А. И. Гусева, С. А. Бачурин

ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ В ДИНАМИЧЕСКИХ МОДУЛЬНЫХ СИСТЕМАХ

Непрерывный рост масштабов программного обеспечения приводит к значительному увеличению его сложности, в том числе в такой высокорисковой области, как управление безопасностью. Это означает быстрое удорожание стоимости владения и поддержки. В первую очередь это касается отраслевых предприятий России в силу их размеров, распределенности, меняющейся организационной структуры. Реорганизация информационного пространства с помощью сервис-ориентированных решений позволяет значительно увеличить эффективность его использования [1]. Спецификация *OSGi* (*Open System Gateway Initiative*) является одним из наиболее успешных и популярных на Западе подходов к построению таких систем и послужила хорошей отправной точкой в проведенной разработке.

Согласно спецификации, разработанная сервисная платформа разворачивается в среде *Java*. Это обеспечивает должный уровень мобильности для поддержки продуктов на различных устройствах. Приложения конструируются из небольших, повторно используемых и взаимодействующих друг с другом пакетов. При этом одна из ключевых задач, выполняемых платформой, — обеспечение тонкого и гибкого контроля над безопасностью взаимодействия пакетов друг с другом и с системными ресурсами (файлы, оперативная память, сетевые порты) [2]. Такой контроль возможен благодаря тому, что модель безопасности *OSGi* основана на спецификации *Java 2*, т. е. регламентируется выполнение всех проверок безопасности в соответствии с требованиями, предъявляемыми архитектурой безопасности *Java 2* [3].

Платформа выполняет аутентификацию пакетов двумя способами:

- анализ расположения (анализируется имя, данное пакету во время его разворачивания администратором. Имя является постоянным уникальным идентификаторам разворачиваемого сервиса. Обычно в качестве такого имени выбирается *URL*-адрес упаковочного файла);
- анализ подписи владельца пакета (пакеты должны быть подписаны электронной цифровой подписью (ЭЦП)).

При этом управление разрешениями на самом верхнем уровне производится с помощью двух служб платформы:

- сервис управления разрешениями — позволяет управлять разрешениями, выдаваемыми на основании расположения пакета;
- сервис управления условными разрешениями — позволяет управлять разрешениями, выдаваемыми на основе сложной модели условий. При этом условия строятся на данных о расположении пакета и его ЭЦП.

Следует заметить, что в отличие от идентификатора расположения, который является уникальным для каждого пакета, издатель, поставивший ЭЦП, может быть общим для нескольких пакетов. Это позволяет задавать политику безопасности для группы пакетов. По сути, постановка цифровой подписи на документ привносит в него дополнительное свойство безопасности, которое позволяет:

- аутентифицировать лицо, подписавшее документ;
- гарантировать, что подписанный документ не был изменен после того, как на него была поставлена подпись издателя.

Поэтому управление политикой происходит на основании того, что платформа ассоциирует пакет с его подписью и далее оперирует с издателем подписи для задания общих разрешений.

Такая модель делегирования полномочий позволяет администратору *OSGi*-платформы единожды определить политику безопасности для поставщика сервисов, после чего поставщик может создавать упаковочные файлы и разворачивать их на платформе самостоятельно. Подписанные поставщиком файлы будут обладать заданными полномочиями, не требуя дополнительного вмешательства администратора.

В качестве механизма для создания ЭЦП выбрана криптография с открытым ключом, которая базируется на системе с двумя математически связанными ключами. Открытый ключ является общедоступным, т. е. может свободно распространяться. Обычно для этого используются сертификаты. Закрытый ключ держится в секрете. Пакеты подписываются закрытым ключом и могут быть верифицированы только с помощью соответствующего открытого ключа [4]. Пакет представляется упаковочным файлом, который может быть подписан даже несколькими издателями.

Для генерации подписи был выбран наиболее широко распространенный алгоритм открытого ключа *RSA* (*Rivest, Shamir and Adleman*). Файлы алгоритма хранятся в формате, определенном в спецификации *PKCS#7*, и содержат ЭЦП и сертификат для использованного открытого ключа. Открытый ключ используется для проведения верификации ЭЦП и соответствующего **.SF*-файла, содержащего дайджест для *Manifest*-файла. Сертификат является подписанным документом, содержащим открытый ключ, т. е. обеспечивает гарантии надежности открытого ключа. Основными элементами сертификата являются:

