ключевое слово	Настройка	Значение по умолчанию
1	ListenPort	Номер порта для входящих соединений	8128
2	ListenIp	IP-адрес (или имя) сетевой карты	127.0.0.1
3	ReadTimeout	Таймаут чтения из сокета, с	180
4	WriteTimeout	Таймаут записи в сокет, с	60
5	CertificateFile	Полное имя файла сертификата открытого ключа шлюза (или имя директории для поиска)	/media
6	PrivateKeyFile	Полное имя файла закрытого ключа шлюза (или имя директории для поиска)	/media
7	TrustedDir	Полное имя директории доверенных сертификатов	/etc/ssl/certs

Каждая настройка размещается в отдельной строке. В таблице перечислены основные параметры и связанные с ними ключевые слова.

Первые четыре параметра вполне характерны для сетевых серверных программ. Сертификат и закрытый ключ шлюза загружаются из файлов с расширением ".pem" с помощью функций SSL_CTX_use_certificate_file() и SSL_CTX_use_Privatekey_file() библиотеки libssl соответственно (они могут быть подготовлены с помощью утилиты openssl). Отметим, что вместо имен файлов могут быть указаны имена директорий: в этом случае шлюз осуществит поиск файлов в соответствующей директории и ее поддиректориях. Сертификат открытого ключа будет загружен из первого же найденного файла, имя которого заканчивается на "cert.pem" (например, My_cert.pem), а закрытый ключ — из первого же найденного файла, имя которого заканчивается на "key.pem" (например, My_key.pem). Такая организация позволяет считывать сертификаты и ключи со съемных носителей, задавая имя директории, в которую они монтируются (например, "/media"). Несмотря на имеющиеся в Linux развитые средства защиты директорий от несанкционированного доступа, хранение клиентских сертификатов и закрытых ключей на съемных носителях представляется наиболее безопасным.

Кроме параметров, указанных в таблице, конфигурационный файл может содержать разделы, отражающие специфику клиентского и серверного шлюзов. Эти разделы прямо относятся к поиску и проверке подлинности серверного шлюза (раздел [Forward] для клиентского шлюза) или к проверке прав клиента на доступ к web-сервисам и отдельным сервисным функциям (раздел [Access] для серверного шлюза). Лучше всего пояснить структуру этих разделов на конкретном примере.

Рассмотрим следующий пример. Предположим, что большая организация Titan имеет центральный офис в Москве и филиалы в областных центрах страны, а на web-серверах каждого офиса имеется web-сервис Staff и набор сервисных функций, предназначенных для регистрации и учета служащих этого филиала. К их числу относится ряд достаточно простых функций, обеспечивающих добавление, удаление и коррекцию записей о служащих в базе данных филиала (AddPerson, DeletePerson и CorrectPerson) и функцию ViewPersons, позволяющую получить список служащих офиса. Кроме того, на web-серверах каждого офиса имеется web-сервис Stat, который содержит функцию Report, позволяющую сформировать отчет о кадровой статистике филиала за любой период времени (например, о динамике средней зарплаты служащих в разрезе подразделений). Предположим также, что служащие компании имеют возможность обращаться к web-сервисам компании дистанционно со своих служебных или домашних рабочих станций через Интернет с использованием защищенных HTTPS-каналов. В целях безопасности web-серверы каждого офиса "спрятаны" в его частной сети, а доступ к сервисам из Интернета осуществляется через серверный шлюз защищенного канала, размещенный на выделенном сервере, имеющем подключения и к частной сети, и к Интернету. Будем считать, что компания снабжает каждого служащего личным закрытым ключом и сертификатом открытого ключа, удостоверенным электронной подписью центрального офиса. Файл сертификата главного офиса имеется на всех рабочих станциях и серверных шлюзах в директории /etc/ssl/certs/main_office. На рис. 2 приведен пример конфигурационного файла клиентского шлюза (размещенного на клиентской рабочей станции), а на рис. 3 — пример конфигурационного файла серверного шлюза.

