УДК 004.415.2

ВСТРАИВАНИЕ СЕРТИФИЦИРОВАННОГО ФСБ СРЕДСТВА КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ В СЕРВИС ЭЛЕКТРОННОЙ ОТЧЕТНОСТИ В РОСПРИРОДНАДЗОР

Пушкин Иван Александрович

студент
Санкт-Петербургский государственный университет телекоммуникаций им. М.А. Бонч-Бруевича, Санкт-Петербург

author@apriori-journal.ru

Аннотация. Рассмотрена реализация встраивания средства криптографической защиты информации сертифицированного ФСБ РФ в SaaS приложение электронной отчетности в Росприроднадзор написанного на PHP и Java при помощи RPC.

Ключевые слова: программная инженерия; СКЗИ; криптопровайдер; SaaS; ЭДО; электронная отчетность; RPC; удаленный вызов процедур.

INTEGRATION OF RUSSIAN FEDERAL CERTIFICATED CRYPTOGRAPHIC SERVICE PROVIDER IN SERVICE OF ELECTRONIC REPORTING IN THE FEDERAL SERVICE FOR SUPERVISION OF NATURAL RESOURCES

Pushkin Ivan Alexandrovich

student
The Bonch-Bruevich Saint-Petersburg State University of Telecommunications, Saint-Petersburg

Abstract. The article describes the integration of Russian Federal certificated cryptographic service provider in service of electronic reporting in The Federal Service for Supervision of Natural Resources written with PHP and Java using RPC.

Key words: programm engeniring; CSP; cryptoprovider; SaaS; EDI; electronic reporting; RPC; remote procedure call.

Введение

В 2012 году появилась возможность отчитываться перед Росприроднадзором (далее – РПН) в электронной форме. На веб-портале РПН принимаются отчеты 2-ТП (по отходам) и отчеты от субъектов малого и среднего предпринимательства в виде электронных документов в формате XML. Благодаря этому природопользователи могут сэкономить время, используя специальные программные средства для отчетности в РПН, а операторы электронного документооборота (ЭДО) могут заработать деньги, разрабатывая и предоставляя такие сервисы природопользователям. Кроме возможности отправлять электронные документы в РПН, эти сервисы могут автоматизировать процесс сбора и обработки информации для отчета, могут включать в себя базу данных со сведеньями об образовании и движении отходов, могут рассчитывать плату за природопользование и делать другие полезные вещи.

Электронные документы – это документы, в которых информация представлена в электронно цифровой форме и которые являются юридически полноценными аналогами бумажных документов.

Юридическая полноценность электронных документов достигается за счет выполнения оператором ЭДО массы технических и организационных условий. Главным условием является использование в системе ЭДО сертифицированного ФСБ средства криптографической защиты информации – криптопровайдера (список сертифицированных ФСБ средств криптографической защиты информации (СКЗИ) [4]).

Криптопровайдер – это реализация криптографических алгоритмов (набор функций криптографических преобразований). Если в языке или среде, предназначенной для использования криптопровайдера, имеется какой-либо стандартный интерфейс криптопровайдера, то криптопровайдер должен реализовывать этот интерфейс.

Криптопровайдер может быть сертифицированным и не сретифицированным. Не имеющий сертификата ФСБ криптопровайдер может ра-

ботать не хуже сертифицированного аналога, но достоверность любой подписи в системе использующей не сертифицированный криптопровайдер можно успешно оспорить в суде, а это создает широкие возможности для мошенников. Поэтому операторы ЭДО обязаны использовать сертифицированные криптопровайдеры.

В статье речь пойдет о встраивании сертифицированного ФСБ криптопровайдера в SaaS приложение электронной отчетности в РПН написанного на PHP и Java.

Постановка задачи

Главные критерии при выборе средств и методов разработки данной системы отчетности: простота поддержки, прозрачность [2] и безопасность. Проблема состоит в том, как прозрачно и безопасно работать с Java объектами и методами составляющими криптопровайдер из кода на PHP.

