Уважаемые коллеги, добрый день.

Я представляю реальный проект, который используется в нашем банке. Это программа предназначена для обмена данными между электронными торговыми площадками(ЭТП) и банком в рамках федерального закона 44-ФЗ.

Смысл закона простой: государственные организации могут купить товары или заказать услуги только после проведения конкурсов. ЭТП публикуют условия, организации и граждане регистрируются на площадках и участвуют в конкурсах.

Если вы думаете, что там только огромные монстры и корпорации, то это не так. Самые обычные люди тоже участвуют. Уборщицы, например.

Вся эта деятельность как-то велась до прошлого лета, пока Минфин не вмешался и не сказал, что «палочки должны быть попендикулярны», форматы обмена данных должны быть согласованы, банки и площадки посчитаны, а если кого-то что-то не устраивает, то можно уйти с рынка и не обижайтесь. Кстати, на все у вас три месяца, а форматов обмена нет.

Я в это время проводил свой досуг в Юрмале.

Банков 18, площадок 8, у каждого свое мнение о том, как нужно работать. На наше счастье Сбербанк нам помог организоваться, предложил использовать его форматы, остальные вдоволь наоравшись согласились.

Итак приложение, которое нам нужно было спроектировать должно было делать следующее:

* Выступая в роли клиента https, зайти по определенному url-адресу по протоколу TLS получить данные из ЭТП в формате xml. Проверить электронную цифровую подпись(ЭЦП) в формате CryproPro, которой было подписано тело полученного сообщения. Проверить соответствие формату обмена, если все хорошо, то передать это сообщение в АБС (автоматизированную банковскую систему).
* Выступая в роли web-сервиса дать возможность ЭТП получать ответы на свои запросы. Получать ответы от АБС, ставить на них банковсую ЭЦП.

Итого получили следующие компоненты системы:

* Сканер входного каталога
* Сервис для подписания/проверки подписи полученных сообщений
* Сервис отправки данных на площадки
* Контроллер, обслуживающий запросы площадок
* Сервис проверки сообщений по XSD-схеме

Разумеется, это не все. Где-то далеко стоит банковская часть, которая обрабатывает запросы ЭТП и формирует ответы. Кстати, это все сверху прижато очень жесткими временными рамками.

На всякий случай скажу, что АБС выполняет разные другие задачи, помимо этой и является довольно нагруженной системой.

Это все я рассказал для того, чтобы вы поняли в какой тяжелой ситуации мы оказались.

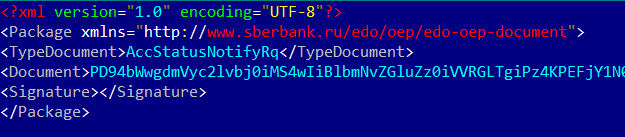
В качестве технологического стека был выбран Spring\SpringBoot\CryptoPro\Oracle\ PL\SQL

Начнем, пожалуй со сканера входного каталога. Да, это не 21 век, да можно поставить брокер сообщений, можно поставить Кафку, внедрить шину данных, можно даже оформить документацию по ГОСТу. Только вот срок внедрения отодвинуть нельзя, поэтому говорим АБС, что ждем от нее файлов во входном каталоге.

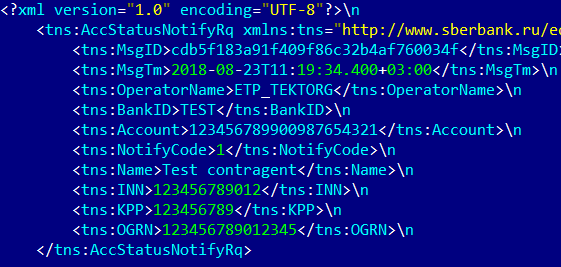
Ну и чтобы не было совсем грустно, делаем наш сканер наследником Thread, и заводим отдельный поток на каждую площадку.



Давайте разберемся, что мы отправляем на площадки. Вот так выглядит файл с транспортным пакетом. Внутри него есть тег с типом запроса, сам запрос в base64 и подпись в формате того самого страшного ГОСТа 34.10-2012.