Раздел [Forward] включает последовательность строк, каждая из которых содержит описание одного серверного шлюза. Это описание связывает интернет-имя адресуемого web-сервера с основными характеристиками обслуживающего его серверного шлюза — IP-адресом (или интернет-именем) и требования к реквизитам владельца (в формате, соответствующем стандарту X509), которым должен удовлетворять сертификат сервера. Например, если клиентская программа выполняет вызов web-сервиса с URL, равным http://www.titan.spb.ru/staff.asmx, то в соответствии со второй строкой в разделе [Forward]:

```
ListenPort=8128
CertificateFile=/media
PrivateKeyFile=/media
TrustedDir=/etc/ssl/certs/main_office

[Forward]
www.titan.moscow.ru 193.182.12.1 C=Russia,L=Moscow,O=Titan,CN=TitanSrv
www.titan.spb.ru 47.172.33.2 C=Russia,L=StPeterburg,O=Titan,CN=TitanSrv
. . .
www.titan.omsk.ru 146.58.130.3:8443 C=Russia,L=Omsk,O=Titan,CN=TitanSrv
```

Рис. 2. Пример конфигурационного файла клиентского шлюза

```
ListenPort=443
ListenIp=47.172.33.2
CertificateFile=/media
PrivateKeyFile=/media
TrustedDir=/etc/ssl/certs/main_office

[Access]
www.titan.spb.ru/staff.asmx C=Russia,L=StPeterburg,O=Titan
  AddPerson   C=Russia,L=StPeterburg,O=Titan,OU=Inform Department
  DeletePerson C=Russia,L=StPeterburg,O=Titan,OU=Inform Department
  CorrPerson C=Russia,L=StPeterburg,O=Titan,OU=Inform Department
  ViewPersons
www.titan.spb.ru/stat.asmx
  Report  C=Russia,L=StPeterburg,O=Titan,CN=Admin*:* |
         C=Russia,O=Titan,CN=Top*:*
```

Рис. 3. Пример конфигурационного файла серверного шлюза

- клиентский шлюз защищенного канала выполнит сетевое соединение с серверным шлюзом с IP-адресом 47.172.33.2 (порт 443 предполагается по умолчанию) и инициирует создание защищенного канала в соответствии с технологией SSL/TLS;
- если поле Subject Name в сертификате, полученном от серверного шлюза в результате процедуры "рукопожатия" (*handshaking*), будет содержать реквизиты владельца, отличные от C=Russia, L=StPeterburg, O=Titan, CN=TitanSrv (страна Russia, город StPeterburg, организация Titan, имя/название TitanSrv), то клиентская программа получит сообщение о сфальсифицированном серверном шлюзе и соединение с последнем будет немедленно разорвано.

Раздел [Access] на рис. 3 содержит описание ограничений доступа к web-сервисам санкт-петербургского филиала. Здесь легко увидеть строки, содержащие интернет-имена двух web-сервисов (www.titan.spb.ru/staff.asmx и www.titan.spb.ru/stat.asmx). Кроме имени сервиса каждая из этих строк может содержать требования к реквизитам клиентов, которым разрешено обращаться к данному сервису в уже знакомой нам форме: содержание поля Subject Name в сертификате, полученном от клиентского шлюза в результате процедуры "рукопожатия" (*handshaking*), должно соответствовать этим требованиям. После строки, содержащей имя web-сервиса, могут следовать строки, содержащие имена отдельных функций. Эти строки также могут содержать требования к реквизитам клиентов, специфичные для той или иной функции. В приведенном примере только служащие санкт-петербургского офиса из подразделения "Inform Department" имеют право вызывать функции AddPerson, DeletePerson и CorrPerson, а функция ViewPerson доступна любому служащему офиса.

Права доступа к функции Report определяются несколько сложнее. В данном примере предполагается, что в соответствии с корпоративным стандартом реквизит CN (*Common Name*) в сертификате служащего должен быть задан в формате "должность:имя". В приведенном примере право доступа к функции имеют все служащие санкт-петербургского офиса, название должности которых начинается со слова "Admin" (символ "*" означает "любая подстрока"), а также служащие любого офиса компании, название должности которых начинается со слова "Top" (символ "|" в данном случае означает "или").

Сообщение о сфальсифицированном сервере или о нарушении прав доступа к сервисам передаются в программу клиента в форме HTTP-ответа с кодом 500 (Внутренняя ошибка сервера), содержащего текст сообщения в формате SOAP. Этот ответ создает исключительную ситуацию в программе клиента (если вызов сервисной функции заключен в блок "try—catch", то программа клиента легко может получить текст сообщения из параметра предложения catch). Разумеется, все это справедливо и в отношении любых других диагностических сообщений от шлюзов, связанных с неудачной попыткой сетевого соединения, с ошибкой чтения из сокета или записи в сокет, с некорректным сертификатом и т. п. Если диагностическое сообщение порождено клиентским шлюзом, то оно передается непосредственно программе клиента, а если серверным шлюзом, то оно сначала передается в клиентский шлюз, а потом программе клиента.

