DOI 10.17223/20710410/5/11

УДК 004.056

РЕАЛИЗАЦИЯ МЕТОДА ЗАЩИТЫ УЧЁТНЫХ ДАННЫХ ДЛЯ ДОСТУПА К СУБД В ВЕБ-ПРИЛОЖЕНИЯХ[1]

П. А. Паутов

Томский государственный университет, г. Томск, Россия

E-mail: __Pavel__@mail.ru

В статье рассматриваются особенности аутентификации в веб-приложениях и возникающие при этом проблемы безопасности. Предлагается метод защиты аутентификационных данных для доступа к СУБД. Рассматриваются подходы к реализации метода на языке программирования PHP: с использованием только стандартных средств языка и специального модуля для работы с СУБД, позволяющего получать доступ к установленному соединению с СУБД без имени пользователя и пароля.

Ключевые слова: *многоуровневые приложения, веб-приложения, безопасность веб-приложений, аутентификация в многоуровневом приложении, защита учётных данных для доступа к СУБД, язык программирования PHP.*

Введение

В многоуровневых приложениях при аутентификации одного уровня приложения перед другим возникает проблема защиты хранимых аутентификационных данных каждого уровня, начиная со второго. В работе автора [1] для решения этой проблемы предложено использовать схему с использованием асимметричного шифрования. В данной работе рассматривается реализация этой схемы в трёхуровневом приложении на языке программирования PHP. Результатом реализации является программная библиотека, которую разработчики веб-приложений могут использовать для внедрения рассматриваемой схемы в свои проекты. Рассматриваются два варианта реализации: с использованием 1) только стандартных средств языка PHP и 2) специального модуля для работы с СУБД, позволяющего получать доступ к установленному соединению с СУБД без имени пользователя и пароля. В первом случае учётные данные для доступа к СУБД открываются на прикладном уровне при обработке каждого запроса пользователя, а во втором — только в начале сеанса работы пользователя.

Тезисное изложение результатов этой работы приведено в [2].

1. Многоуровневое приложение

Многоуровневое (или N-уровневое) приложение является приложением, разделённым на N самостоятельных уровней, каждый из которых может выполняться на отдельной платформе. Типичным примером такого подхода является трёхуровневая архитектура, которая подразумевает разделение приложения на следующие уровни:

1) уровень представления — отвечает за представление данных для пользователя;
2) уровень приложения (логики приложения) — отвечает за обработку данных в соответствии с логикой приложения;

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы. Результаты работы докладывались на Международной конференции с элементами научной школы для молодёжи, г. Омск, 7–12 сентября 2009 г.

3) уровень данных — отвечает за хранение данных.

Трёхуровневая архитектура используется в веб-приложениях, где уровни распределены следующим образом:

1) уровень представления — браузер;
2) уровень приложения — программа, исполняемая на веб-сервере;
3) уровень данных — СУБД.

Распределение приложения на несколько изолированных уровней позволяет достичь большей гибкости при масштабировании приложения или смене используемых технологий на каком-либо уровне [3].

2. Аутентификация в трёхуровневом приложении

Применение трёхуровневой архитектуры приводит к появлению нескольких звеньев аутентификации. В классическом двухуровневом приложении клиент аутентифицируется перед сервером. В трёхуровневом приложении имеются два звена аутентификации: клиент аутентифицируется перед прикладным уровнем, а прикладной уровень аутентифицируется перед СУБД. На рис. 1 представлен данный механизм с применением парольной аутентификации.

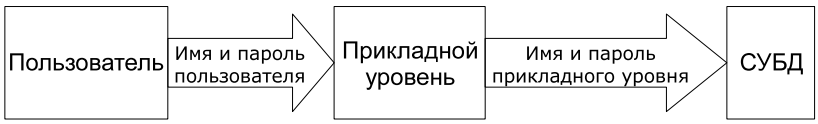

Рис. 1. Два звена аутентификации в веб-приложении

Можно выделить два частных случая (подхода) в организации трёхуровнего приложения:

1) прикладной уровень имеет фиксированный набор пользователей СУБД, от имени которых он может работать;
2) каждому пользователю приложения соответствует пользователь СУБД.