Использовать функции типа exec не безопасно [5] и не прозрачно. PHP extension не подойдет. Хорошее решение – это использование удаленного вызова процедур (Remote Procedure Call – RPC) [2] с разработанным специально под данный проект протоколом. RPC обеспечит прозрачность, безопасность и масштабируемость, не усложнив поддержку системы.

Чтобы использовать RPC для работы с криптопровайдером в системе электронной отчетности в РПН необходимо разработать протокол взаимодействия клиента (кода на PHP, вызывающего удаленно процедуры работающие с криптопровайдером) и сервера (обертки для криптопровайдера), написать клиентские и серверные заглушки.

За основу протокола взаимодействия может быть взят любой существующий протокол или формат данных. Для этой системы был выбран JSON.

Клиентские заглушки – это программный интерфейс к удаленно вызываемым процедурам. Клиентская заглушка принимает входные дан-

ные, сериализует и отсылает их, указывая, какую процедуру надо вызвать, дожидается ответа, принимает, десериализует и возвращает результат в вызывающий код.

Сервераня заглушка ожидает вызова, принимает входные данные, десериализует их, определяет и вызывает нужную процедуру, передает ей входные данные, получает результат работы процедуры, сериализует и отправляет результат на клиент.

Такая система является вертикально распределенной [2]. Криптопровайдер с серверыми заглушками представляет из себя сокетный сервер. Общая схема такого сервера и принцип его работы хорошо описана здесь [6].

Реализация

Для того, чтобы использовать для обращения к криптопровайдеру RPC, необходимо соглашение – протокол, определяющий формат сообщений, который соблюдался бы всеми серверными и клиентскими заглушками.

В данной системе в сообщение вызывающее процедуру должно входить две вещи: имя вызываемой процедуры и входные данные (параметры). Общий вид сообщения вызывающего процедуру может быть таким:

```
{
    'mathod': METHOD_NAME,
    'params': [
        PARAM_1,
        PARAM_2,
        PARAM_N
    ]
}
```

Здесь METHOD_NAME – это имя вызываемой процедуры, PARAM_1 … PARAM_N – это сериализованные параметры для вызываемой процедуры. Клиентские заглушки должны уметь формировать такие сообщения

и отправлять на сервер, а сервер, принимая такие сообщения, должен десериализовывать полученные данные, идентифицировать и вызывать необходимую процедуру.

В сообщение-ответ в данной системе должен входить сам ответ и сообщение об ошибке, которое может быть опущено в случае успешного завершения работы вызванной процедуры:

```
{
    'response': [
        RESULT_1,
        RESULT_2,
        RESULT_N
    ],
    'error': ERROR
}
```

Здесь RESULT_1 … RESULT_N – это сериализованные объекты возвращенные вызванной процедурой, а ERROR – это ошибка, которая может быть опущена, в случае, если никаких ошибок не произошло. Серверные заглушки должны уметь формировать такие сообщения и отправлять клиентским заглушкам, а клиентские заглушки должны уметь разбирать такие сообщения.

В данной системе, клиентские заглушки состоят из класса, который символизирует саму заглушку, выполняя при этом минимум операций, и класса-коннектора, который реализует механизм обмена сообщениями: сериализует, отправляет, получает, десериализует данные и возвращает их заглушке. Классы-заглушки – дочерние для класса коннектора, а класс-коннектор может быть абстрактным и выглядеть примерно так:

```
<?php

// Класс-коннектор
abstract class CryptoServiceConnector {

    // Параметры подключения
```

```
    static $addr = '127.0.0.1';
    static $port = '23456';

    // ...

    // Метод для вызова процедур
    protected static function _callMethod($method, $params) {
        // ...
    }

}
```