Логика работы шлюзов

Как и у всякой программы на языке C++ (использование libssl естественно приводит к выбору именно этого языка программирования), сразу после запуска шлюза управление получает "главная" функция main. Она немедленно выполняет ряд действий, характерных для фоновых программ:

- переключение в режим "демона" с разрывом связей с монитором, клавиатурой и мышью;
- установка собственных обработчиков так называемых "сигналов" ОС Linux для потребного реагирования на экстренные ситуации (например, для создания записи в журнале о получении сигнала SIGTERM перед прекращением работы по команде оператора).

Кроме того, функция main выполняет чтение всех настроек из конфигурационного файла, загружает закрытый ключ, сертификат открытого ключа и управляющую информацию (из разделов [Forward] или [Access]) во внутренние структуры данных и выполняет подключение и инициализацию средств поддержки SSL/TLS из библиотеки libssl. После этого она выполняет действия, характерные для любого сетевого сервера, основанного на TCP/IP [14, 15]: создает сетевой сокет для приема запросов на соединение, привязывает его к порту, номер которого задан в параметре ListenPort (системный вызов bind()); включает режим прослушивания порта (системный вызов listen()); входит в "вечный цикл", обеспечивающий обнаружение и обслуживание запросов на сетевое соединение (системный вызов accept()).

Как и всякий многоканальный сервер, шлюз создает отдельный обрабатывающий программный поток ("нить") для обслуживания каждого запроса таким образом, что сразу несколько запросов могут одновременно находиться в состоянии обслуживания. Каждая обрабатывающая программная нить (вернее, ее основная функция) получает от функции main главный параметр — номер сетевого сокета для "удаленного общения" с программой, запросившей соединения и обслуживания.