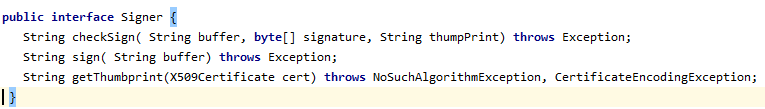
Так выглядит запрос в дешифрованном виде. Вот его и надо подписывать в расшифрованном виде, разумеется. Потом нужно записать подпись в тег Signature.



АБС в нашем банке умеет многое, но вот как-то работать с CryptoPro не нужно было. Пожалуй, реализуем это в нашей программе.

На наше счастье, CryproPro снабжено большим количеством примеров. В них есть почти все, но … не все.

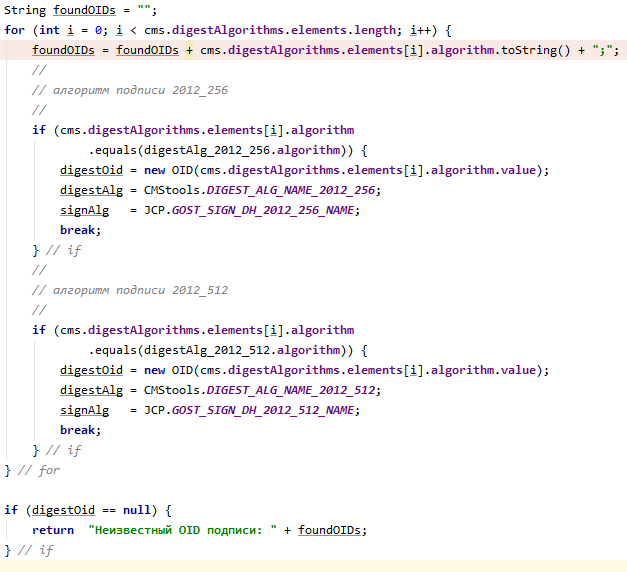
Сделаем сервис, который умеет обрабатывать переданный буфер на основе ГОСТ 34.10-2012, ставить, проверять подпись и получать отпечаток подписи.



В основу берем файлы из каталога CMS\_samples из примеров CryproPro. Включаем … не работает. Причин этого две:

* ЭТП не видят разницу между ГОСТ 34.10-2001 и ГОСТ 34.10-2012 и присылают файлы в устаревшем 2001 ГОСТе с комментарием «А чего такова???»
* В ГОСТ 34.10-2012 длина ключа может быть 256 бит или 512 бит, в протоколе обмена этот вопрос никак не урегулирован и в нашей версии появляются строки, которые все это учитывают

Проверяем длину ключа, проверяем OIDы подписи. Площадки бомбардируют письмами с требованиями принимать ГОСТ 34.10-2001, мы отвечаем им сообщением согласно формату «Неизвестный OID подписи». Через несколько недель крики стихают, и площадки выпускают ключи на современном 2012 ГОСТе.



Если интересно, можете побродить по контейнеру с ключом <https://lapo.it/asn1js/>

Почитать про ключевые контейнеры <https://habr.com/ru/post/256367/>

На самом деле все это не просто, тянет на отдельную профессию. У меня были и другие проекты с CryptoPro, поэтому все было не так ужасно.

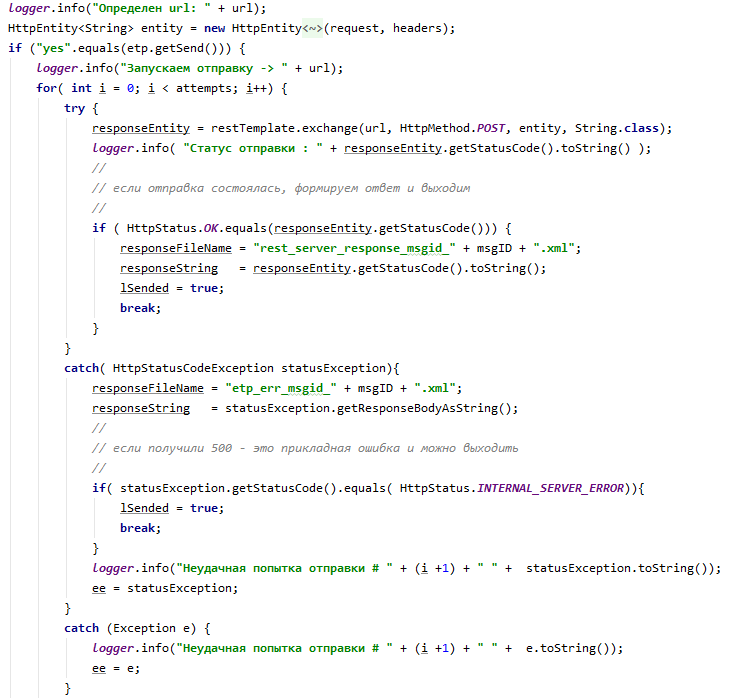
Научились ставить подпись, давайте теперь запишем ее в тег Signature. Тут тоже все просто. Преобразуем полученный файл dom-документ, читаем содержимое тега Document, снимаем с него base64, подписываем, и полученную подпись, помещаем в тег Signature.