Далее будем рассматривать только первый случай с применением парольной аутентификации. Такой способ организации системы широко распространён в современных веб-приложениях. Чаще всего для взаимодействия с базой данных используется всего один пользователь. Например, так устроены приложения «1С-Битрикс» и «PHP-Nuke» [4, 5]. Самые доступные тарифы хостинг-провайдеров ограничивают количество пользователей для доступа к СУБД [6, 7]. А так как количество пользователей приложения, как правило, превышает заданный лимит пользователей СУБД, то остаётся возможность использовать только первый подход. В этом случае каждому пользователю приложения ставится в соответствие пользователь СУБД с минимально необходимыми привилегиями.

Таким образом, возникает необходимость хранения аутентификационных данных (имён и паролей) пользователей СУБД на прикладном уровне. Во многих современных веб-приложениях эти данные хранятся в открытом виде в некотором файле конфигурации на веб-сервере. При компрометации прикладного уровня злоумышленник может получить доступ к этим данным. В работе [1] рассмотрено несколько схем защиты паролей пользователей СУБД, хранимых на прикладном уровне. Предпочтительной для использования является схема с использованием асимметричного шифра.

3. Схема аутентификации с использованием асимметричного шифра

Каждому пользователю системы ставится в соответствие пара — открытый и закрытый ключи. На прикладном уровне предлагается хранить таблицу следующего вида:

1) имя пользователя приложения;
2) учётные данные для его доступа к СУБД — имя и пароль соответствующего пользователя СУБД, зашифрованные с помощью открытого ключа пользователя приложения;
3) открытый ключ пользователя приложения;
4) закрытый ключ пользователя приложения, зашифрованный с помощью пароля.

Запись, соответствующая администратору приложения, должна содержать имена и пароли всех пользователей СУБД.

Алгоритм аутентификации пользователя перед приложением будет выглядеть следующим образом:

1. Пользователь приложения передаёт свои имя и пароль прикладному уровню.
2. Прикладной уровень находит в описанной выше таблице имя пользователя и с помощью пароля расшифровывает закрытый ключ.
3. Прикладной уровень с помощью закрытого ключа пользователя расшифровывает учётные данные, необходимые для доступа к СУБД.
4. Прикладной уровень устанавливает соединение с СУБД от имени учётной записи, полученной на предыдущем шаге.
5. В случае успешного соединения считается, что пользователь прошёл аутентификацию.

Данная схема позволяет хранить данные для доступа к СУБД в закрытом виде, но, как видно из алгоритма аутентификации, имя и пароль пользователя СУБД на некоторое время появляются в открытом виде в памяти прикладного уровня. Поэтому если злоумышленник имеет возможность перехватывать данные из памяти прикладного уровня, то данная схема только на некоторое время от момента взлома злоумышленником прикладного уровня может отсрочить получение им доступа к СУБД, а именно до того, как очередной пользователь выполнит вход в систему. Если злоумышленник получит доступ, например, только к файлам конфигурации приложения, то он не сможет получить доступ к СУБД. Использование данного метода позволяет сократить время, в течение которого учётные данные для доступа к СУБД будут находиться в открытом виде.

При использовании рассматриваемой схемы усложняются операции по управлению пользователями. Ниже приведены алгоритмы выполнения данных операций.

Смена пароля пользователем.

1. Пользователь присылает свой старый и новый пароли.
2. Прикладной уровень расшифровывает закрытый ключ пользователя с помощью пароля.
3. Прикладной уровень расшифровывает учётные данные СУБД с помощью закрытого ключа.
4. Прикладной уровень устанавливает соединение с СУБД для проверки правильности старого пароля.
5. В случае успеха предыдущего шага прикладной уровень зашифровывает закрытый ключ пользователя на новом пароле.

Создание нового пользователя. Создать нового пользователя приложения может только администратор.

