САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ - ПРОЦЕССОВ УПРАВЛЕНИЯ
КАФЕДРА КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ И МНОГОПРОЦЕССОРНЫХ СИСТЕМ

Панкратов Станислав Сергеевич

Выпускная квалификационная работа бакалавра

Децентрализованная система аутентификации

Направление 010300
Фундаментальная информатика и информационные технологии

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Корхов В. В.

Санкт-Петербург
2017

Введение

Одним из краеугольных камней информационной безопасности является вопрос аутентификации, то есть проверки подлинности одной из сторон информационного обмена. Традиционно аутентификация вызывает множество сложностей как у обычных пользователей, так и у компаний, запрашивающих подтверждение у пользователей. Неполный список этих проблем включает в себя потери паролей, кражи аккаунтов, утечки пользовательских данных.

С аутентификацией также неразрывно связана более широкая проблема идентификации, то есть наличия у человека законной личности — набора персональных данных о человеке как субъекте права, его правах и обязанностях. Согласно некоммерческой организации ID2020, проводящей ежегодные конференции в штаб-квартире ООН, одна пятая населения Земли живёт без надёжного способа идентифицировать себя, таким образом выпадая из правового поля и становясь уязвимыми для вовлечения в криминальную деятельность [1].

Для большинства людей на данный момент самый главный способ удостоверить свою личность — это паспорта, выдаваемые государственными органами. При этом один и тот же человек может быть гражданином нескольких стран и иметь несколько идентифицирующих документов, а следовательно — несколько личностей.

Возникает ещё больше сложностей, когда речь заходит об электронной идентификации. Многие поставщики услуг в сети Интернет требуют регистрации, хранения пользовательских данных на своих серверах. Каждый такой аккаунт вносит вклад в раздробленность личностей. Возникает обоснованное желание разработать единый механизм пользовательской

идентификации и аутентификации. И такие механизмы уже есть; главная их проблема в том, что среди них нет однозначно глобального и универсального, а главное — прозрачного. Такие механизмы, как правило, полностью полагаются на какую-либо коммерческую организацию, которой мы доверяем свои данные и электронные личности. В данной работе будут рассмотрены уже существующие решения для единой аутентификации пользователей и предложим альтернативный вариант на базе технологии blockchain, призванной устранить зависимость от одной удостоверяющей компании и децентрализовать механизм выдачи законных личностей в сети Интернет.

Постановка задачи

Целью данной работы является разработка для сети Интернет единой и универсальной системы идентификации и аутентификации, удовлетворяющей критериям, сформулированным ниже. Под системой идентификации будем понимать:

- **Хранилище личностей**, каждая из которых ассоциируется с определённым реальным человеком.
- **Способ**, которым человек может доказать владение этой личностью (аутентификация).

Под системой аутентификации будем понимать набор следующих структур:

- **Хранилище данных**, в котором содержатся зашифрованные персональные данные пользователей и их публичные ключи.
- **Узлы сети**, обслуживающие хранилище данных и отвечающие за логику процесса аутентификации.
- Легковесное **приложение-клиент**, способное принимать и подтверждать запросы на аутентификацию, связанные с конкретной личностью.
- **Протокол**, по которому взаимодействуют перечисленные выше части системы.
- **Библиотеки**, с помощью которых разработчик легко сможет встроить аутентификацию по протоколу на свой сайт.

Вместе с этим представленная здесь система идентификации и аутентификации должна отвечать следующим критериям:

- **Универсальность:** имея личность в данной системе, человек должен иметь возможность аутентифицироваться на любом подключенном к системе веб-сайте.
- **Безопасность:** пользователи сами контролируют безопасность доступа к своим именам и не зависят от политик безопасности третьих сторон.
- **Децентрализованность:** ни одно физическое или юридическое лицо не может единолично контролировать систему и процесс аутентификации пользователей. Владение именем подтверждается или отвергается на основе коллективного решения участников сети. Ни один узел не является точкой отказа.

Поэтому в задачи работы входит:

1. Анализ существующих протоколов аутентификации, их преимуществ и недостатков.
2. Подбор подходящего протокола аутентификации и имеющейся инфраструктуры (если такая у него имеется): хранилище личностей, хранилище данных, узлы сети, приложение-клиент.
3. Разработка библиотек разработчика и распространение их по каналам Open Source.
4. Разработка тестового сайта, использующего систему аутентификации.
5. Анализ решения на указанные выше критерии.

Обзор литературы

Учебник [7] был использован как пособие по технологии блокчейн. Особенно стоит отметить главу 9 (применение блокчейна, включая DNS-криптовалюту Namecoin) и 10 (безопасность блокчейна).

Статьи [12][14][17] — главные источники информации о Blockstack. В них описывается архитектура сети Blockstack, мотивация и проблемы, с которыми столкнулись разработчики. Описаны принципы виртуального блокчейна и почему в качестве основного блокчейна был выбран именно Bitcoin. Ветки на форуме Blockstack [13][15] содержат информацию о планах разработчиков и возможностях для улучшения сети, а открытые репозитории [19][20] дают возможность понять реализацию отдельных компонентов системы Blockstack.

Статистика LoginRadius [28] даёт представление о масштабах использования протоколов аутентификации типа Single Sign On.

Работа [34] по шагам описывает процесс взлома веб-сервера через уязвимость Heartbleed. Статья [35] оценивает потенциальные риски, связанные с этой уязвимостью.