Пирожок готов!



Переходим к отправке данных на площадку. Берем самый обычный restTemplate и отправляем методом POST. Вроде просто все, кроме обработки ошибок. Если что не так, то получаем 500 ошибку. Так положено по протоколу. Сохраняем ошибку в файл, АБС разберется.

Единственное дополнение: согласно правилам обмена нужно отправлять несколько раз. В случае прикладной ошибке возвращается 500 статус, в противном случае продолжаем отправки. Spring Retry приделать не удалось, поэтому вручную отслеживаем определенное количество попыток и отсчитываем паузу между ними.



Переходим к контроллеру для обслуживания запросов площадок банку. Саму обработку запроса как положено выносим в сервис dataPackageConsumer. Получаем строку в RequestBody, потому что в случае ошибочного сообщения через ExceptionHandler не получилось добиться правильного ответа в сторону площадки.



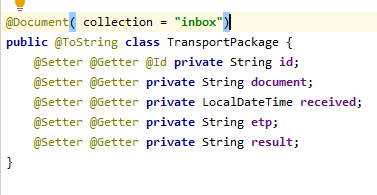
Сервис dataPackageConsumer делает много разных вещей, но самое главное он проверяет подпись полученного пакета. Мы храним у себя отпечаток сертификата открытого ключа и сравниваем его с тем, что пришло в подписи. Ну и, конечно, делаем обход всей этой крипто-мути, памятуя о вольностях площадок.



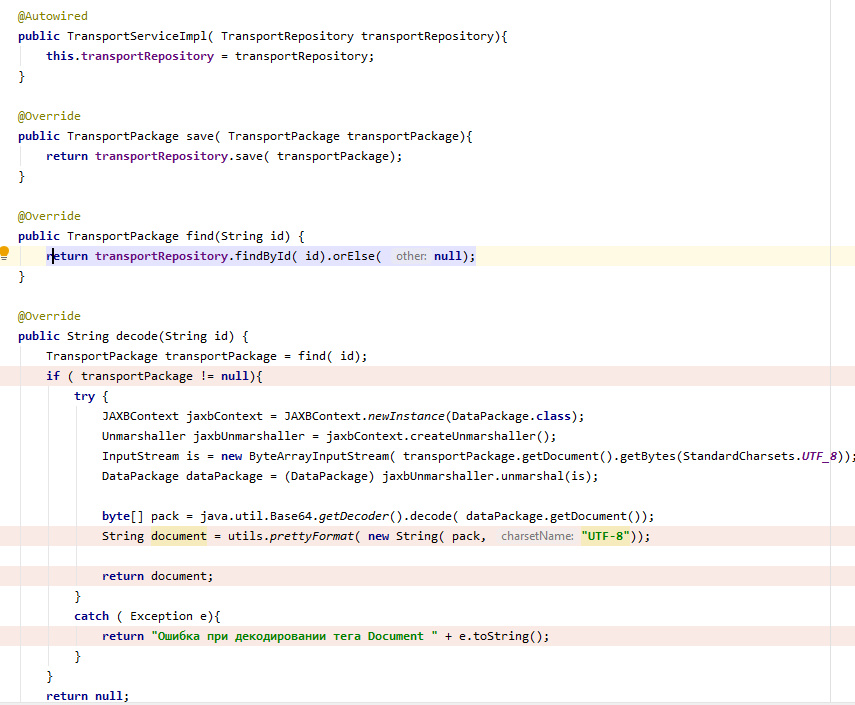
Добавляем несколько совершенно лишних технологий, но требуемых по правилам выпускного проекта.