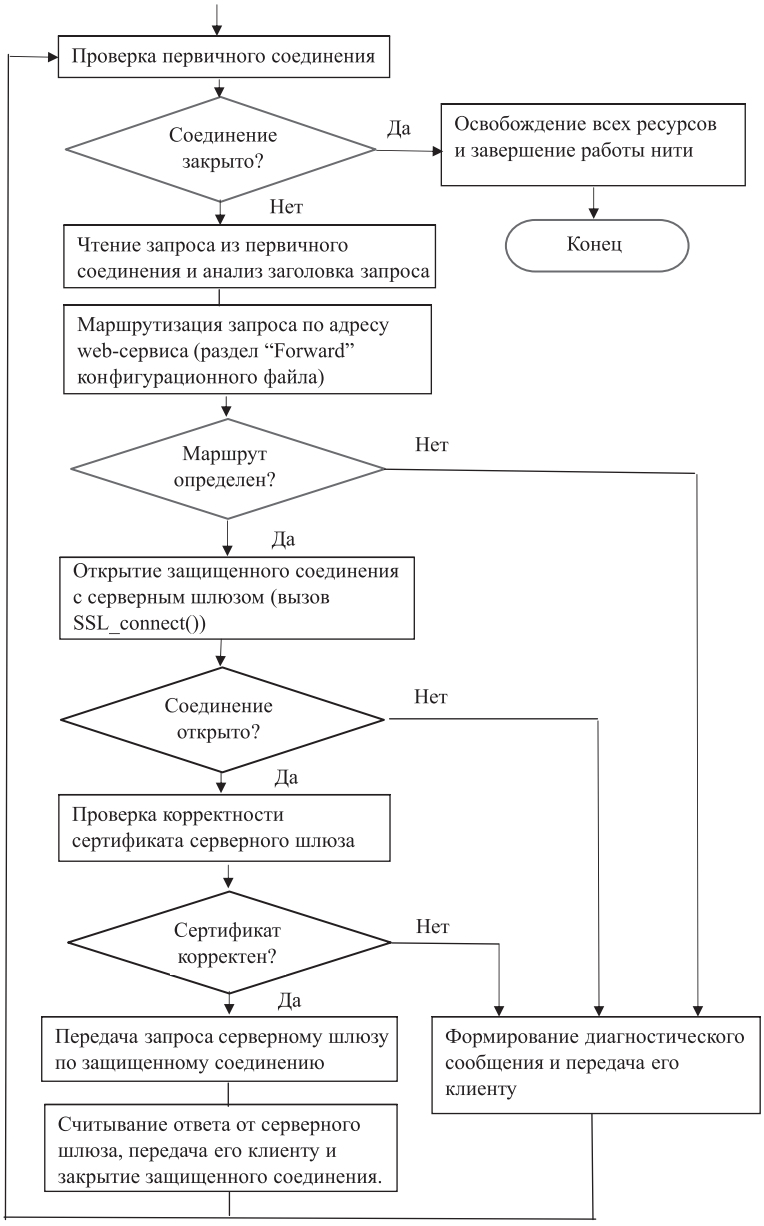

Рис. 4. Алгоритм работы обрабатывающей нити в клиентском шлюзе

Блок-схемы алгоритмов работы обрабатывающих нитей в клиентском и серверных шлюзах приведены на рис. 4 и 5 соответственно. В их "поведении" имеются общие черты: каждая нить соединяет в себе функции сервера и клиента одновременно. Она получает сокет "первичного" соединения от функции main, запрашивает "вторичное" соединение с удаленным сервером и выполняет перенос данных из первичного соединения во вторичное и обратно. Для клиентского шлюза первичное соединение используется для связи с программой клиента, а вторичное — для связи с серверным шлюзом. Для серверного шлюза первичное соединение используется для связи с клиентским шлюзом, а вторичное — с web-сервером. На этом общие черты в поведении нитей заканчиваются.

Главное отличие в логике работы обрабатывающих нитей в клиентском и серверном шлюзах вытекает из того, что в первом случае первичное соединение является незащищенным, а вторичное — защищенным, а во втором случае — наоборот. Поэтому обращение к функциям библиотеки libssl (вызов SSL_connect()) в клиентском шлюзе осуществляется на более поздних этапах обработки запроса, чем в серверном шлюзе. Как видно на блок-схеме рис. 4, создание защищенного канала выполняется после анализа и маршрутизации информационного запроса (поиска адекватного серверного шлюза), а в блок-схеме на рис. 5 — сразу же после создания обрабатывающей нити.

Временные оценки

Как видно из приведенного описания, использование защищенного канала предполагает включение двух промежуточных серверов-шлюзов между клиентской программой и web-сервером. Такая организация не может не вызвать вопроса о размере неизбежной дополнительной задержки в обработке запросов к web-серверу. Для оценки этой задержки была выполнена реализация защищенного канала на основе библиотек libssl и libcrypt (пакет libssl-dev версии 1.1.1d, версия TLS 1.2) в Linux Debian 10 на языке C++ и проведена серия экспериментов в лабораторной сети с использованием web-сервера apache2 версии 2.4.

Так как значение дополнительной задержки при вызове web-сервиса, очевидно, зависит от размеров информационного запроса и результата его выполнения (сетевого трафика), были использованы несколько сервисных функций с различным объемом передаваемых данных в обоих направлениях и различными временами обработки. На рис. 6 приведено время обращения к сервисной функции с малым временем обработки (100 мс) для различных объемов передаваемых данных (10 и 200 Кбайт) в двух режимах: без защиты ("напрямую") и через защищенный канал. На рис. 7 приведено время обращения в тех же условиях к более "медленной" функции (время обработки 500 мс).

Как видно на рис. 6 и 7, при трафике 10 Кбайт после подключения защищенного канала время выполнения запроса к функции с временем обработки 100 мс увеличилось на 13 мс (12 %), а при трафике 200 Кбайт — на 27 мс (21 %). При вызове же функции с временем обработки 500 мс дополнительная задержка составила 14 мс (2,7 %) при трафике 10 Кбайт и 29 мс (5,5 %) при трафике 200 Кбайт. Как и следовало ожидать, при вызове более "медленной" функции потери быстродействия выглядят более умеренными.

