симплексного поиска. Структура синтезированной трехсвязной марковской цепи, соответствующая процессу поиска на этапе восхождения, представлена стохастическим графом и описывает поиск с запретом возврата. Данный подход к синтезу структуры многосвязной цепи Маркова позволяет получить оптимальные алгоритмы на этапах восхождения и доводки при различных критериях эффективности поиска.

ЛИТЕРАТУРА

1. *Кузнецов Н. А., Кульба В. В., Микрин Е. А.* Информационная безопасность систем организационного управления. Теоретические основы. М.: Наука, 2006. 430 с.

УДК 004.056

РЕАЛИЗАЦИЯ МЕТОДА ЗАЩИТЫ АУТЕНТИФИКАЦИОННЫХ ДАННЫХ В МНОГОУРОВНЕВЫХ ПРИЛОЖЕНИЯХ

П. А. Паутов

Многоуровневое (или N-уровневое) приложение — приложение, разделенное на N самостоятельных уровней, каждый из которых может выполняться на отдельной платформе. Типичным примером является трехуровневая архитектура, используемая в веб-приложениях, где уровни распределены следующим образом:

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

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. В современных веб-приложениях прикладной уровень работает с СУБД от имени фиксированного числа пользователей (часто от имени одного пользователя). При использовании парольной аутентификации возникает необходимость хранения аутентификационных данных (имен и паролей) пользователей СУБД на прикладном уровне. Данная проблема была рассмотрена в работе [1], в которой было предложено несколько схем защиты аутентификационных данных СУБД, хранимых на прикладном уровне. Предпочтительной для использования является схема с использованием асимметричного шифра.

Для реализации данной схемы был выбран язык программирования PHP. Программная библиотека, реализующая схему с использованием асимметричного шифра, решает следующие задачи:

1) реализацию рассмотренных в [1] 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 описанное в работе [1] хранилище учетных данных.

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

Браузер посылает веб-серверу HTTP-запрос. Веб-сервер запускает соответствующую PHP-программу. Программа *устанавливает соединение с СУБД*, получает необ-

ходимые данные, *закрывает соединение*, оформляет полученные данные в виде HTML-страницы и передает последнюю веб-серверу. Веб-сервер отправляет страницу браузеру, как ответ на HTTP-запрос.

Для работы такой системы прикладному уровню необходимы имя и пароль для доступа к СУБД с целью обработки каждого запроса от браузера. Так как в рассматриваемой схеме защиты для получения учетных данных СУБД необходим пароль пользователя, то для обработки каждого запроса прикладному уровню понадобится пароль пользователя. Но пароль предъявляется пользователем только в начале сеанса работы. Для решения этой проблемы после аутентификации пользователя перед приложением предлагается выполнить следующие действия:

1. На прикладном уровне генерируется случайный ключ K симметричного шифрования. На время сеанса работы пользователя этот ключ хранится на сервере.
2. С помощью ключа K шифруются учетные данные СУБД.
3. Зашифрованные учетные данные СУБД отправляются браузеру в теле HTTP-ответа.

Далее с каждым запросом браузер посылает веб-серверу зашифрованные учетные данные для доступа к СУБД. Таким образом, пароль пользователя требуется только в начале сеанса работы.

Другой подход к решению рассматриваемой задачи заключается в использовании механизма постоянных соединений с СУБД. Такие соединения остаются открытыми после окончания обработки запроса PHP программой. Но в стандартном программном интерфейсе PHP для работы с постоянными соединениями имя пользователя и пароль необходимы как для установления нового соединения, так и для получения доступа к уже установленному [2]. Но фактически 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 использовать для получения соединения с СУБД при обработке последующих запросов.

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 в открытом виде постоянно. Рассматриваемая схема позволяет хранить учетные 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. Но при использовании такой реализации учетные данные для доступа к СУБД «открываются» на прикладном уровне при обработке каждого запроса пользователя. Подход с применением специального модуля для работы с СУБД предоставляет более высокий уровень защиты, так как учетные данные для доступа к СУБД «открываются» только в начале сеанса работы пользователя. На данный момент такой модуль реализован только для СУБД MySQL.

ЛИТЕРАТУРА

1. *Паутов П. А.* Проблема аутентификации в многоуровневых приложениях // Прикладная дискретная математика. 2008. № 2. С. 87–90.
2. *Olson P.* PHP Manual. 2009.