Математические структуры и моделирование
2016. № 4(40). С. 143–150

УДК 004.056

ДОКАЗАТЕЛЬСТВО С НУЛЕВЫМ РАЗГЛАШЕНИЕМ КАК МЕТОД АУТЕНТИФИКАЦИИ В ВЕБ-ПРИЛОЖЕНИЯХ

И.Д. Сиганов
аспирант, e-mail: ilya.siganov@gmail.com

Омский государственный университет им. Ф.М. Достоевского

Аннотация. В статье рассмотрен альтернативный протокол аутентификации для веб-приложений, основанный на доказательстве с нулевым разглашением. Такой подход позволяет избежать многих уязвимостей, которым подвержены системы с обычной парольной аутентификацией, таких как перехват процесса входа в систему, подсматривание секрета, взлома и анализа базы данных сервера для поиска паролей. Реализованная система состоит из веб-сервера и андроид-приложения. Процесс аутентификации происходит с помощью сканирования QR-кода со страницы веб-сайта приложением–менеджером аутентификации. Достоинством системы можно считать её удобство, высокий уровень защищённости, возможность безбоязненного использования на публичных и подозрительных компьютерах.

Ключевые слова: компьютерная безопасность, аутентификация, доказательство с нулевым разглашением, веб-приложения, андроид.

Введение

Как известно, выделяют три сущности аутентификации — что субъект знает, что имеет и кто субъект есть. Дешевле всего для реализации взять первую сущность, основанную на знании какого-то секрета, например, пароля. Такие системы проще всего проектировать, на первый взгляд, и они не требуют от пользователя дополнительных устройств. Скажем так, это наиболее привычный метод аутентификации где бы то ни было — вводить пароль в систему. При всей своей простоте и преимуществах у этого подхода есть серьёзный недостаток. Безопасность полностью зависит от самого пользователя, который придумал секрет, предполагая, что сервера хранят пароли достаточно безопасно. Но люди — очень плохой источник энтропии, кроме того, им нужно помнить пароль, поэтому они придумывают очень простые секреты. К тому же этот подход плохо масштабируется на пользователях. Им приходится помнить столько паролей, сколько сервисов они использует. Всевозможные менеджеры и генераторы паролей призваны решить эти проблемы. Но не стоит забывать, что они реализуются как специальные приложения или расширения для браузеров, которые надо ещё установить на компьютер. Это не приемлемо для исполь-

зования на общедоступных компьютерах, например, в школах, университетах, интернет-кафе, на рабочих местах. Итак, проблемы:

1. Люди — плохой источник энтропии, поэтому создают очень простые секреты, которые легко взломать.
2. Люди не могут помнить сотни паролей, от этого страдает сложность паролей ещё сильнее. Кроме того, это просто не удобно.
3. Возможен перехват пароля во время его ввода.
4. Если пароль перехвачен, то его нужно сразу же менять. Но не всегда можно вовремя обнаружить перехват.
5. База данных с хэшированными паролями всегда поддаётся криптоанализу. Постоянно в СМИ появляются материалы об очередной утечке баз паролей.

Исходя из проблем, можно сформулировать критерии идеального протокола аутентификации:

1. Стойкость не зависит от выбора пользователя, т.е. система сама создаст необходимые ключи; эталон, хранимый на сервере, не должен быть уязвим к прямому взлому для вычисления самого ключа аутентификации.
2. Стойкость к пассивному прослушиванию, подглядыванию.
3. Стойкость к атаке воспроизведением.
4. Достаточно высокая скорость работы, чтобы не сильно нагружать вычислительные мощности клиента, что позволит охватить распространённые устройства.
5. Не создавать высокие нагрузки на сервер.