На рис. 8 приведены результаты экспериментальной оценки быстродействия защищенного канала, но не в режиме одиночных запросов, а в условиях

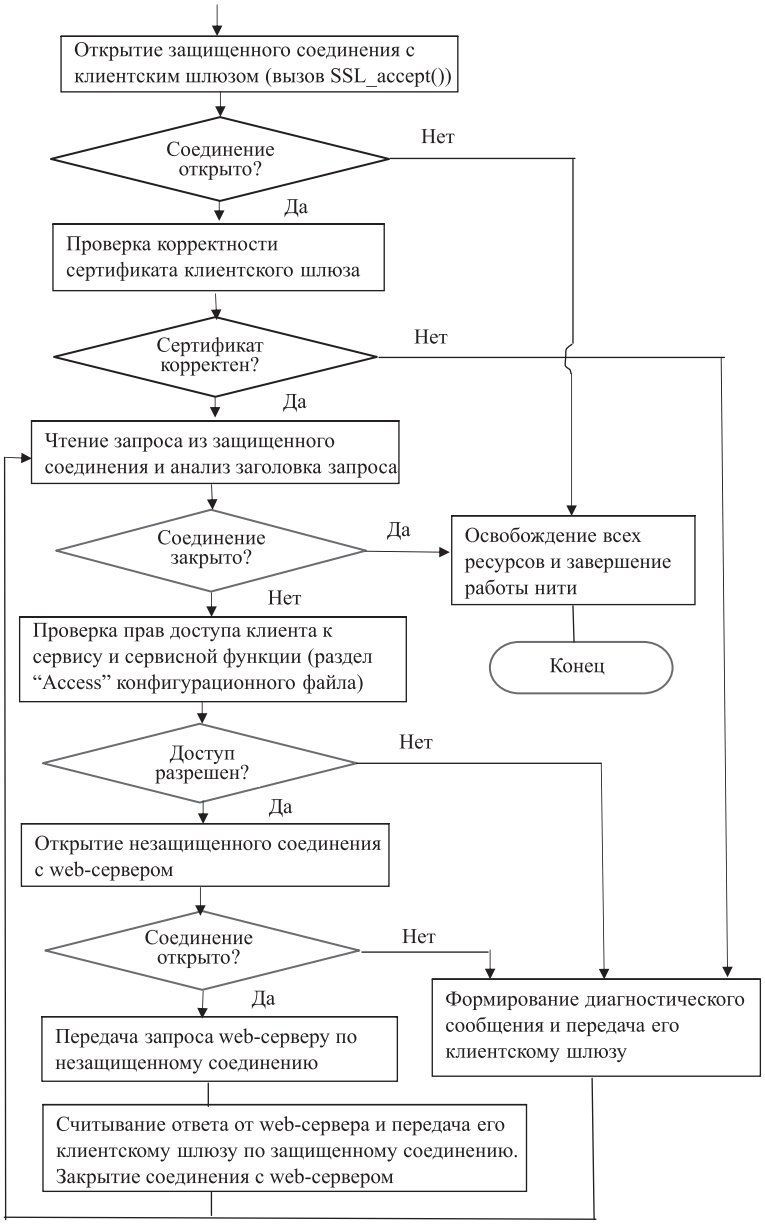

Рис. 5. Алгоритм работы обрабатывающей нити в серверном шлюзе

высокой нагрузки: при параллельной обработке пакетов одновременно поступивших информационных запросов. Кривые, представленные на рис. 8,

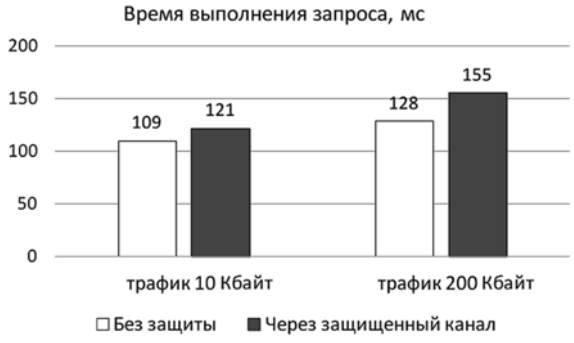

Рис. 6. Время выполнения запроса к "быстрой" функции (100 мс)

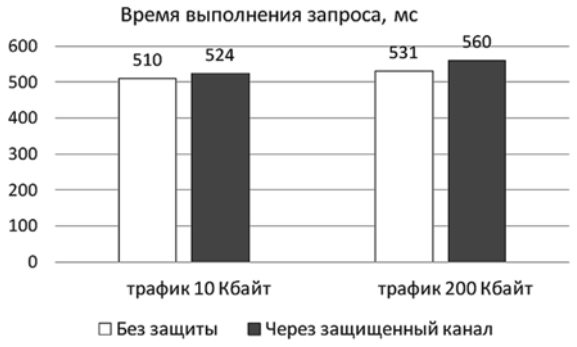

Рис. 7. Время выполнения запроса к более "медленной" функции (500 мс)