1. Прежде всего добавим работу с базой данных. Документы, полученные из площадок, будем хранить в MongoDB. После oracle любая база дает +100 к карме.

Вот класс для хранения документов. Lombook нам в помощь



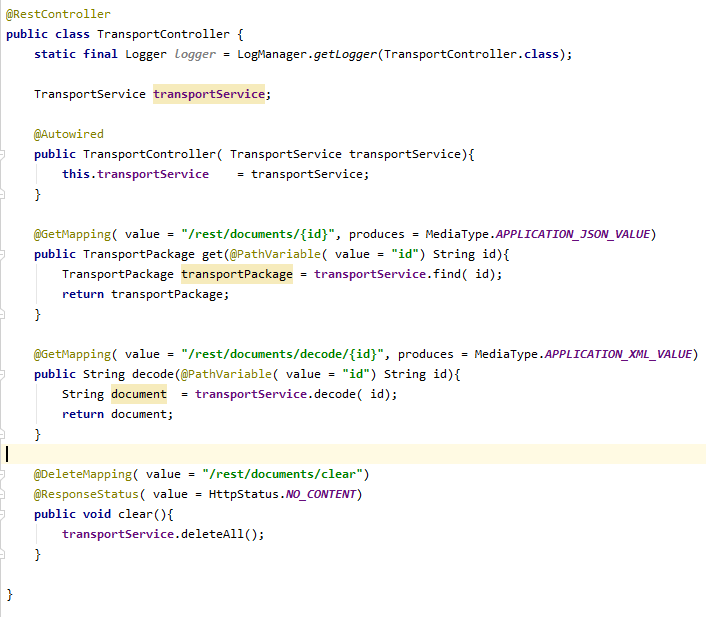
Сервис для поиска полученного пакета и декодирования содержимого. Маршаллинг наше все.



1. Spring Shell. Сделаем три команды: поиск документов, показ декодированного содержимого и удаление документов



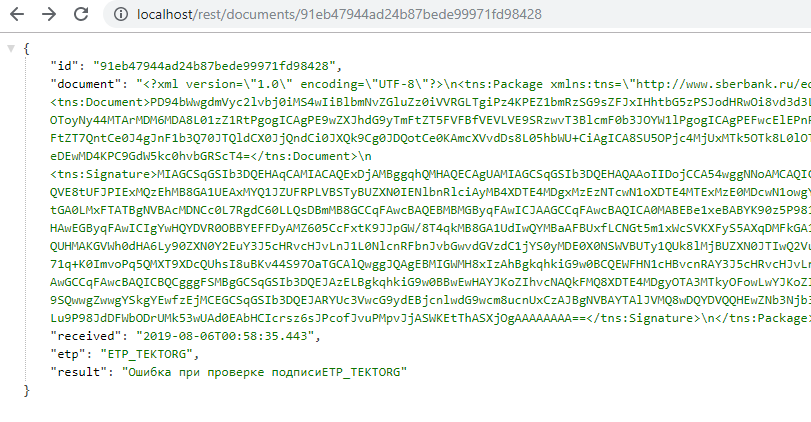
1. Выставляем тот же функционал на уровень web



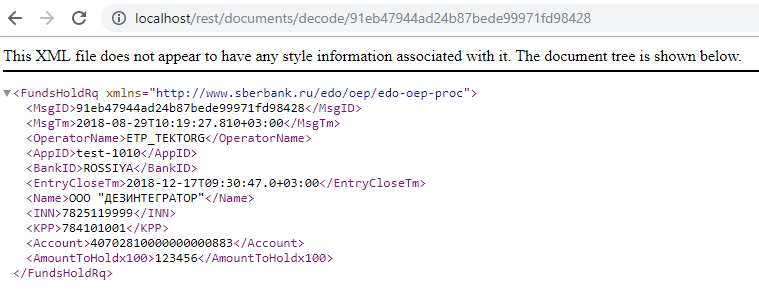
1. Закрываем url c документами Spring Security. Основной url /callback закрывать не нужно, так на него можно попасть только с определенных внешних ip-адресов предъявив сертификат. Актуатор, конечно, тоже открыт.