В общем случае аутентификация — это доказательство: Виктор должен удостовериться, что Пегги действительно знает какой-то секрет. Ограничения, которые были выписаны, можно представить в виде желания Пегги не раскрывать свой секрет, но каким-то образом доказать Виктору, что она знает секрет, причём Виктор действительно поверит Пегги. К тому же Пегги хочет, чтобы Виктор не смог потом воспользоваться доказательством и выдать себя за неё. Протоколы, реализующие данную схему, называются протоколами нулевого разглашения. Классическим объяснением этого протокола служит пример с пещерой, в которой есть дверь с паролем. Формальное определение даётся через вероятностные машины Тьюринга, которые разделяют общую ленту [3, с. 81]. Если описать простыми словами, то все сводится к тому, что у Пегги есть неограниченные вычислительные мощности, а у Виктора имеется лишь обычная машина. Виктор отправляет вопросы Пегги, на которые в силах ответить только она, но проверить правильность ответа может каждый. Предполагается также, что вероятность обмана крайне мала. К тому же есть дополнительное ограничение, называемое как раз нулевым разглашением — т.е. Виктор не сможет воспользоваться доказательствами Пегги, чтобы выдать себя за неё.

Как видно, протоколы нулевого разглашения устойчивы к пассивному прослушиванию, повторному воспроизведению и криптоанализу базы данных эталонов. К сожалению, большинство таких протоколов интерактивные и требуют

прохождения большого числа испытаний для построения доказательства. Связь между числом испытаний n и вероятностью аутентичности выражается формулой:

$$p(n) = 1 - \frac{1}{2^n}.$$

Для хоть сколько-то приемлемой величины нужно пройти хотя бы 10 испытаний. В случае растущего потока людей будет генерироваться большая нагрузки на сервер аутентификации. Чтобы этого избежать, придумывают протоколы нулевого разглашения, где необходимо не так много итераций. Одним из таких примеров может служить схема Шнорра. Это модификация схемы Эль-Гамаля и Фиата-Шамира, в основе использующая проблему дискретного логарифмирования в конечном поле. Есть, правда, некоторые вопросы в том, действительно ли этот протокол нулевого разглашения, так как нет доказательства за и против.

1. Описание системы

Концептуально разработанная система описывается взаимодействием между сервером аутентификации и клиентским приложением — менеджером аутентификации. Сервер аутентификации может работать как отдельный сервис или как микросервис, встроенный, например, в другое веб-приложение. В данной работе реализован второй подход. В любом случае сервер аутентификации имеет БД пользовательских данных, необходимых для процесса аутентификации, и дополнительные хранилища, необходимые для функционирования самого протокола. Клиентское приложение написано под ОС андроид; оно хранит список всех учётных записей пользователя в своей внутренней шифрованной БД вместе с приватными ключами и осуществляет процесс аутентификации. Клиент может работать с любыми веб-приложениями, использующими разработанный сервер аутентификации, то есть нет привязки к определённому адресу сервера и реализации веб-приложения, главное чтобы API взаимодействия с сервером аутентификации оставался прежним. Благодаря этому приложение может использоваться для любых сайтов, использующих этот сервер аутентификации. Особенностью данной реализации является то, что система разрабатывалась без жёсткой привязки к определённому протоколу аутентификации, поэтому поддерживает многие схемы. В качестве примера были реализованы протоколы s/key и схема Шнорра, но подробное описание архитектуры приложения не входит в рамки данной статьи.

2. Описание протокола на основе схемы Шнорра

На примере алгоритма Шнорра покажем детально, как передаются данные между сервером аутентификации, браузером и приложением-менеджером аутентификации. Схема Шнорра — это протокол аутентификации, основанный на сложности дискретного логарифмирования в конечном поле. В нём принимают участие две стороны — Пегги, которая хочет подтвердить свою личность, и Виктор, проверяющий её доказательство. Пегги имеет у себя два ключа, один

из которых закрытый и должен храниться в безопасном месте, а второй общедоступный. Таким образом, Виктор проверяет знание Пегги закрытого ключа по её открытому ключу. Протокол проходит в три шага и считается обладающим свойством нулевого разглашения.