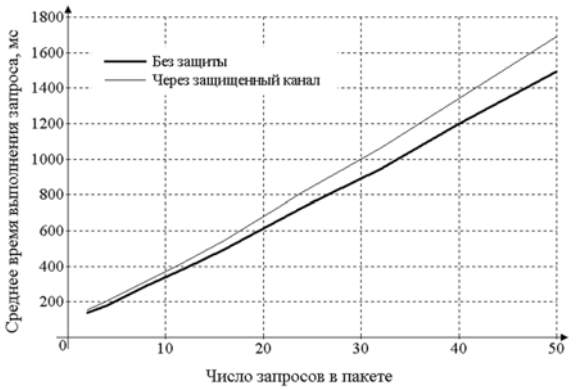

Рис. 8. Среднее время выполнения запроса в пакете одновременных запросов

отражают характерную зависимость среднего времени выполнения запроса от их числа в пакете при вызове "быстрой" сервисной функции со временем обработки 100 мс и объемом передаваемых данных 50 Кб в двух режимах: без защиты ("напрямую") и через защищенный канал. Как видно на рис. 8, обе кривые демонстрируют устойчивый рост, так как с увеличением числа запросов увеличивается и число обрабатывающих нитей, а значит и нагрузка на сервер. Однако относительное увеличение среднего времени выполнения запроса вследствие подключения защищенного канала не превышает 15 %. Примечательно, что это относительное увеличение остается довольно стабильным с ростом числа запросов в пакете.

Как видно из приведенных результатов, при вызове "быстрых" сервисных функций с временем обработки 0,1 с относительные потери быстродействия могут быть существенными, особенно при большом объеме передаваемых данных (см. рис. 6 и 8). Однако при увеличении времени обработки сервисной функции до 0,5 с относительные потери снижаются до нескольких процентов даже при трафике 200 Кбайт. По-видимому, именно сервисные функции со временем обработки 0,5 с и выше составляют область наиболее эффективного применения описанного подхода.

Заключение

Разумеется, при разработке любой РС закладываются и реализуются те или иные средства аутентификации и авторизации еще на уровне клиентских компонентов. Как правило, эти средства бывают "привязаны" к особенностям конкретного проекта. Описанный подход к организации проверки и разграничения прав доступа к сервисам и сервисным функциям в рамках защищенного сетевого канала позволяет наложить общие ограничения на поведение любых клиентов, использующих этот канал. Это особенно важно в тех случаях, когда потребителям информации предоставляется возможность разработки собственных клиентских компонентов для доступа к опубликованным web-сервисам и информационным ресурсам. Как уже отмечалось, в описанном подходе некорректные (нарушающие права доступа) запросы вообще не допускаются до сервера, а отвергаются на уровне канального шлюза. Это дает возможность добавить средства авторизации к уже существующим сервисам, не изменяя их.

Применение описанного выше защищенного канала не сильно усложняет задачу сетевого администрирования. Самое общее требование к сетевой инфраструктуре РС заключается в том, что межсетевые экраны не должны запрещать сетевые соединения по адресам и портам, которые используются канальными шлюзами. Другими словами, клиент должен иметь возможность соединения с клиентским шлюзом, клиентский шлюз — с серверным шлюзом, а северный шлюз — с сервером.

Несколько слов о подключении клиентского шлюза: как следует из логики работы канала, сертификат открытого ключа клиентского шлюза определяет права пользователя на доступ к web-сервисам и сервисным функциям. Фактически это означает, что запущенный клиентский шлюз должен обслуживать только клиентские программы, запущенные определенным пользователем. Наиболее естественный способ удовлетворить это требование заключается в размещении клиентского шлюза на рабочей станции пользователя (см. рис. 1) и задании значения 127.0.0.1 в качестве адреса "прослушивания" (значение по умолчанию) с тем, чтобы только локальные клиентские программы могли им пользоваться. Если же не один, а несколько пользователей имеют право поочередного доступа к рабочей станции (достаточно редкая ситуация в наши дни), то следует воспользоваться возможностью предоставления сертификатов и ключей на съемном носителе. Разумеется, на такой рабочей станции нужно запретить дистанционный вход через удаленный рабочий стол, ssh или telnet [14].

