Д. А. Хафизов

ОГРАНИЧЕНИЯ ИСПОЛЬЗОВАНИЯ ПРОТОКОЛА ЗАЩИТЫ ТРАНСПОРТНОГО УРОВНЯ SSL ДЛЯ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ВЗАИМОДЕЙСТВИЯ WEB-СЕРВИСОВ

Эволюция архитектуры информационных систем приводит к образованию качественно новых подходов к их проектированию. Переход от клиент-серверных моделей к трехуровневым и многоуровневым архитектурам с целью получения дополнительной функциональности усложняет взаимодействие между компонентами. Следствием такого процесса является формирование новых методологий в области разработки корпоративных приложений.

Основной целью данной работы является выявление набора ограничений, которые необходимо учитывать при построении систем обеспечения безопасного взаимодействия в распределенной корпоративной информационной среде, построенной на базе web-сервисов. Анализ стратегических планов развития корпоративной информационной инфраструктуры на основе сформированных критериев может послужить основой для детализации требований к самой системе безопасности на этапе проектного обследования.

Проблемная область обеспечения безопасности сервис-ориентированной архитектуры

В настоящее время при реализации информационных систем все большее значение приобретает подход, основанный на концепции сервис-ориентированной архитектуры (SOA, service-oriented architecture) [1]. В его основе лежит представление об информационной системе как о системе с распределенной, гибкой и модульной архитектурой, обладающей возможностью многократного использования кода. Существует несколько технологических платформ от ведущих производителей программного обеспечения, позволяющих, руководствуясь отмеченными принципами, создавать приложения уровня предприятия. Значительное место при создании такого рода приложений играют web-технологии. Долгое время большинство web-приложений строилось по принципам трехуровневой, или трехзвенной, архитектуры (рис. 1).

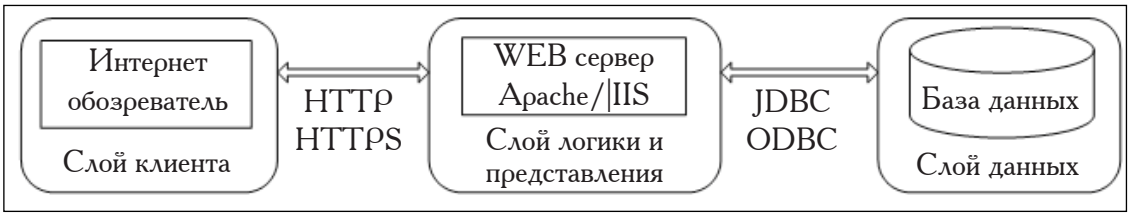

Рис. 1. Трехзвенная архитектура web-приложений

В представленной трехзвенной архитектуре на каждом слое обработки информации существуют свои методы и средства защиты. На уровне данных — средства из арсенала защиты СУБД, на уровне логики и представления — механизмы защиты серверов приложений, а на уровне клиента — средства обеспечения безопасности клиентских машин и механизмы обеспечения безопасного взаимодействия с серверами.

С развитием технологических возможностей в качестве основных компонентов для реализации корпоративных информационных систем стали использоваться web-сервисы. Это предоставляет значительные удобства с точки зрения взаимодействия между модулями реализуемой системы, поскольку они в этом случае имеют стандартные унифицированные интерфейсы для своего собственного описания и описания способов взаимодействия с ними. К наиболее популярным спецификациям, лежащим в основе web-сервисов, относятся WSDL (Web Services Description Language) [2] и SOAP (Simple Object Access Protocol) [3]. WSDL — язык, позволяющий задать унифицированный программный

интерфейс для взаимодействия с web-сервисами. SOAP — протокол обмена сообщениями, которые, в свою очередь, представлены XML-документами специального вида. Отметим, что в дальнейшем под web-сервисами мы будем понимать именно конечные программные модули, или, как их еще называют, конечные точки (web-службы), например, так же как и в ISO/IEC 29362.

Важной особенностью сервис-ориентированной архитектуры в этом случае является наличие открытых интерфейсов взаимодействия, а также возможность взаимодействовать по открытым коммуникационным каналам (HTTP) через межсетевые экраны [4]. Поэтому в рамках сервис-ориентированного подхода обязательно следует уделить внимание вопросам организации надежного и безопасного взаимодействия между компонентами информационных систем.

Всегда надо помнить о том, что значимыми инфраструктурными элементами являются операционные системы сами по себе, а также средства, предоставляемые ими в виде различных программных интерфейсов (API) и встроенных служб. Все эти логические уровни и средства обеспечения безопасности на каждом из них необходимо обязательно принимать во внимание при построении распределенной среды web-сервисов, поскольку они являются фундаментом и необходимым условием для обеспечения безопасности сервис-ориентированной архитектуры в целом.