Рис. 1. Диаграмма входа в систему

На рисунке 1 изображена диаграмма, иллюстрирующая процесс входа в систему. На ней пользователь — это браузер клиента. Сервер — это веб-приложение, в котором клиент хочет авторизоваться. Менеджер ключей — это мобильное приложение клиента со всеми его существующими аккаунтами. Можно заметить любопытную особенность — фактически оказалось, что в протоколе участвует три субъекта. Обычно запрос аутентификации, сама аутентификация и все действия авторизованного пользователя происходят в рамках одной сессии. Ключевой особенностью этой системы является разделение процесса аутентификации между двумя субъектами. Благодаря этому привычный менеджер паролей удалось вынести за пределы устройства, с которого будет производиться вход. Итак, запрос на вход в систему отправляется через браузер, сессию которого мы хотим аутентифицировать. Но подтверждение аутентичности вместе с идентификацией проходит через другой канал, в котором приложение на смартфоне доказывает серверу, что та новая открытая сессия должна быть ассоциирована с таким-то пользователем. Информация о сессии попадает в приложение через QR-код, который необходимо сканировать со страницы браузера. После этого пользователь выбирает аккаунт для входа именно на этот сайт.

3. Описание токена

Как было описано выше, через QR-код передаётся некоторая информация об открытой сессии. Предоставим полную структуру токена, который создаёт сервер. Поле «type» используется для того, чтобы приложение автоматически предлагало пользователю либо зарегистрировать новый аккаунт, либо выбрать существующий для входа. Может принимать значение «LOGIN» или «SIGNUP». Поле «domainName» представляет собой имя сервиса. Предполагается, что оно уникально и не меняется. В приложении поиск аккаунтов происходит по этому ключу. Поле «path» содержит в себе путь, по которому нужно обратиться на сервис для осуществления входа или регистрации. Так как в рамках своего приложения разработчики имеют право использовать любую адресацию, то было решено вынести адрес конечной точки в токен. Поле «token» содержит случайное 16-байтное число в шестнадцатеричном формате UUID-v4. Оно используется для поиска сессии в БД на сервере для ассоциации сессии браузера с авторизованным через приложение пользователем. Поле «expiresAt» — дата истечения жизни токена в формате числа — количества миллисекунд со времени UTC. В данной реализации время жизни любого токена 30 секунд. В течение этого времени необходимо осуществить вход или регистрацию в систему. После этого времени токен больше не принимается сервером и удаляется из БД. Поле «algorithm» используется для выбора нужного алгоритма для аутентификации и регистрации в приложении. Так как предполагается, что приложение работает со многими протоколами, поэтому имя протокола было вынесено в это поле. В дополнение к вышеописанным полям было добавлено ещё одно поле «requestInfo», содержащее информацию о том, откуда был вызван запрос на вход или регистрацию. Это было добавлено с целью защиты от фишинга и «обмана, выполненного мафией». Подробнее об этом написано далее, где будут рассмотрены возможные атаки на протокол. В этом поле содержится информация об IP адресе хоста и о клиентском приложении, с которого был выполнен запрос. После заполнения всех полей структуры данные конвертируются в формат JSON и отображаются в браузере как QR-код или как intent-link, специальная ссылка, при переходе по которой открывается приложение. Это удобно если вы хотите зайти на сайт с самого смартфона.

4. Сценарий использования приложения

Опишем процесс взаимодействия пользователя с системой. Для начала сценарий регистрации.

1. Пользователь заходит на сайт и нажимает на кнопку «зарегистрироваться». Сервер в этот момент создаёт специальный token и отдаёт его пользователю в двух форматах — как QR-код и как intent-link.
2. Пользователь с помощью приложения аутентификатора сканирует QR-код или переходит по intent-link.
3. Приложение разбирает токен и приглашает пользователя ввести новый login.