1. Администратор присылает свой пароль и пароль нового пользователя.
2. Прикладной уровень с помощью пароля администратора получает учётные данные СУБД.
3. Прикладной уровень генерирует открытый и закрытый ключи нового пользователя.
4. Прикладной уровень шифрует соответствующие пользователю учётные данные СУБД на закрытом ключе пользователя.
5. Прикладной уровень шифрует закрытый ключ нового пользователя с помощью пароля.

Смена учётной записи СУБД. При смене учётной записи СУБД необходимо с помощью открытых ключей соответствующих пользователей зашифровать новые учётные данные.

Изменение пользователя СУБД для данного пользователя приложения.

1. С помощью пароля администратора расшифровывается закрытый ключ администратора, а с помощью последнего расшифровываются учётные данные СУБД.
2. Учётные данные СУБД шифруются с помощью открытого ключа пользователя.

4. Реализация схемы аутентификации с использованием асимметричного шифра

4.1. Результат реализации

Для реализации данной схемы был выбран язык программирования PHP, так как он являются одним из самых распространённых средств для разработки веб-приложений. Результатом является программная библиотека, которую разработчики веб-приложений могут использовать для внедрения рассматриваемой схемы в свои проекты. Задачи, решаемые программной библиотекой, следующие:

1) реализация рассмотренных алгоритмов управления пользователями;
2) управление сеансами работы пользователей и подключениями к СУБД.

Первая задача была решена с использованием библиотеки OpenSSL и базы данных SQLite. OpenSSL предоставляет необходимые криптографические функции, а модуль SQLite позволяет работать с локальным файлом как с SQL-совместимой базой данных. Так было организовано хранилище учётных данных, описанное в п. 3.

Рассмотрим вторую задачу подробнее. Для этого опишем работу типичного веб-приложения, разработанного с использованием языка программирования PHP.

4.2. Схема веб-приложения

На рис. 2 изображены основные компоненты такого приложения.

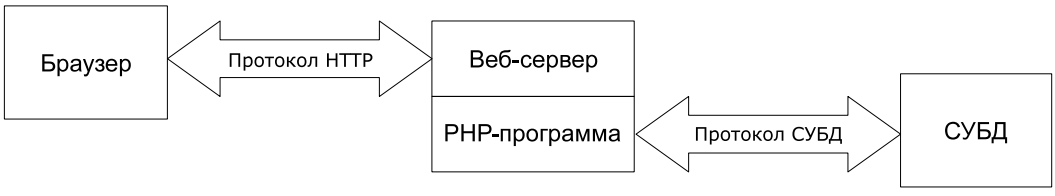

Рис. 2. Компоненты веб-приложения

Браузер посылает веб-серверу HTTP-запрос. Веб-сервер запускает соответствующую PHP-программу. Программа *устанавливает соединение с СУБД*, получает необходимые данные, *закрывает соединение*, оформляет полученные данные в виде HTML-страницы и передаёт последнюю веб-серверу. Веб-сервер отправляет страницу браузеру как ответ на HTTP-запрос.

Протокол HTTP не имеет состояния, т. е. веб-сервер не хранит информации о клиентах между запросами. Однако для того чтобы веб-приложение могло контролировать доступ пользователя и выдавать различное содержимое страниц в зависимости от пользователя, необходимо отслеживать сеанс работы пользователя. Для решения данной задачи применяется механизм сессий. Во время первого обращения пользователя к приложению генерируется некоторый случайный идентификатор (идентификатор сессии), который отсылается браузеру в теле HTTP-ответа. Далее, в теле каждого HTTP-запроса браузер посылает этот идентификатор веб-серверу. PHP-программа ведёт базу данных сессий, в которой для каждого идентификатора хранятся различные данные о пользователе: имя, уровень доступа и т. п.