Дальнейшее рассмотрение взаимного влияния технологий на информационную архитектуру и архитектурных принципов на выбор технологической платформы проведем в соответствии со следующим набором требований:

- конфиденциальность (confidentiality — безопасность передачи данных);
- целостность (integrity — сохранность данных);
- невозможность отказа от авторства (nonrepudiation — неаннулируемость);
- доступность (availability).

Сравним, как набор этих требований удовлетворяется относительно web-приложений и web-сервисов. Следует отметить, что с технологической точки зрения между web-сервисами и web-приложениями много общего. Они, как правило, могут выполняться на одних и тех же платформах или серверах приложений (Microsoft IIS, Apache, Tomcat, Sun Application Server, OC4J, IBM WebSphere) и взаимодействовать по одним и тем же протоколам (POP3, IMAP, XMPP, SMTP и HTTP). Но, с другой стороны, важно понимать, что, несмотря на некоторую внешнюю технологическую схожесть, сервисы, тем не менее, требуют собственной среды выполнения, например, такой как Axis или Windows Communication Foundation. Что касается особенности их использования, то логика вызова web-сервисов может в некоторых случаях принципиально отличаться от web-сайтов и web-приложений. Например, обычные web-сайты предназначены главным образом для людей — клиентов, нуждающихся в графическом пользовательском интерфейсе. Web-сервисы в качестве конечных точек предназначены для использования другой программой и, как правило, даже не клиентской. Зачастую это может быть какой-либо другой сервис или даже интеграционный сервер, осуществляющий транзакции через несколько сервисов. Web-сервисы в общем случае должны использоваться в качестве составных модулей (компонентов) композитного приложения. Эти различия между web-сервисами и web-приложениями и являются принципиальными при рассмотрении вопросов реализации механизмов их безопасности. Для того чтобы говорить о безопасности web-сервисов, необходимо также понимать особенности планируемого использования их в корпоративной информационной среде.

Обеспечения безопасности взаимодействия web-сервисов на примере протокола SSL

Для организации безопасного соединения клиента и сервера приложений в подходе с трехзвенной архитектурой, реализованной на основе прямого взаимодействия (по методу точка—точка или потребитель—поставщик), традиционно применяется хорошо известный протокол SSL [5]. Этот протокол — распространенный механизм защиты взаимодействия между клиентом и сервером

приложений. SSL использует шифрование с отрытым и закрытым ключом, и с его помощью можно организовать безопасный обмен данными и подтверждение подлинности отправителя и получателя. В качестве протокола защиты взаимодействия точка—точка мы рассмотрим пока SSL, а не более современный TLS, потому что SSL является характерным представителем и принципы его использования распространимы на его модифицированные разновидности и протоколы более низкого уровня (TLS; JSSE; SSH; IPSEC; SOCKS). Важно отметить, что выбор для нашего рассмотрения именно протокола SSL не нанесет большого ущерба пониманию сути и степени его влияния на архитектуру информационных систем в целом. К тому же SSL наиболее известен и часто встречается как в корпоративных информационных средах, где он обладает существенной инструментальной поддержкой на базе продуктов различных поставщиков [5], так и в публичной сети Интернет, например, при взаимодействии пользователей с некоторыми сайтами и различными бесплатными службами, требующими обеспечения закрытой передачи информации.

Реализация защиты web-сервисов через SSL потребовала бы тех же усилий, что и реализация защиты обычного web-приложения, за исключением того, что, как правило, взаимодействие пользователей с порталами и web-приложениями осуществляется через стандартные браузеры. Поддержка сертификатов и соединений по HTTPS в них уже встроена, в то время как взаимодействие с web-сервисом напрямую, скорее всего, будет осуществляться с применением специализированного клиентского приложения. При проектировании и разработке приложения в нем потребуется реализация соединения по протоколу SSL и поддержка работы с сертификатами. Поскольку приложения делаются разными разработчиками на разных языках и с применением разных технологий, это приведет к множеству вариантов реализации и сильной дифференциации решений одной и той же задачи.

Рассмотрим другой вариант организации web-приложения, который в большей мере соответствует основной идее перехода на web-сервисы, а именно декомпозиция и многократное использование существующего кода. В соответствии с этой концепцией множество web-сервисов могут быть скомбинированы в композитные приложения, такие, что отдельные разработчики, которые создавали свои специализированные сервисы, не имели бы даже представления о конечных вариантах системы. Безусловно, это негативно влияет на их общую концепцию безопасности. Принципиальная архитектура такой системы может выглядеть, например, как представлено на рисунке 2.