Вот и результат обращения к нашему сервису.



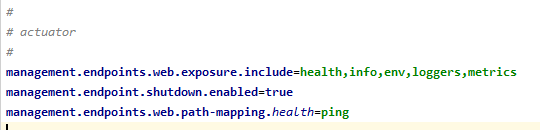
Декодированная посылка



Осталось еще несколько тонких моментов.

1. Админы бунтуют и требуют метрики приложения. Actuator! Вот наш ответ.

Добавляем в application.properties

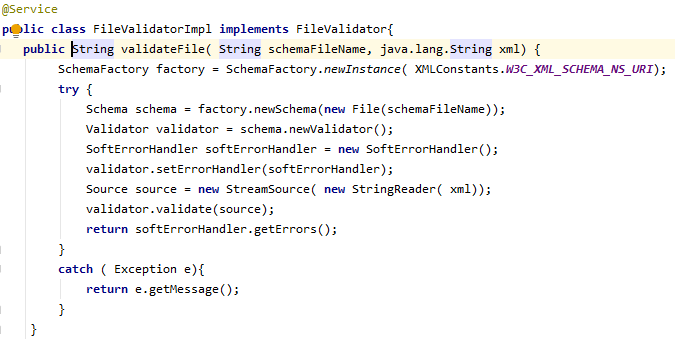


Остается только сказать админам url актуатора, они подключат его к системе мониторинга. Площадки в массовом порядке отказываются проверять доступность наших ресурсов.

1. Нам надо научить программу рапортовать о возникающих ошибках. И самый простой способ найден: привязаться к файлам протоколов. Как только возникает ошибка, сообщаем в АБС. Но как? Да просто в том же формате, что и сообщения web-сервиса. К нашим услугам org.apache.log4j.RollingFileAppender



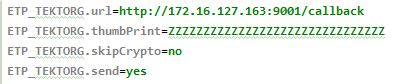
1. У нас все ж таки межпрограммное взаимодействие, и поэтому нам бы хотелось понимать язык собеседника. В этом нам поможет SAX-анализатор. Подключаем



1. По условиям задачи мы подключаемся через TLS. Принципиально это можно реализовать на Spring. Очень удачно, что для tomcat 8.5 CryptoPro изменило порядок интеграции. В SpringBoot на момент разработки был встроен как раз tomcat 8.5. Ни старый ни новый порядок подключения реализовать не удалось, несмотря на помощь техподдержки CryptoPro. Времени у нас нет и мы берем бесплатное решение sTunnel от CryptoPro. Ставим его впереди нашего web-сервиса и клиента.

Почитать про него можно тут <https://www.cryptopro.ru/products/other/stunnel>

Наш сервисный инженер, заходит на тестовый стенд, выгоняет нас, разворачивает miniApache, ставит сертификаты, настраивает трансляцию портов, попутно занимаясь какими-то своими делами, пара рестартов, наш конфигурационный файл принимает странный вил, но … работает.



Сервисный инженер связывается с площадками и начинает настраивать TLS на их стороне. Спустя какое-то время мы узнаем, что почти все банки и площадки поставили sTunnel. Редко кто реализовал TLS на Spring’е или на C#.

1. Сбербанк-АСТ угрожает нагрузочным тестированием. Предварительно нам бы и самим хотелось понять на каком мы свете. Напишем простенького клиента, который запустит 150 потоков, каждый поток сделает по 50 запросов. Хиленькая виртуалка с лвумя процессорами и 4Гигами памяти прекрасно вытягивает нагрузку.



PS. Наш банк в неофициальном зачете Минфина занимает третье место по успешности тестов, пропустив перед собой Сбербанк и Газпромбанк. В этом заслуга людей, программировавших АБС, вытянувших-таки ее на нужные параметры производительности и обеспечивших корректную работу бизнес-логики.

А я предлагал всем вместе написать программу обмена, все равно она почти у всех была бы одинаковой ☺