Важно отметить, что описанный канал может быть использован не только в клиентских компонентах РС для вызова функций web-сервисов, но и в служебных программах. Например, утилита wsdl, широко используемая для считывания формализованного описания web-сервиса по сети и автоматической генерации текста связующего модуля, вполне уверенно работает через описанный канал (при условии, что в системных сетевых настройках указаны адрес и порт клиентского шлюза в качестве параметров proxy-сервера). Другими словами, предложенная технология может быть использована не только при эксплуатации РС, но и при их разработке.

В имеющейся на настоящее время реализации защищенного канала функции клиентского и сервисного шлюзов выполняются одной и той же программой (разница только в конфигурационных настройках). Шлюзы приспособлены к работе в режиме "демонов" — невидимых фоновых программ, разрывающих связи с монитором, клавиатурой и мышью сразу после запуска. Во время работы шлюз занимает всего около 1,2 Мбайт оперативной памяти.

Список литературы

1. **Козлов А. Д., Орлов В. Л.** Методы и средства обеспечения информационной безопасности распределенных корпоративных систем. М.: ИПУ РАН, 2017. 156 с.
2. **Салимова Ш. А.** Кибербезопасность в России: актуальные угрозы и пути обеспечения в современных условиях // Достижения вузовской науки 2021: сб. статей XVII Международного научно-исследовательского конкурса, Пенза, 20 января 2021 г. Пенза: Наука и Просвещение, 2021. С. 207—214.
3. **Жаранова А. О., Птицына Л. К.** Анализ влияния распределенности на качество функционирования комплексных систем защиты информации // Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО 2020): сб. науч. статей IX Международной научно-технической и научно-методической конференции. СПб: СПБГУТ, 2020. С. 324—327.
4. **Згоба А. И., Маркелов Д. В., Смирнов П. И.** Кибербезопасность: угрозы, вызовы, решения // Вопросы кибербезопасности. 2014. № 5. С. 30—38.
5. **Шапошников И. В.** Web-сервисы Microsoft.NET. СПб: БХВ-Петербург, 2002. 336 с.
6. **Мак-Дональд М., Шпушта М.** Microsoft ASP.NET 3.5 с примерами на C# 2008 и Silverlight 2 для профессионалов. М.: Вильямс, 2009. 1408 с.
7. **Liu M.** WCF multi-layer services development with entity framework. Birmingham: Packt Publishing, 2014. 378 p.
8. **Lowy J., Montgomery M.** Programming WCF services: design and build maintainable service-oriented systems. N. Y.: O'Reilly Media, 2015. 1018 p.
9. **Костин Е. И.** Создание службы WCF для использования в клиент-серверном приложении // Инновационные подходы в решении научных проблем: сб. тр. по материалам IV Международного конкурса научно-исследовательских работ, Уфа, 01 марта 2021 г. Уфа: Научно-издательский центр "Вестник науки", 2021. С. 121—126.
10. **Negus C.** Linux Bible. N. J.: Wiley, 2015. 912 p.
11. **Запечников С. В., Милославская Н. Г., Толстой А. И.** Основы построения виртуальных частных сетей. — М: Горячая линия-Телеком, 2011. 248 с.
12. **Акушуев Р. Т.** Принцип работы VPN и его особенности // Modern Science. 2020. № 7. С. 312—314.
13. **Baka P., Schatten J.** SSL/TLS under lock and key: a guide to understanding SSL/TLS cryptography. Keyko books, 2020. 132 p.
14. **Хант К.** TCP/IP. Сетевое администрирование. СПб.: Питер, 2007. 816 с.
15. **Снейдер Й.** Эффективное программирование TCP/IP. Библиотека программиста. СПб.: Символ-Плюс, 2002. 320 с.

Secure Network Channel for Web Services based on SSL/TLS in a Linux Environment

R. E. Asratian, rea@ipu.ru, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, 117997, Russian Federation

Corresponding author:

Asratian Ruben E., Leading Researcher, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, 117997, Russian Federation
E-mail: rea@ipu.ru

