УДК 004.738

Р. Э. Асратян, канд. техн. наук, вед. науч. сотр., rea@ipu.ru,
В. Н. Лебедев, канд. техн. наук, зав. лаб., lebvini@ipu.ru,
В. Л. Орлов, канд. техн. наук, вед. науч. сотр., ovl@ipu.ru
Институт проблем управления им. В. А. Трапезникова РАН, г. Москва

Организация защищенных каналов взаимодействия на основе применения протокола HTTPS в прокси-серверах

Рассматривается метод организации безопасного информационного взаимодействия в распределенных системах, функционирующих в мультисетевой среде, основанный на применении протокола HTTPS и технологии прокси-серверов. Суть подхода заключается в организации защищенных HTTPS-соединений между прокси-серверами для безопасной передачи информационных запросов из одной частной локальной сети в другую через глобальную сеть с освобождением клиентских и серверных компонентов системы от функций информационной защиты.

Ключевые слова: *распределенные системы, Интернет-технологии, сетевые протоколы, прокси-серверы, информационное взаимодействие, информационная безопасность*

Введение

Сетевой протокол HTTPS [1] представляет собой одно из наиболее распространенных и удобных средств организации защищенного информационного взаимодействия в глобальных сетях. В последние годы значение этого протокола заметно возросло в связи с появлением возможности защиты информационных запросов к электронным сервисам в рамках технологии .NET [2, 3] на его основе (протокол SOAP "поверх" HTTPS). Этому же способствовало и широкое распространение распределенных информационных систем [4], ориентированных на работу в сложных мультисетевых средах (т. е. средах, включающих несколько связанных между собой сетей различных размеров и административной подчиненности), а также постоянно возрастающие требования к безопасности данных [5, 6].

Возможность применения HTTPS в разработках распределенных систем в значительной степени обусловлена его совместимостью с современными средствами интеграции сетей и в том числе с прокси-серверами (основы технологии HTTPS-взаимодействия через серверы-посредники предложены еще в конце 1990-х годов). Однако именно здесь разработчики распределенных систем сталкиваются с рядом проблем.

Рассмотрим мультисетевую среду, включающую несколько частных локальных сетей предприятий, объединенных глобальной сетью (рис. 1). Так как прямая *IP*-маршрутизация между частными сетями и глобальной сетью невозможна, задача обмена данными между локальными сетями в данном примере решается с помощью прокси-серверов, каждый из которых оснащен двумя сетевыми интерфейсами: один из них подключен к частной сети (будем называть его "внутренним"), а другой — к глобальной сети (будем называть его "внешним"). Данный подход к решению задачи межсетового обмена часто оказывается наиболее предпочтительным по сравнению с остальными (например, с технологиями NAT [7] или VPN [7, 8]), так как на прокси-серверы можно возложить дополнительные функции по учету (регистрации), контролю, фильтрации и маршрутизации информационных запросов. В частности, контроль может включать проверку типа исходящего из частной сети HTTP-запроса (например,

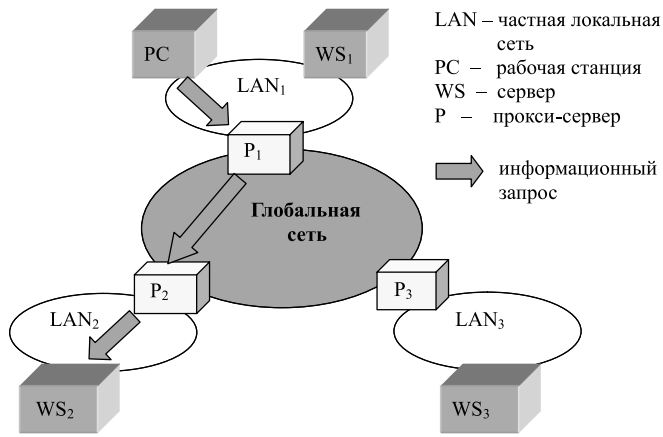

Рис. 1. Маршрутизация запросов в мультисетевой среде

допускаются только SOAP-запросы к электронным сервисам) или же проверку права доступа у источника входящего извне запроса к информационному ресурсу в частной сети.

