2008 Математические основы компьютерной безопасности № 2(2)

DOI 10.17223/20710410/2/19

УДК 004.056

ПРОБЛЕМА АУТЕНТИФИКАЦИИ В МНОГОУРОВНЕВЫХ ПРИЛОЖЕНИЯХ

П.А. Паутов

Томский государственный университет

E-mail: __Pavel__@mail.ru

В статье рассматриваются особенности аутентификации в многоуровневом приложении и возникающие при этом проблемы безопасности. Предлагается метод усиления защищенности наиболее распространенной в современных веб-приложениях двухзвенной схемы аутентификации.

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

Понятие многоуровневого приложения

Многоуровневое приложение (или *N*-уровневое приложение) – приложение, разделенное на *N* самостоятельных уровней, каждый из которых может выполняться на отдельной платформе. Типичным примером такого подхода является трехуровневая 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.
3. Уровень данных – отвечает за хранение данных.

Трехуровневая архитектура используется в веб-приложениях. В веб-приложениях 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 [1].

Аутентификация в многоуровневом приложении

Применение трехуровневой архитектуры приводит к появлению нескольких звеньев аутентификации. В классическом двухуровневом приложении клиент аутентифицируется перед сервером. В трехуровневом приложении появляется второе звено аутентификации – клиент аутентифицируется перед прикладным уровнем, а прикладной уровень аутентифицируется перед СУБД. На рис. 1 представлен данный механизм с применением парольной аутентификации.

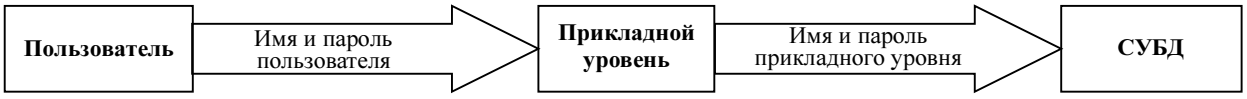

Рис. 1. Два звена аутентификации в веб-приложении

Далее можно выделить два частных случая:

1. Прикладной уровень имеет фиксированный набор пользователей СУБД, от имени которых он может работать.
2. Каждому пользователю приложения соответствует пользователь СУБД.

В современных веб-приложениях больше распространен первый подход. Вероятно, это обусловлено тем, что самые доступные тарифы хостинг-провайдеров ограничивают количество пользователей для доступа к СУБД (например, masterhost.ru, peterhost.ru). Чаще всего для взаимодействия с базой данных используется всего один пользователь. Например, приложения «1С-Битрикс» и «PHP-Nuke» используют данный метод [2, 3]. Таким образом, независимо от уровня привилегий пользователя приложения прикладной уровень всегда работает с СУБД от имени пользователя с максимальными привилегиями. Но современные веб-приложения, как правило, содержат несколько групп пользователей с различными привилегиями, например: «гость», «пользователь», «оператор», «администратор». Соответственно «администратор» обладает большими при-

вилегиями, чем «оператор», «оператор» имеет больше прав, чем «обычный пользователь», а «гость» обладает минимальным набором прав. При использовании для взаимодействия с СУБД только одного пользователя контроль прав доступа пользователей полностью осуществляется на прикладном уровне. Следовательно, если «обычный пользователь» как-то обойдет контроль прав на уровне приложения, то он получит доступ к СУБД с максимальными правами. Например, при успешной SQL-инъекции вставленный злоумышленником SQL-код будет выполнен с максимальными привилегиями. Реализация разделения прав также и на уровне СУБД позволяет уменьшить ущерб от таких атак. Для этого необходимо создать несколько пользователей СУБД, соответствующих группам пользователей веб-приложения, и назначить этим пользователям соответствующие привилегии [4]. То есть для доступа к базе данных будет использоваться один из нескольких пользователей, в зависимости от того, к какой группе пользователей относится пользователь приложения. При таком подходе, если, например, пользователь типа «оператор» как-то обойдет контроль доступа прикладного уровня, он сможет выполнить только те действия, которые позволены соответствующему пользователю СУБД. И, следовательно, ущерб от такой атаки будет меньше, так как привилегии пользователя СУБД «оператор» ограничены.