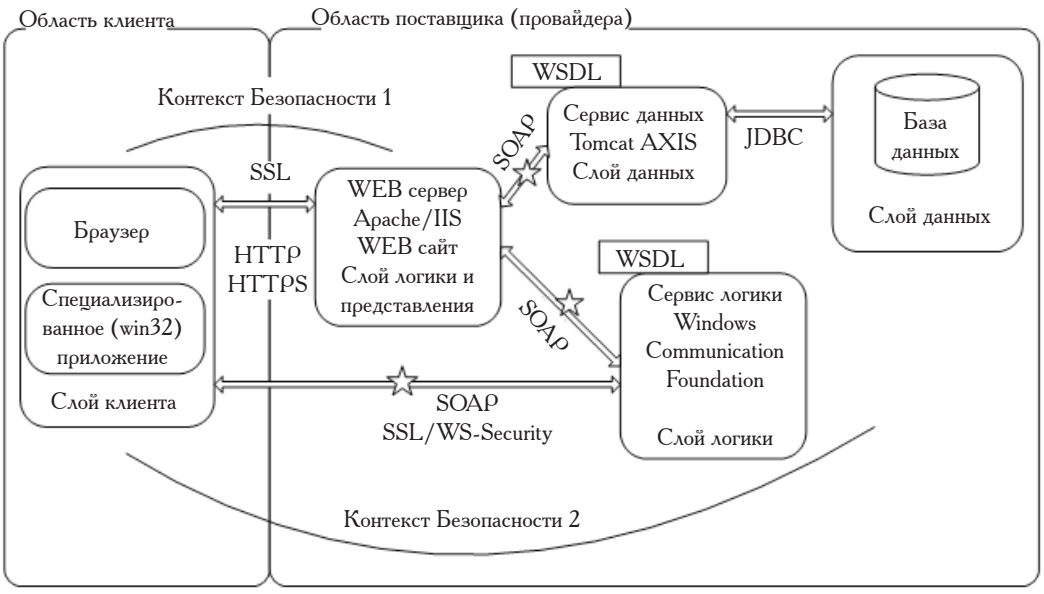

Рис. 2. Непрямой доступ к web-сервису. Сервис-ориентированный подход

Предположим, что существует web-сервис, который может быть доступен пользователю не напрямую, а по следующему сценарию. Сначала клиент получает доступ к пользовательскому интерфейсу в виде web-сайта через защищенное SSL-соединение. Web-сайт, в свою очередь, имеет подсистему, которая сама вызывает необходимый ей удаленный метод сервиса. Он по аналогии с объектно-ориентированной парадигмой в парадигме конечных точек (web-служб), так же как метод класса, выполняет конкретную атомарную операцию или функцию и возвращает конечный результат определенного типа данных. Получается вариант, когда серверная программа или web-сервис вызывают другой сервис, так что этот процесс не связан с браузером или даже протоколом HTTP. Более того, этот процесс может повторяться не единожды и быть сквозным, проходя через несколько сервисов, формируя при этом соответствующие транзакции. При реализации такого сценария возникает закономерный вопрос: каким именно образом следует защищать второе соединение между web-сервером и web-сервисом?

Следует отметить, что таким образом может быть расположено большинство web-сервисов, формирующих конечные композитные приложения. На рисунке 2 дополнительно представлено прямое соединение клиентского приложения с сервисом по методу точка—точка, о котором упоминалось ранее. Надо заметить, что применение для его защиты SSL уже не вполне однозначно.

В дальнейшем будем называть контекстом безопасности совокупность значимых для принятия решений внешних условий по отношению к логическому соединению, т. е. пути передачи информации от одной конечной точки до другой. На рисунке 2 мы видим, что для одного и того же логического соединения возможны разные физические соединения, с разными внешними условиями и принципами организации самих соединений, или, другими словами, разными контекстами безопасности. Соединения, соответствующие второму контексту безопасности, на рисунке помечены звездочками. В итоге в предложенном сценарии имеем, как минимум, два контекста безопасности, нуждающихся в защите: между клиентом и web-сайтом, между клиентом и web-сервисом.

Вывод: второй контекст безопасности, относящийся непосредственно к безопасности SOAP-сообщений, на которых построен обмен информацией с web-сервисами как запроса, так и ответа, должен быть достоверен более чем для одного клиент-серверного соединения. Иначе говоря, есть необходимость в обеспечении постоянной, независимой от соединений собственной безопасности SOAP-сообщений. В то время как SSL шифрует каждый поток данных целиком и независимо от остальных, он не поддерживает безопасности от «начала и до конца» или сквозной безопасности через несколько приложений или компаний, задающих разные контексты безопасности. Используя этот протокол, сложно установить, где он позволяет открыть данные между пользователем и провайдером WEB-сервиса. Протокол SSL имеет ряд принципиальных ограничений для защиты WEB-сервисов, которые заключаются в следующем:

1. SSL обеспечивает безопасность соединений точка—точка или операций между конечными точками (а не приложениями). Для распределенных web-приложений необходима безопасность от начала и до конца, при которой может существовать множество промежуточных звеньев между двумя соединяющимися точками. В среде web-сервисов для XML-документа, проходящего через множество посредников при интеграции большого числа приложений и сервисов, будет сложно организовать все соединения на разных платформах, операционных системах и серверах приложений с использованием одного лишь SSL. В случае усложнения среды и увеличения числа взаимодействующих компонентов по безопасному каналу увеличивается сложность контроля и управления средой в целом. Все это порождает проблему масштабируемости.