Рассмотрим ситуацию, в которой клиентская программа, работающая в одной из частных сетей, посылает защищенный запрос к серверу, размещенному в другой частной сети по протоколу HTTPS. Каноническая форма поддержки HTTPS в прокси-сервере заключается в переключении последнего в так называемый "прозрачный режим" с помощью HTTP-метода (команды) CONNECT сразу после установления соединения с удаленным сервером. Этот режим заключается в "слепом" копировании всех данных, поступивших на внутренний сетевой интерфейс, во внешний интерфейс и наоборот вплоть до закрытия одного из соединений. Другими словами, создание защищенного канала в данном случае берут на себя клиентская и серверная программы, а от посредника требуется только "прозрачность". Так как данные от клиента и от сервера поступают в прокси-сервер в зашифрованной форме, он, очевидно, лишается возможности регистрации, контроля и вообще какой-либо обработки информационных запросов. Другими словами, в рассмотренной ситуации технология прокси-серверов теряет свое главное преимущество, например, перед технологией NAT.

В данной работе рассматривается альтернативный подход к обработке HTTPS-запросов в мультисетевой среде. Суть подхода заключается в организации защищенных HTTPS-тоннелей между прокси-серверами (т. е. между удаленными частными сетями) через глобальную сеть. При данном подходе удаленные клиентские и серверные программы пользуются протоколом HTTP и нисколько не заботятся об информационной защите, а на прокси-серверы возложены дополнительные функции шлюзов "HTTP → HTTPS" и "HTTPS → HTTP". Преимущество подхода заключается в том, что в данном случае прокси-серверы "видят" проходящие через них информационные запросы клиентов и ответы серверов и способны обеспечить необходимый контроль и обработку потока запросов.

Принципы организации HTTPS-тоннеля

При организации информационного взаимодействия между удаленными частными сетями предприятий в мультисетевой среде (см. рис. 1) на первый план выходят две взаимосвязанные задачи:

- обеспечение информационной безопасности данных в условиях угроз, исходящих от глобальной сети;
- маршрутизация запросов между клиентскими и серверными программами, функционирующими в разных частных сетях.

Суть рассматриваемого подхода заключается в соединении возможностей технологии HTTPS и технологии прокси-серверов для решения этих задач. Главная особенность подхода заключается в обеспечении "высокоуровневых" средств защиты и маршрутизации, основанных на заголовках и данных HTTP-запросов, а не отдельных IP-пакетов (отметим, что решение этих задач на сетевом уровне, например, на основе VPN может оказаться невозможным или совершенно неэффективным, если число частных сетей исчисляется десятками) [9].

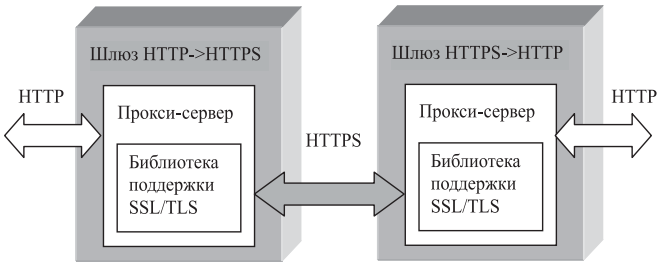

Рис. 2. Принципы организации HTTPS-тоннеля

Организация защищенного HTTPS-тоннеля с помощью двух прокси-серверов, выполняющих функции шлюзов "HTTP → HTTPS" и "HTTPS → HTTP", проиллюстрирована на рис. 2. Как видно из рисунка, каждый из прокси-серверов оснащается программной поддержкой SSL/TLS-технологии, обеспечивающей построение защищенного двустороннего канала взаимодействия "поверх" обычного TCP-соединения (отметим, что для канонического "прозрачного" режима такая поддержка не нужна). Главные особенности, которые режим шлюза привносит в поведение обычного прокси-сервера, заключаются в следующем.