Received on January 27, 2022
Accepted on February 14, 2022

An approach to the organization of secure interaction in distributed systems via a public network is considered, based on the organization of secure communication channels based on SSL/TLS technology. Unlike VPN technology, the described approach is strictly focused on supporting only HTTP/SOAP interactions in distributed systems, which allows you to implement authentication and authorization based on HTTP-header data and client public key certificates as ready-made technical solutions. The approach implies the use of special gateways that provide switching from HTTP to HTTPS on the client side and switching from HTTPS to HTTP on the web server side and make up a "transparent" communication channel for system components. It is assumed that both client programs and web servers are located in the same secure private network (or even on the same network node) with the gateways serving them, and only the interaction between the gateways is carried out through the public network. The work of gateways is based on the use of SSL/TLS technology to add a secure channel over an already open TCP connection. The main idea of the approach is that in this case, security tools are connected at high levels of the OSI protocol hierarchy, which allows gateways to analyze high-level parameters of information requests and responses of web servers contained in HTTP-headers. And this, in turn, allows you to add additional "intelligence" to the gateways associated with authentication of servers and clients, as well as with the differentiation of access rights to information resources up to individual functions (methods) of web services based on the data contained in "Subject Name" attribute of public key certificates. The implementation of the approach in the Linux environment and the results of an experimental study are described. In particular, the study showed that when calling service functions with a runtime of 0.5 seconds or higher, the secure channel increases the total query execution time by only a few percent, even with a rather large amount of data being transmitted (up to 200 kilobytes).

Keywords: *distributed systems, information security, web services, SSL/TLS technology, public key certificates, Linux*

For citation:

Asratian R. E. Secure Network Channel for Web Services based on SSL/TLS Technology in a Linux Environment, *Programmnaya ingeneria*, 2022, vol. 13, no. 3. 124—131.

DOI: 10.17587/prin.13.124-131

References

1. **Kozlov A. D., Orlov V. L.** *Methods and tools for ensuring information security of distributed corporate systems*, Moscow, IPU RAN, 2017, 156 p. (in Russian).
2. **Salimova S. A.** Cybersecurity in Russia: current threats and ways to ensure in modern conditions, *Dostizhenija vuzovskoj nauki 2021: sbornik statej XVII Mezhdunarodnogo nauchno-issledovatelskogo konkursa,* Penza, 20 January 2021, Penza, Nauka i Prosveshhenie, 2021, pp. 207—214 (in Russian).
3. **Zharanova A. O., Pticyna L. K.** Analysis of the impact of distribution on the quality of functioning of complex information security systems, *Aktual'nye problemy infotelekommunikacij v nauke i obrazovanii (APINO 2020): sbornik nauchnyh statej IX Mezhdunarodnoj nauchno-tehnicheskoj i nauchno-metodicheskoj konferencii*, Saint Petersburg, SPBGUT, 2020, pp. 324—327 (in Russian).
4. **Zgoba A. I., Markelov D. V.** Cyber security: threats, challenges, decisions, *Voprosy kiberbezopasnosti*, 2014, no. 5, pp. 30—38 (in Russian).
5. **Shaposhnikov I. V.** *Web-services Microsoft.NET*, Saint Petersburg, BHV-Peterburg, 2002, 336 p. (in Russian).
6. **Mak-Donald M., Szpuszta M.** *Pro ASP.NET 3.5 in C#2008 Includes Silverlight 2*, Moscow, Willams, 2009, 1408 p. (in Russian).
7. **Liu M.** *WCF multi-layer services development with entity framework*. Birmingham: Packt Publishing, 2014, 378 p.
8. **Lowy J, Montgomery M.** *Programming WCF services: design and build maintainable service-oriented systems*, N. Y., O'Reilly Media, 2015, 1018 p.
9. **Kostin E. I.** Creating a WCF service for use in a client-server application, *Innovacionny'e podxody v reshenii nauchnyh problem: sbornik trudov po materialam IV Mezhdunarodnogo konkursa nauchno-issledovatelskih rabot,* Ufa, 01 March 2021, Ufa, Nauchno-izdatelskij centr "Vestnik nauki", 2021, pp. 121—126 (in Russian).
10. **Negus C.** *Linux Bible*, N.J., Wiley, 2015, 912 p.
11. **Zapechnikov S. V., Miloslavskaya N. G., Tolstoy A. I.** *Virtual private networks building principles*, Moscow, Goryachaya linia, 2011, 248 p. (in Russian).
12. **Akushuev R. T.** The principle of VPN operation and its features, *Modern Science*, 2020, no. 7, pp. 312—314.
13. **Baka P., Schatten J.** *SSL/TLS under lock and key: a guide to understanding SSL/TLS cryptography*, Keyko books, 2020, 132 p.
14. **Hant K.** *TCP/IP. Network administration*, Saint Petersburg, Piter, 2007, 816 p. (in Russian).
15. **Snader J.** *Effective TCP/IP programming*, Saint Petersburg, Simvol-Pljus, 2002, 320 p. (in Russian).