2. SSL работает на транспортном уровне и не работает на уровне сообщений. Другими словами, сообщения защищены только пока они находятся в состоянии перемещения. А это значит, что, однажды сохранив SOAP-сообщение, позже вы уже не сможете доказать, что это то же самое сообщение и что оно не было изменено. SSL не поддерживает шифрование SOAP-сообщений в файлах или базе данных.

3. SSL не поддерживает невозможность отказа от авторства на уровне SOAP-сообщений. Используя SSL, взаимодействующие партнеры не могут доказать, что другая сторона выполнит свою часть транзакции. Это значит, что SSL не поддерживает сквозной аудит в течение всего времени от запроса сервиса до ответа.

4. SSL шифрует весь трафик между взаимодействующими компонентами. При его использовании происходит криптографическое преобразование даже той информации, которая вряд ли представляет для кого-либо интерес. SSL не поддерживает выборочного, или, как еще говорят, интеллектуального, шифрования и подписи элементов XML-документа. При получении большого XML-документа вам может понадобиться, например, подписать и зашифровать информацию для одного приложения только по кредитной карте, а для другого — только информацию по адресам заказчиков, а это сложно сделать, используя один лишь SSL. Он изначально был спроектирован для обеспечения безопасности на транспортном уровне, в отличие от уровня сообщений, как это нужно для web-сервисов и протокола SOAP. Накладные расходы по расшифровке и дешифровке всего сообщения целиком для разных приложений однозначно нецелесообразны. К тому же непонятно, как быть с тем фактом, что для кого-то часть информации должна быть открыта, а для кого-то закрыта. Может оказаться так, что разные части одного и того же сообщения должны быть для разных участников процесса обмена сообщениями закрыты. В случае больших распределенных систем это опять же приведет к сложности управления такой инфраструктурой и снижению производительности.

5. SSL-трафик беспрепятственно проходит через межсетевой экран, а поскольку он зашифрован, то никаким образом нельзя просмотреть его содержание, например, на предмет вредоносных данных для аутентифицированных и авторизовавшихся клиентов. Этот трафик также нельзя отфильтровать программным образом или провести маршрутизацию SOAP-сообщений.

Заключение

Несмотря на отмеченные недостатки, SSL долгое время был основным способом защиты взаимодействия с web-сервисами. Это проверенный и не очень сложный в эксплуатации вариант обеспечения безопасного взаимодействия. В настоящее время он широко используется в случае взаимодействия с web-сервисами по методу REST. Его применение оправданно при необходимости защиты только по методу точка—точка и в средах с небольшим количеством сервисов. Стоит задуматься о возможностях организации безопасного взаимодействия на основе SSL также и в средах с технологиями от одного производителя, где реализация была бы однотипной и согласованной.

Теоретически можно попробовать адаптировать SSL для нужд защиты сервисов, даже в сильно распределенной среде, т. е. в среде, которая соответствует информационной инфраструктуре крупных предприятий. Хотя в то же время надо понимать, что указанные ограничения являются весьма существенными в том числе и для крупных компаний, имеющих территориально-распределенную структуру. Информационная корпоративная среда таких компаний может не ограничиваться сотней, а порой и тысячей используемых сервисов. Для того чтобы количество связей в сети не разрасталось, приводя к потере контроля над всей инфраструктурой, сервисная архитектура требует дополнительных технических расширений и новых механизмов и протоколов защиты взаимодействия, а также выработки методик их использования.

СПИСОК ЛИТЕРАТУРЫ:

1. Reference Model for Service Oriented Architecture // OASIS, 2006. URL: http://www.oasis-open.org/committees/download.php/16587/wd-soa-rm-cd1ED.pdf (дата обращения 12.05.2009).
2. Web Services Description Language // W3C Note, 2001. URL: http://www.w3.org/TR/wsdl (дата обращения 12.05.2009).
3. Simple Object Access Protocol // W3C Note, 2000. URL: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/ (дата обращения 12.05.2009).

4. *Лукацкий А. В.* Безопасность SOA // ИКС. 2008. № 4. С. 60.
5. *Bimal S., Aya Z.* What is SSL and why should I care // IBM developerWorks, 2008. URL: http://www.ibm.com/developerworks/autonomic/library/ac-iscssl1/ (дата обращения 12.05.2009).
6. Web Services Security: SOAP Message Security 1.0 // OASIS, 2004. URL: http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf (дата обращения 12.05.2009).