- После получения HTTP-запроса от клиента, шлюз "HTTP → HTTPS" извлекает Интернет-имя (URL) адресуемого информационного ресурса из заголовка запроса и по имеющейся в его составе таблице маршрутизации определяет IP-адрес и номер порта удаленного шлюза "HTTPS → HTTP", защищающего этот ресурс (механизм маршрутизации рассмотрим чуть ниже). Кроме того, он проверяет корректность типа запроса (строка Content-type заголовка), если это необходимо.
- После установления TCP-соединения с удаленным шлюзом "HTTPS → HTTP" проводится построение двунаправленного защищенного HTTPS-канала "поверх" этого соединения с помощью программной поддержки SSL/TLS. Главное содержание этой операции заключается в согласовании методов шифрования и электронной подписи и в обмене сертификатами, содержащими открытые ключи контрагентов (в данном случае — двух шлюзов или двух организаций). С этого момента все данные, заносимые в защищенный канал, будут автоматически подписываться и шифроваться, а все данные, полученные из защищенного канала, — расшифровываться и аутентифицироваться. Сразу же после

успешного построения защищенного канала полученный от клиента информационный запрос немедленно отправляется удаленному шлюзу.

- После получения сертификата контрагента удаленный шлюз "HTTPS → HTTP" проверяет его корректность и извлекает из него имя владельца (название организации) для проверки его прав доступа к ресурсам в локальной сети. Если проверка заканчивается успешно, шлюз "HTTPS → HTTP" считывает информационный запрос по защищенному каналу, извлекает URL адресуемого информационного ресурса из заголовка запроса и по имеющейся в его составе таблице маршрутизации определяет IP-адрес и номер порта сервера в локальной сети, содержащего адресуемый ресурс. Далее следует установление TCP-соединения с этим сервером, передача запроса по новому соединению и чтение ответа сервера.
- Возврат ответа клиентской программе проводится аналогично через защищенный канал и через первичное соединение с клиентом, которые во все время обработки запроса остаются открытыми.

Как видно из изложенного, основные отличия рассмотренного подхода от канонического заключаются в следующем:

- Ни один из участников взаимодействия не использует HTTP-команду CONNECT.
- Клиентская и серверная программы не участвуют в создании защищенного канала через глобальную сеть.
- Прокси-серверы имеют возможность анализа, фильтрации и маршрутизации информационных запросов.
- Авторизацию информационных запросов и проверку прав доступа к информационным ресурсам выполняют не серверные программы, а прокси-серверы. Благодаря этому, не заслуживающие доверия запросы вообще не проникают внутрь частной сети, что повышает уровень информационной безопасности.

Маршрутизация запросов в мультисетевой среде

Как уже отмечалось, каждый запрос, направленный из одной частной сети в другую, проходит, по крайней мере, через два прокси-сервера и через два этапа маршрутизации (см. рис. 1). Из ближайшего прокси-сервера запрос сначала передается в удаленный прокси-сервер по глобальной сети, а уже потом к адресуемому информационному ресурсу. На каждом этапе маршрутизация проводится на основе символического имени адресуемого ресурса, содержащегося в HTTP-заголовке запроса, с помощью двух таблиц маршрутизации: "глобальной", связывающей Интернет-имя ресурса с адресом удаленного прокси-сервера, и "локальной", связывающей Интернет-имя ресурса с его адресом в частной сети.

Предположим, например, что в LAN_2 имеется Web-сервис **http://192.168.0.20/orgdb/srv.asmx**, а в LAN_3 имеется Web-сервис **http://192.168.0.20/orgdb/prs.asmx** (т. е. обоим сервисам соответствует один и тот же IP-адрес). Предположим, что эти сервисы анонсированы для "внешних" пользователей под символическими именами **http://myorg.kaluga.prs.ws** и **http://myorg.omsk.prs.ws** соответственно. Для организации доступа к сервисам по этим именам в локальную таблицу маршрутизации для LAN_2 следует внести строку, связывающую символическое имя с URL:

http://myorg.kaluga.prs.ws
http://192.168.0.20/orgdb/srv.asmx,

а в таблицу маршрутизации для LAN_3 — строку:

http://myorg.omsk.prs.ws
http://192.168.0.20/orgdb/prs.asmx

Предположим, что прокси-серверу P_2 соответствует IP-адрес 193.232.208.2 в глобальной сети, а прокси-серверу P_3 — IP-адрес 193.232.208.3. В этом случае в глобальную таблицу маршрутизации должны быть добавлены две строки, связывающие символические имена с адресами ПС:

http://myorg.kaluga.* 193.232.208.2
http://myorg.omsk.* 193.232.208.3

Легко заметить, что в обеих строках использованы усеченные символические имена, оканчивающиеся символом "*". Такая нотация обеспечивает маршрутизацию множества имен одной строкой. В данном случае, например, любой запрос к ресурсу с именем, начинающимся с http://myorg.kaluga., будет перенаправлен в LAN_2.