4.3. Использование только стандартных средств PHP

Для работы описанной схемы веб-приложения прикладному уровню необходимы имя и пароль для доступа к СУБД для обработки каждого запроса пользователя. Так как в рассматриваемой схеме учётные данные СУБД зашифрованы открытым ключом пользователя, а закрытый ключ пользователя зашифрован с помощью пароля пользователя, то для обработки каждого запроса в сеансе прикладному уровню понадобится пароль пользователя. Пароль же предъявляется пользователем только в начале сеанса работы и не должен храниться в течение сеанса. Если же на каждый запрос требовать от пользователя предъявления пароля, то использование операций асимметричной криптографии при обработке каждого запроса может (при большом количестве пользователей) негативно сказаться на производительности системы. Для решения этих проблем после аутентификации пользователя перед приложением выполняются следующие действия:

1. На прикладном уровне генерируется случайный ключ K симметричного шифрования. Этот ключ записывается в сессию текущего пользователя.
2. С помощью ключа K шифруются учётные данные СУБД.
3. Зашифрованные учётные данные СУБД отправляются браузеру в теле HTTP-ответа.

Далее с каждым запросом браузер посылает веб-серверу идентификатор сессии и зашифрованные на ключе K учётные данные для доступа к СУБД. Таким образом, пароль пользователя требуется только в начале сеанса работы и используется для аутентификации пользователя с применением асимметричного шифрования, как описано выше, а во время обработки запросов для получения учётных данных СУБД используется симметричный шифр.

4.4. Использование постоянных соединений с СУБД

Классическая схема работы PHP-программы, когда соединение с СУБД устанавливается и разрывается при обработке каждого запроса пользователя, не всегда является оптимальной. Если установка соединения с СУБД сопровождается большими накладными расходами, то выгоднее поддерживать постоянное соединение и использовать это соединение для обработки запросов. Поэтому в PHP существует механизм работы с постоянными соединениями с СУБД [8]. В стандартном программном интерфейсе PHP для работы с СУБД имя пользователя и пароль необходимы как для того, чтобы

установить новое соединение, так и для того, чтобы получить доступ к уже установленному соединению. Фактически же для работы с уже установленным соединением имя и пароль для СУБД не нужны.

На основе стандартного модуля PHP для СУБД MySQL был реализован модуль, позволяющий получать доступ к уже установленному соединению без имени пользователя и пароля. В этом модуле функция установки соединения с СУБД возвращает идентификатор, который можно сохранить в сессии и использовать для получения соединения с СУБД при обработке последующих запросов.

Заключение

При использовании парольной аутентификации прикладному уровню необходимы имя и пароль пользователя СУБД в открытом виде для установления соединения с СУБД. В большинстве веб-приложений эти данные хранятся на прикладном уровне в открытом виде постоянно. Реализованная схема аутентификации позволяет хранить учётные данные для доступа к СУБД в закрытом виде и получать их в открытом только при необходимости.

Реализация схемы с помощью только стандартных средств языка PHP не требует специального модуля для работы с СУБД и легче переносима на разные платформы. Однако при использовании такой реализации учётные данные для доступа к СУБД открываются на прикладном уровне при обработке каждого запроса пользователя. Подход же с применением специального модуля для работы с СУБД предоставляет более высокий уровень защиты, так как учётные данные для доступа к СУБД открываются только в начале сеанса работы пользователя.

ЛИТЕРАТУРА

1. *Паутов П. А.* Проблема аутентификации в многоуровневых приложениях // Прикладная дискретная математика. 2008. № 2. С. 87–90.
2. *Паутов П. А.* Реализация метода защиты аутентификационных данных в многоуровневых приложениях // Прикладная дискретная математика. Приложение. 2009. № 1. С. 50–51.
3. `www.wikipedia.org` — Википедия. 2009.
4. Руководство по инсталляции «1С-Битрикс: Управление сайтом 6.xx». 2007. `www.1c-bitrix.ru`
5. *Karakas C., Erba C.* PHP-Nuke: Management and Programming. 2005. `www.karakas-online.de/EN-Book`
6. `www.masterhost.ru` — 2009.
7. `www.peterhost.ru` — 2009.
8. *Olson P.* PHP Manual. 2009. `www.php.net/manual`