Защита аутентификационных данных для доступа к СУБД

В случае, когда для работы с СУБД прикладной уровень использует фиксированное количество учетных записей, возникает необходимость хранения аутентификационных данных этих записей (например, имен пользователей и паролей). Во многих современных веб-приложениях эти данные хранятся в открытом виде в некотором файле конфигурации на веб-сервере. Таким образом, при компрометации прикладного уровня злоумышленник может получить доступ к этим данным. Существующие подходы к решению данной проблемы в основном пытаются ограничить доступ к этим данным на уровне одного локального пользователя ОС сервера, на котором выполняется прикладной уровень [5 – 7]. Реквизиты для доступа к СУБД записываются в некоторый файл или хранилище, доступные только одному пользователю ОС, а сам веб-сервер запускается от имени этого пользователя. Такие методы не делают различия между пользователями веб-приложения, то есть независимо от того, какого типа пользователь («администратор» или «оператор») обращается к веб-приложению, прикладной уровень всегда будет иметь доступ к аутентификационным данным всех пользователей СУБД.

В данной работе предлагается защитить аутентификационные данные для доступа к СУБД, зашифровав их с помощью паролей пользователей веб-приложения. Такой подход позволяет не хранить реквизиты для доступа к СУБД в открытом виде, но при этом несколько усложняются операции по управлению пользователями. Далее рассматриваются три схемы реализации данного подхода. Для каждой схемы предлагаются алгоритмы выполнения следующих задач:

1. Смена пароля пользователем.
2. Создание нового пользователя.
3. Смена учетной записи СУБД.
4. Перевод пользователя из одной группы в другую.

1. Прямая схема. Для работы данной схемы прикладной уровень веб-приложения должен хранить таблицу, состоящую из двух колонок:

1) имя пользователя веб-приложения;

2) имя и пароль пользователя СУБД, соответствующего группе пользователя приложения, зашифрованные с помощью пароля пользователя веб-приложения.

Запись таблицы, соответствующая администратору приложения, должна содержать учетные данные всех пользователей СУБД. Данную таблицу можно хранить в БД. Для доступа к ней можно использовать пользователя СУБД, единственной привилегией которого был бы доступ к данной таблице на чтение. Такой пользователь СУБД может соответствовать группе пользователей приложения «Гости». Алгоритм аутентификации пользователя перед веб-приложением будет выглядеть следующим образом:

1. Пользователь веб-приложения передает свои имя и пароль прикладному уровню.
2. Прикладной уровень находит в описанной выше таблице имя пользователя и с помощью пароля пользователя расшифровывает учетные данные, необходимые для доступа к СУБД.
3. Прикладной уровень устанавливает соединение с СУБД от имени учетной записи, полученной на 2-м шаге.
4. В случае успешного соединения считается, что пользователь прошел аутентификацию.

Далее прикладной уровень работает с СУБД по установленному соединению от имени пользователя с привилегиями, соответствующими группе пользователя веб-приложения. Рассмотрим задачи по управлению пользователями.

1.1. Смена пароля пользователем.

1. Пользователь присылает свой старый и новый пароли.
2. Прикладной уровень расшифровывает учетные данные СУБД.

3. Прикладной уровень устанавливает соединение с СУБД для проверки правильности старого пароля.

4. В случае успеха шага 3 прикладной уровень зашифровывает учетные данные СУБД на новом пароле пользователя.

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

1. Администратор присылает свой пароль и пароль нового пользователя.

2. С помощью пароля администратора расшифровываются учетные данные СУБД.

3. С помощью пароля пользователя зашифровываются учетные данные СУБД, соответствующие группе пользователя.

1.3. Смена учетной записи СУБД. При смене учетной записи СУБД необходимо зашифровать новые учетные данные на паролях соответствующих пользователей. В рамках данной схемы это требует получения паролей этих пользователей, что накладывает сильное ограничение на применение данной схемы.

1.4. Перевод пользователя из одной группы в другую. При переводе пользователя в другую группу необходимо также знать пароль пользователя, так как необходимо зашифровать реквизиты СУБД на пароле пользователя. Это ограничивает применимость данной схемы на практике.