4. Приложение генерирует секрет, соответствующий указанному в токене протоколу, и сохраняет его в локальной шифрованной базе данных. Из секрета создаётся публичная часть ключа и отправляется на сервер.
5. Сервер обрабатывает полученный публичный ключ, сохраняет в БД и автоматически аутентифицирует сессию, в которой была запрошена регистрация.
6. Дополнительные данные, такие как e-mail, имя и прочее, пользователь будет вводить уже на самом сайте, так как эти данные уже не участвуют в протоколе.

Сценарий входа в систему:

1. Пользователь запрашивает token для входа в систему.
2. Пользователь сканирует код. Далее приложение ищет в БД аккаунт для данного сервиса и протокола по доменному имени. Если пользователь имеет несколько аккаунтов, то можно будет выбрать нужный. Кроме того, пользователь будет видеть мета-информацию о системе, сессию браузера которой мы хотим аутентифицировать. В метаинформации могут содержаться сведения о координатах, IP-адресе, операционной системе, т.е. все, что удалось получить из http запроса на вход.
3. Начинается фаза доказательства аутентичности. Если она проходит успешно, то страница в браузере автоматически обновится, а сессия будет авторизована.

Процесс восстановления утерянного секрета не входит в рамки исследования, так как не является частью протокола. Реализовать восстановление кода можно, как обычно, через e-mail или sms.

5. Анализ угроз

Рассмотрим основные угрозы в сети интернет.

1. Атака «человек по середине». Будем рассматривать только пассивную атаку, т.е. прослушивание канала связи. В этом случае злоумышленник может находиться между сервером и клиентом, между устройством ввода и программой или просто подсмотреть через плечо ввод данных. Если от перехвата через сам канал связи нас может защитить HTTPS, то от кейлоггеров построить защиту сложнее, так как выдвигаются требования к конечному устройству. Как описывалось в самом начале статьи, это нельзя гарантировать на публичных компьютерах.
2. Обычно атаку MITM провести сложно, поэтому злоумышленники прибегают к социальной инженерии и фишингу. Суть этих атак заключается в заманивании жертвы на ресурс, очень похожий на оригинальный, где жертва, ничего не подозревая, оставит свои секретные сведения. Этот тип атак направлен прямо на человека, поэтому противостоять им очень сложно.
3. Не стоит недооценивать и вероятность утечки базы с паролями. Обычно для защиты используют хэширование паролей с модификатором, такой подход сильно усложняет криптоанализ, но тем не менее, не делает его

невозможным. И в дополнение, криптоанализ тем легче, чем проще придуманные секреты, так как они с большой вероятностью уже будут в заготовленных таблицах у злоумышленников.

Исходя из построения системы и выдвинутых ранее критериев, система защищена от перечисленных угроз. Но как оказалось, существует одна интересная атака, которой подвержены все протоколы с нулевым разглашением — это «атака, выполненная мафией» или «проблема гроссмейстера».

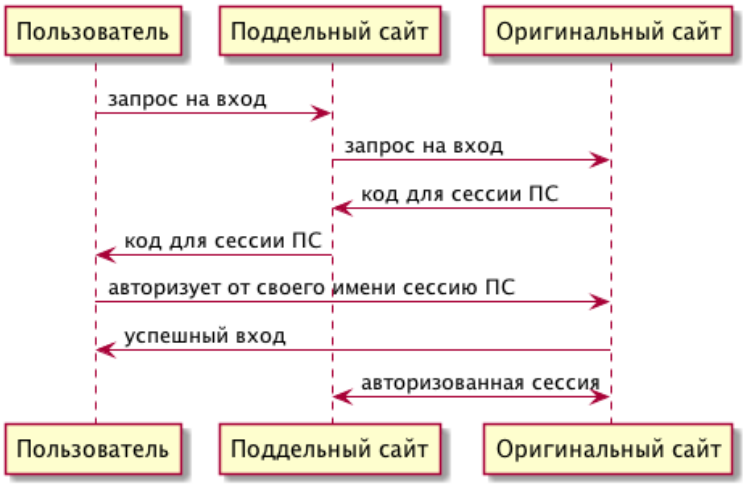

Рис. 2. Атака, выполненная мафией

Суть её заключается в следующем (см. рис. 2):