Заглушка должна знать имя своей процедуры и список параметров. Перед попыткой вызова процедуры заглушка может проверять переданные ей параметры и в случае каких-то несоответствий сразу выбрасывать исключение. Ниже приведена заглушка для функции хеширования:

```
<?php
class GostHash extends CryptoServiceConnector {

    // Константы алгоритмов хешей
    const HASH_ALGO_GOST3411_94_CryptoPro =
        'GOST3411-94-CryptoPro';
    const HASH_ALGO_GOST3411_94_Test = 'GOST3411-94-Test';
    const HASH_ALGO_GOST3411_94_Var1 = 'GOST3411-94-Var1';
    const HASH_ALGO_GOST3411_2012_256 = 'GOST3411-2012-256';
    const HASH_ALGO_GOST3411_2012_512 = 'GOST3411-2012-512';

    static function Hash($algo, $data="")
    {
        switch ($algo) {
        case self::HASH_ALGO_GOST3411_94_CryptoPro:
        case self::HASH_ALGO_GOST3411_94_Test:
        case self::HASH_ALGO_GOST3411_94_Var1:
        case self::HASH_ALGO_GOST3411_2012_256:
        case self::HASH_ALGO_GOST3411_2012_512:
```

```
            return self::_callMethod(
                "Hash",
                array($algo, $data)
            );
        }

        // Выбрасываем исключение: не правильный идентификатор
        // хеша
    }
}
```

В приведенном примере клиентской заглушки передается два параметра: идентификатор алгоритма хеширования и данные для хеширования. В этом случае возможно сразу проверить идентификатор алгоритма хеширования, и, если идентификатор не правильный, выбросить исключение. Также можно проверить, не передана ли в качестве данных для хеширования пустая строка, и, в случае, если передана, можно сразу вернуть значение хеша пустой строки.

Серверные заглушки могут быть реализованы на манер сервлетов:

```
public class Encrypt extends Proc {

    public JSONObject Run(JSONObject params) throws Exception {

        // Работаем с криптопровайдером

    }

}
```

Вызывается метод Run с переданными параметрами params. Дальше идет какая-то работа с криптопровайдером, например, шифрование. В случае какой-либо ошибки выбрасывается исключение, а если все прошло без ошибок возвращается JSONObject, который будет отправлен клиентской заглушке.

Для SaaS приложений, крайне важно, минимизировать время отклика системы. Учитывая тяжесть операции шифрования по ГОСТ 28147-89, следует предусмотреть кеш проверок подписей. PHP коду не нужно знать про этот кеш. Доступ к этому кешу должен иметь только код на Java, работающий непосредственно с криптопровайдером. Это обеспечит большую прозрачность системы.

Заключение

В статье рассмотрена реализация встраивания криптопровайдера в SaaS приложение электронной отчетности в РПН написанного на PHP и Java при помощи RPC.

Описанные в статье приемы обеспечат прозрачность системы, то есть уменьшат затраты времени и сил на поддержку кода.

В статье приведен протокол взаимодействия клиентских и серверных заглушек использованный для данной системы отчетности, предложена и описана архитектура серверной части приложения, затронуты некоторые юридические тонкости, касающиеся работы операторов ЭДО.

Список использованных источников

1. Федеральный закон от 06.04.2011 № 63-ФЗ «Об электронной подписи».
2. Таненбаум Э., Ван Стеен М. Распределенные системы. Принципы и парадигмы. СПб.: Питер, 2003. 877 с.
3. Клюев И.В. Создание системы защищенного электронного документооборота «Документ онлайн» // APRIORI. Серия: Естественные и технические науки. 2013. № 1 [Электронный ресурс]. Режим доступа: http://apriori-journal.ru/seria2/1-2013/Klyuev.pdf
4. Перечень средств защиты информации, сертифицированных ФСБ России [Электронный ресурс]. Режим доступа: http://clsz.fsb.ru/certification.htm (дата обращения 24.02.2015).
5. Коломыцев М.В., Носок С.А. Уязвимости приложений к некорректным входным данным // Радіоелектроніка, інформатика, управління . 2011. № 1 (24). С. 35-39.
6. Ильченко Е.А. Сокетный сервер для удаленного взаимодействия пользователя с системой управления распределенными вычислениями // Вестник Тамбовского университета. Серия: Естественные и технические науки . 2014. № 2. С. 551-557.