Отметим два важных свойства описанного подхода.

- Все прокси-серверы используют одну и ту же глобальную таблицу маршрутизации для обработки "исходящих" запросов.
- При изменении URL информационного ресурса в частной сети достаточно внести соответствующую коррекцию в локальную таблицу маршрутизации, чтобы никто из "внешних" клиентов не заметил этого изменения.

Совпадение IP-адресов у серверов в разных частных сетях (как в рассмотренном примере) в данном случае не создает проблем для адресации.

Принципы функционирования шлюзов "HTTP → HTTPS" и "HTTPS → HTTP"

Наиболее специфичная черта всякого прокси-сервера заключается в том, что он объединяет в себе функциональность клиента и сервера одновременно: в отношении обслуживаемого клиента он играет роль сервера, а в отношении обслуживающего сервера — роль клиента. Это в полной мере относится и к шлюзам "HTTP → HTTPS" и "HTTPS → HTTP".

Базовая структура шлюзов проиллюстрирована на рис. 3 и рис. 4. Их функционирование подчиняется следующим общим правилам.

- Процедура Listener обеспечивает "серверное" поведение шлюза: прослушивание входящих запросов на соединение на специальном выделенном порте (системный вызов *listen*), открытие нового (первичного) TCP-соединения с клиентом (вызов *accept*) и порождение параллельной программной нити для обработки каждого входящего соединения [10]. Вся дальнейшая обработка соединения выполняется в рамках программной нити.
- Блок "Обработчик первичных соединений" обеспечивает считывание HTTP-запроса (заголовка и тела запроса) из первичного соединения, разбор заголовка запроса и извлечение URL адресуемого ресурса из тела запроса.
- Блок "Обработчик вторичных соединений" обеспечивает "клиентское" поведение шлюза: запрос и установление нового (вторичного) соединения с адресуемым сервером с помощью системных вызовов *socket* и *connect* (процедура Connector); передача HTTP-запроса серверу; прием HTTP-ответа от сервера через вторичное соединение и, наконец, передача его клиенту через первичное.

Компоненты, обеспечивающие специфическое поведение шлюзов, направленные на создание защищенного HTTPS-тоннеля, выделены на рисунках темным цветом. К их числу относятся следующие:

- библиотека функций поддержки SSL/TLS (Open SSL);
- закрытый ключ шлюза, используемый для формирования электронной подписи в исходящих данных и расшифровки входящих;
- сертификат, содержащий открытый ключ шлюза, который передается шлюзу-контрагенту при построении защищенного HTTPS-канала;
- таблица маршрутизации информационных запросов.

Как видно из изложенного, принципы функционирования шлюза "HTTP → HTTPS" и шлюза "HTTPS → HTTP" в целом схожи. Основные различия заключаются в следующем.

- В шлюзе "HTTP → HTTPS" защищенный канал надстраивается над вторичным соединением с использованием сокета, полученного с помощью системного вызова *socket*. В шлюзе "HTTPS → HTTP" защищенный канал надстраивается над первичным соединением с использованием сокета, полученного с помощью системного вызова *accept* процедурой Listener.
- Шлюз "HTTPS → HTTP" выполняет не только маршрутизацию информационных запросов к серверным программам в частной сети, но также их авторизацию и проверку прав доступа на основе данных, содержащихся в полученных сертификатах контрагентов. Эта проверка может быть реализована, например, путем добавления в каждую строку локальной таблицы маршрутизации списка владельцев сертификатов (названий организаций), которым разрешен доступ к соответствующему информационному ресурсу.