1. Пользователь заходит на фишинговый сайт. Запрашивает токен для входа.
2. Сайт злоумышленника проксирует этот запрос на оригинальный сайт, тем самым получая токен для своей сессии, и отдаёт его пользователю неизменным.
3. Жертва аутентифицирует сессию злоумышленника, ничего не заметив.

Чтобы избежать этой уязвимости, каждый токен должен содержать в себе информацию о запросе токена. Из всех возможных данных подделать нельзя только IP-сокет источника запроса, из которого можно определить приблизительные координаты клиента. Тогда перед выбором аккаунта в приложении будет показана эта метаинформация, и пользователь сможет понять, подставляют ли его или нет. Кроме того, в приложение можно встроить чёрный список фишинговых сайтов и постоянно его пополнять. Токен в данном случае придётся подписывать сертификатом сервера, чтобы избежать модификации метаинформации.

6. Заключение

Предложенная в статье модель аутентификации с использованием протоколов нулевого разглашения призвана избавиться от недостатков существующих схем и решить проблемы уязвимости к атаке, направленной на самого человека.

Исходя из самих критериев, озвученных в начале, следует, что такие проблемы как пассивное подслушивание трафика, не дадут злоумышленнику ничего. Пользователь не вводит свои секретные данные на каких-то устройствах ввода в незнакомых местах, он всегда использует только свой защищённый смартфон, поэтому вирусы и кейлогеры не представляют опасности. Пользователь не обязан придумывать сложные пароли и помнить их все, таким образом решается проблема слабых паролей и забывания сложных. Так как серверы не должны хранить пароли, то у злоумышленников больше нет вектора атаки на базы данных с пользовательскими секретами. Вместо этого серверы хранят публичные ключи пользователей, и предполагается, что по ним создать закрытую часть ключа очень сложно, по крайней мере намного сложнее, чем перебрать пароли по словарю. Вместе со всем этим следует и удобство в использовании.

Исходные коды сервера аутентификации доступны по адресу: https://github.com/blan4/ZeroKnowledgeProofServer, а приложение клиент — https://github.com/blan4/ZeroKnowledgeProofClient. Тестовый сервер развернут по адресу: https://zkpauth.herokuapp.com/.

Литература

1. Шнайер Б. Прикладная криптография. М. : Триумф, 2002. 541 с.
2. Шнайер Б. Протоколы, алгоритмы, исходные тексты на языке Си. М. : Триумф, 2002. 816 с.
3. Варновский Н.П. Криптография и теория сложности // Математическое просвещение. 1998. Сер. 3. Вып. 2. С. 71–86.
4. Доказательства с нулевым разглашением. URL: `http://citforum.ru/security/cryptography/yaschenko/16.html` (дата обращения: 15.02.2016)

ZERO KNOWLEDGE PROOF AUTHENTICATION ON WEB APPLICATIONS

I.D. Siganov
Postgraduate Student, e-mail: ilya.siganov@gmail.com

Dostoevsky Omsk State University

Abstract. In the article we overview how to implement zero knowledge proof authentication protocol in the web. The proposed system consists of two parts: a server side and Android application. Despite the classical password-based approach, users have to install special application and use it as authentication manager. We use QR-codes to send necessary data from the server to the application, so login experience is just scanning this code. Finally, because of zero knowledge proof features the system is resistant to interception, secret-cracking and fishing, so supposed even to use on insecure public devices.

Keywords: zero knowledge Proof, web authentication, passwordless.

Дата поступления в редакцию: 23.06.2016