2. Схема с мастер-ключом. В данной схеме вводится некоторый мастер-ключ, с помощью которого шифруются учетные данные пользователей СУБД. Таким образом, таблица из предыдущей схемы приобретает следующий вид:

1) имя пользователя веб-приложения;

2) мастер-ключ, зашифрованный с помощью пароля пользователя веб-приложения.

Необходимо также хранить таблицу с учетными данными СУБД, зашифрованными мастер-ключом. Рассмотрим задачи по управлению пользователями.

2.1. Смена пароля пользователем. Аналогично схеме 1.

2.2. Создание нового пользователя. Аналогично схеме 1.

2.3. Смена учетной записи СУБД.

1. С помощью пароля администратора расшифровывается мастер-ключ.

2. Новые учетные данные СУБД шифруются с помощью мастер-ключа.

2.4. Перевод пользователя из одной группы в другую. Перевод пользователя в другую группу не требует изменения данных в рассматриваемых таблицах.

Данная схема позволяет решить проблемы предыдущего варианта, но также имеет несколько серьезных недостатков:

1. При смене мастер-ключа мы сталкиваемся с теми же проблемами, что и в схеме 1 при смене учетных данных СУБД. Таким образом, мастер-ключ должен быть постоянным – такой подход может быть неприемлем для приложений, где требуется высокий уровень безопасности.

2. Если легальный пользователь системы сможет получить доступ к таблицам с учетными данными, то он сможет получить мастер-ключ и расшифровать все реквизиты СУБД. В итоге данный пользователь получит доступ к СУБД с максимальными привилегиями, тогда как в схеме 1 такой пользователь смог бы узнать только реквизиты СУБД, соответствующие его уровню доступа.

3. Схема с использованием асимметричного шифра. В данной схеме каждому пользователю системы ставится в соответствие пара – открытый и закрытый ключи. Таблица с учетными данными приобретает следующий вид:

1) имя пользователя веб-приложения;

2) имя и пароль пользователя СУБД, соответствующего группе пользователя приложения, зашифрованные с помощью открытого ключа пользователя;

3) открытый ключ пользователя;

4) закрытый ключ пользователя, зашифрованный с помощью пароля.

Такая схема позволяет решить проблемы двух предыдущих.

3.1. Смена пароля пользователем. Аналогично схеме 1.

3.2. Создание нового пользователя. Аналогично схеме 1, только необходимо сгенерировать открытый и закрытый ключи.

3.3. Смена учетной записи СУБД. При смене учетной записи СУБД необходимо с помощью открытых ключей соответствующих пользователей зашифровать новые учетные данные.

3.4. Перевод пользователя из одной группы в другую.

1. С помощью пароля администратора расшифровывается закрытый ключ администратора, а с помощью последнего расшифровываются учетные данные СУБД.

2. Учетные данные СУБД шифруются с помощью открытого ключа пользователя.

Данная схема позволяет успешно выполнять операции по управлению пользователями, а также обеспечивает более высокий уровень защиты, чем предыдущие схемы.

Заключение

Предложенные схемы позволяют защитить аутентификационные данные для доступа к СУБД и, таким образом, повышают уровень защищенности всего веб-приложения. Данный подход планируется реализовать в веб-приложении «Система тестирования знаний» [8] для экспериментальной проверки предложенного метода.

ЛИТЕРАТУРА

1. *Материалы* Свободной Энциклопедии. http://wikipedia.org.
2. *Karakas C.*, *Erba C.* PHP-Nuke: Management and Programming. 04.08.2005.
3. *Руководство* по инсталляции «1С-Битрикс: Управление сайтом 6.xx». 28.11.2007.
4. *OWASP*. A Guide to Building Secure Web Applications and Web Services 2.0 Black Hat Edition. July 27, 2005.
5. *Meier J.D.* Improving Web Application Security: Threats and Countermeasures. Microsoft Corporation, 2003.
6. *Newbigin J.* Keeping PHP Database Passwords Secure. 14.09.2005. http://uranus.it.swin.edu.au/~jn/linux/php/passwords.htm
7. *Secure* Database Access using PHP4, Apache, suexec, CGI, PostgreSQL, and peer sameuser. http://tril.tunes.org/admin/secure-db.html
8. *Паутов П.А.* Разработка и реализация веб-приложения «Система тестирования знаний» // Вестник ТГУ. Приложение. 2007. № 23. С. 194 – 197.