- Название (Subject Name) — уникальный идентификатор объекта, который подвергается сертификации. В случае человека он может включать имя, национальность, e-mail, организацию и отдел внутри организации. Идентификатор должен соответствовать формату Distinguished Name (DN) [9].
- Имя издателя — это DN-идентификатор лица, подписавшего сертификат.
- Поле расширений — используется для включения в сертификат дополнительных данных, таких как отпечатки пальцев, фотографии, паспортные данные.
- Поле открытого ключа — содержит удостоверяемый сертификатом открытый ключ и имя использованного алгоритма шифрования.
- Срок действия — содержит дату выдачи и дату истечения срока действия сертификата.
- Поле подписи — содержит подпись, которая удостоверяет верность всех перечисленных выше пунктов. Получатель сертификата может сравнить подпись с набором надежных (известных ему) подписей. Если получатель доверяет издателю, поставившему подпись, он может доверять данным, указанным в сертификате.

Сертификаты распространяются открыто, так как они не содержат никакой секретной информации. И это позволяет использовать их в файлах ресурсов ЭЦП. Очевидно, что сертификат, который содержит открытый ключ и зашифрован парным ему закрытым ключом, не может считаться надежным и является сертификатом только фиктивно. Злоумышленник легко создаст такой сертификат с любым содержимым. Получатель же сертификата может только проверить, что сертификат подписан действительно обладателем открытого ключа и в этом смысле не является поддельным. Однако, до того как данные сертификата можно будет считать надежными, необходимо провести аутентификацию самого сертификата. А для этого необходимо было определить модель доверия.

Выбранная модель доверия предполагает помещение сертификатов в репозиторий. И далее любой сертификат, находящийся в репозитории, обрабатывается как подлинно достоверный. Однако такое решение привело бы к очень быстрому росту репозитория, так как не обеспечивает должного масштабирования. Поэтому вводится процесс делегирования, который состоит в том, что новый сертификат подписывается уже доверенным, и эта операция может быть повторена несколько раз. Процесс делегирования, таким образом, предстает в виде цепочки сертификатов.

При этом все сертификаты цепочки передаются в *PKCS#7*-файле: если один из сертификатов цепочки будет найден в репозитории, то все сертификаты, входящие в цепочку, также считаются достоверными, при условии, конечно, что все они являются валидными. Эта модель очень легко масштабируется, потому что только несколько сертификатов надежных подписчиков должны храниться в репозитории.

Наконец, пора сделать выводы. Безопасность, предоставляемую разработанной платформой, можно охарактеризовать следующим образом:

Модели безопасности *Java 2* и механизм ЭЦП широко распространены и доказали свою состоятельность. А так как это основные механизмы, используемые платформой, можно уверенно говорить о ее надежности и предсказуемости.

- Применение ЭЦП для подписи пакетов позволяет задавать для них групповую политику безопасности, используя информацию об издателе.
- Использование уникальных идентификаторов пакетов позволяет получить необходимый уровень детализации, когда требуется задать особую политику безопасности специфическому пакету.
- Использование сертификатов для подписи пакетов обеспечивает хорошие условия для масштабирования приложения.

Хочется также отметить, что разработанная платформа является расширением открытой и распространяемой по лицензии *EPL* платформы *Equinox* и вводит новый тип *OSGi*-пакетов, который используется для разработки *GUI*-приложений [5]. Этот тип характеризуется выделением трех компонентов пакета, в соответствии с архитектурой MVC (*Model-View-Controller*). При этом видимость компонентов *Model* и *View* строго ограничена содержащим их пакетом, а компонент *Controller* может быть экспортирован и использоваться в других пакетах приложения. За всеми тремя компонентами сохраняется общий домен политики безопасности.

СПИСОК ЛИТЕРАТУРЫ:

1. Интеграция на основе SOA в среде унаследованных приложений. URL: http://www.oracle.com/global/ru/oramag/dec2008/w_dev_soa.html.
2. OSGi Technology. URL: http://www.osgi.org/About/Technology.
3. Java Security Architecture. URL: http://java.sun.com/javase/6/docs/technotes/guides/security/spec/security-spec.doc2.html.
4. *Хорстманн К., Корнелл Г.* Java 2. Библиотека профессионала. Т. 2.: Тонкости программирования. Пер. 8-е изд. М.: Вильямс, 2009. – 992 с.
5. *Бачурин С. А.* Проектирование, разработка и сопровождение масштабных программных приложений на Java с помощью автономных модулей // Научная сессия МИФИ-2008. Сборник научных трудов. М.: МИФИ, 2008. Т. 11. С. 42–43.