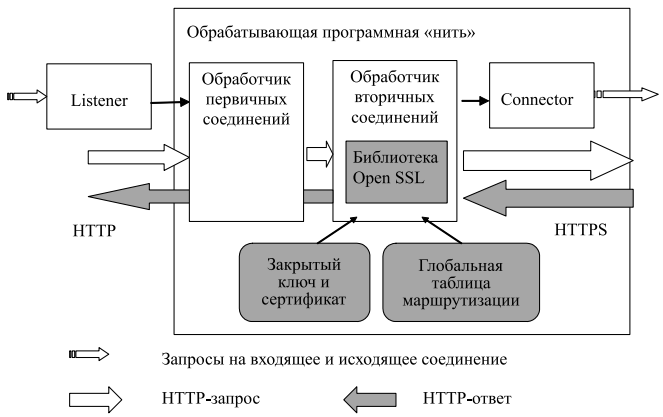

Рис. 3. Базовая структура шлюза "HTTP- > HTTPS"

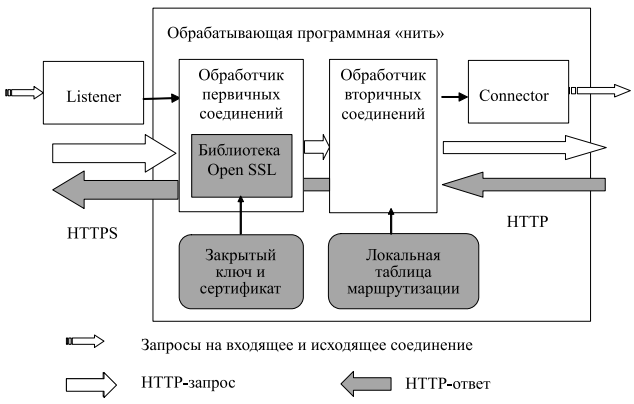

Рис. 4. Базовая структура шлюза "HTTPS- > HTTP"

Заключение

В таблице приведены преимущества и недостатки предлагаемого подхода в сравнении с каноническим методом поддержки HTTPS в прокси-серверах. Как видно из таблицы, основной недостаток подхода заключается в том, что в пределах частной локальной сети данные передаются в открытом виде. Поэтому его применение ограничено ситуациями, в которых частные сети можно рассматривать как "зоны доверия", а единственным источником угроз является глобальная сеть. В этих ситуациях "централизованная" защита данных в прокси-серверах, не зависящая от поведения отдельных клиентских и серверных программ, представляется оправданным решением.

Описанный подход реализован в форме многофункционального прокси-сервера, поддерживающего как канонический "прозрачный" режим, так и

Методы поддержки протокола HTTPS в прокси-серверах

Метод	Основные преимущества	Основные недостатки	Особенности реализации
Канонический (прозрачный режим)	Защита данных на всем пути от клиента до сервера	Невозможность контроля и обработки информационных запросов в прокси-серверах	Не требуется использования какой-либо программной поддержки SSL/TLS в прокси-серверах
HTTPS-тоннель	Возможность контроля и обработки информационных запросов в прокси-серверах	Защита данных только при информационном обмене между прокси-серверами (т.е. только в глобальной сети). Внутри частной сети данные не защищены	Требуется использование библиотек поддержки SSL/TLS в прокси-серверах

режимы шлюзов "HTTP → HTTPS" и "HTTPS → HTTP" (реализация проведена на языке C++ с использованием библиотеки OpenSSL в качестве программной поддержки технологии SSL/TLS). Опыт работы с ним показал, что описанный подход является достаточно общим: однажды созданная система защищенных тоннелей межсетевого взаимодействия может быть использована несколькими совершенно различными распределенными системами, работающими в мультисетевой среде.

Список литературы

1. **Кришнамурти Б., Рэксфорд Д.** Web протоколы. М.: Бином, 2010. 592 с.
2. **Шапошников И. В.** Web-сервисы Microsoft .NET. СПб.: БХВ-Петербург, 2002. 336 с.
3. **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.
4. **Танненбаум Э., Ван Стен М.** Распределенные системы. Принципы и парадигмы. СПб.: Питер, 2003, 877 с.
5. **Згоба А. И., Маркелов Д. В., Смирнов П. И.** Кибербезопасность: угрозы, вызовы, решения // Вопросы кибербезопасности. 2014. № 5. С. 30—38.
6. **Щеглов А. В.** Защита компьютерной информации от несанкционированного доступа. СПб.: Наука и техника, 2004. 384 с.
7. **Хант К.** TCP/IP. Сетевое администрирование. СПб.: Питер, 2007. 816 с.
8. **Андреев А. Г., Беззубов Е. Ю., Емельянов М. М., Кокорева О. И., Чекмарев А. Н.** Windows 2000: Server и Professional. СПб.: БХВ-Санкт-Петербург, 2000. 1056 с.
9. **Асратян Р. Э., Лебедев В. Н.** Применение технологии прокси-серверов в распределенных системах // Информационные технологии. 2013. № 6. С. 7—11.
10. **Снейдер Й.** Эффективное программирование TCP/IP. Библиотека программиста. СПб.: Символ-Плюс, 2002. 320 с.

R. E. Asratian, Leading Researcher, rea@ipu.ru, **V. N. Lebedev,** Head of Laboratory, lebvini@ipu.ru,
V. L. Orlov, Leading Researcher, ovl@ipu.ru, Institute of Control Science RAS

Creating a Secure Communication Channels Based on the use of the HTTPS in Proxy-Servers

The method of organization of secure interactions in distributed systems, operating in multi-network environment, is considered. The approach is based on the use of HTTPS protocol and proxy-server technology. The essence of approach consists in the organization of the protected HTTPS connections between proxy servers for safe transfer of information queries from one private local network in another through a global network while client and server components of distributed system are released from of information protection functions. In this approach client and server programs use HTTP protocol and proxy servers have additional functions of "HTTP- > HTTPS" and "HTTPS- > HTTP" gateways. The main advantage of approach over known (for example, VPN) consists in providing "high-level" means of HTTP queries routing and filtration the based on headers of HTTP queries, but not on separate IP packages. The application of approach is limited to situations in which private local networks can be considered as "a trust zone", and the only source of threats is the global network.

Keywords: *distributed systems, Internet technologies, network protocols, proxy servers, remote interactions, data security*

References

1. **Krishnamurti B., Reksford D.** *Web protokoly*. Moscow, Binom, 2010, 592 p.
2. **Shaposhnikov I. V.** *Web-servisy Microsoft* .NET. SPb: BHV-Peterburg, 2002. 336 p.
3. **Mak-Donal'd M., Shpushta M.** *Microsoft ASP.NET 3.5 s primerami na C# 2008 i Silverlight 2 dlja professionalov*. Moscow, Vil'jams, 2009, 1408 p.
4. **Tannenbaum E., Van Sten M.** *Raspredelennye sistemy. Principy i paradigmy*. SPb.: Piter, 2003, 877 p.
5. **Zgoba A. I., Markelov D. V., Smirnov P. I.** Kiberbezopasnost': ugrozy, vyzovy, reshenija, *Voprosy kiberbezopasnosti*, 2014, no. 5, pp. 30—38.
6. **Shheglov A. V.** *Zashhita komp'juternoj informacii ot nesankcionirovannogo dostupa*б Sankt-Peterburg, Nauka i tehnika, 2004, 384 p.
7. **Hant K.** *TCP/IP. Setevoe administrirovanie*, Sankt-Peterburg, Piter, 2007. 816 p.
8. **Andreev A. G., Bezzubov E. Ju., Emel'janov M. M., Kokoreva O. I., Chekmarev A. N.** *Windows 2000: Server i Professional*. SPb.: BHV-Sankt-Peterburg, 2000, 1056 p.
9. **Asratian R. E., Lebedev V. N.** Primenenie tehnologii proksi-serverov v raspredelennyh sistemah, *Informacionnye tehnologii*, 2013, no. 6, pp. 7—11.
10. **Snejder J.** *Effektivnoe programmirovanie TCP/IP*. Biblioteka programmista, Sankt-Peterburg, Simvol-Pljus, 2002, 320 p.