Глава 1. Подходы к аутентификации

В первую очередь необходимо рассмотреть несколько основных подходов к аутентификации, проанализировать их достоинства и недостатки.

Здесь в первую очередь будут изучены различные методы хранения секретной пользовательской информации, такой как пароли и приватные криптографические ключи, и по ним классифицировать методы аутентификации. Поскольку протоколов аутентификации существует бесчисленное множество, для каждого способа хранения данных будет рассмотрено только несколько примеров протоколов, которые находят широкое применение в повседневном использовании сети Интернет.

1.1 Хранение паролей в незашифрованном виде на стороне поставщика услуг

Достоинства:

- **Простота реализации.**
- **Возможность восстановления старого, утерянного пароля.**

Недостатки:

- **Неустойчивость к взлому хранилища.** Система аутентификации с таким методом хранения паролей является самой незащищённой. При взломе поставщика услуг все пароли становятся доступными хакеру, и он может в любое время аутентифицироваться под видом другого пользователя. Пользовательские данные находятся под большой угрозой.
- **Неуниверсальность.** Пользователь вынужден каждый раз заводить новый аккаунт для каждого поставщика услуг с таким методом аутентификации. Поскольку заводить одинаковые пары логин-пароль

крайне небезопасно (а часто и просто невозможно), это создаёт большую путаницу.

1.2. Хранение паролей в зашифрованном виде на стороне поставщика услуг

Такие системы похожи на предыдущие за исключением того, что поставщик услуг хранит на своих серверах пароли не в открытом виде, а в виде секретного ключа, который генерируется единожды на основе пароля. Для этого используются специальные алгоритмы, например, PBKDF2 [2].

Каждый раз, когда пользователь вводит свой пароль, система заново вычисляет по нему ключ, и если он совпадает с ранее сохранённым для этого пользователя, то пользователь успешно проходит аутентификацию.

Достоинства:

- **Устойчивость к взлому хранилища.** Даже в случае взлома поставщика услуг хакер получает не сами пароли, а результаты применения алгоритмов формирования ключа к паролям. Такие алгоритмы специально разработаны так, чтобы усложнить получение пароля.

Недостатки:

- **Неуниверсальность.** Пользователи всё ещё вынуждены заводить отдельные аккаунты для каждого поставщика услуг.

1.3 OAuth, OpenID

Два открытых стандарта, позволяющих делегировать аутентификацию некоторой третьей стороне — identity provider, поставщику личностей. Как правило, в качестве identity provider выступает огромная корпорация, для

сервисов которой у большинства пользователей уже есть аккаунт (например, Google, Facebook).

Достоинства:

- **Универсальность.** Имея аккаунт у identity provider, можно заходить под своим именем на любые сайты, поддерживающие OpenID или OAuth. Такой аккаунт служит универсальной сетевой личностью и позволяет использовать одни и те же способы аутентификации на множестве площадок.
- **Открытость.** OpenID и OAuth — полностью открытые стандарты, не защищённые патентами и копирайтом.

Недостатки:

- **Доверие к третьей стороне.** Пользуясь OpenID или OAuth, пользователи вынуждены доверять свою аутентификацию некоторой третьей стороны. Из этого вытекает:
    - **Неустойчивость к взломам.** Пользователь не может контролировать хранение своих данных на серверах третьей стороны, и теоретически они могут храниться там в открытом виде.
    - **Отсутствие анонимности.** Identity provider видит и логирует действия клиента: когда он аутентифицировался, куда, с какого ip-адреса, как часто.
    - **Единая точка отказа.** Если сервер аутентификации третьей стороны окажется на время недоступным, или третья сторона вовсе прекратит существование, пользователь потеряет возможность аутентифицироваться под своим единым именем.
    - **В любой момент третья сторона может воспользоваться именем пользователя, так как именно она им и владеет.**

    - **В любой момент третья сторона может конфисковать или заблокировать личность пользователя.**

  Этого недостатка можно избежать, если запустить свой собственный identity server (в случае OpenID). Однако это требует определённой технической подготовки и вычислительных ресурсов.
- **Не все сайты могут поддерживать предпочтительного для пользователя identity provider (в случае OAuth).**

1.4 Mozilla Persona

Единая система аутентификации, разработанная в Mozilla под влиянием идей VerifiedEmailProtocol. Аутентификация строится на основе сертификата, который хранится в памяти браузера пользователя. Каждый сертификат содержит верифицированный email пользователя и криптографически доказывает, что пользователь действительно владеет этим адресом. Каждые 24 часа сертификат необходимо обновлять, отправив пароль на сервера Mozilla. Однако обновлённый сертификат следующие 24 часа служит средством аутентификации на всех сайтах, поддерживающих Mozilla Persona, и не требует ввода пароля.

Достоинства:

- **Универсальность.**
- **Открытость.**
- **Относительная анонимность.** В силу особенностей протокола, identity provider (в данном случае Mozilla) видит только факт обновления сертификата. Identity provider не способен понять, где и когда этот сертификат используется для аутентификации.

Недостатки:

- **Доверие к третьей стороне.** В случае с Mozilla Persona, из этого следует меньше недостатков, чем у OpenID и OAuth. Протокол разработан так, чтобы обеспечивать базовую анонимность и устойчивость к взломам.
- **Отсутствие поддержки со стороны разработчика.** Mozilla прекратила разработку проекта и отдала его на попечение сообществу.

1.5 Системы аутентификации, основанные на blockchain

Blockchain — распределённая база данных, которую впервые начали использовать для криптовалют, таких как Bitcoin, создатель которой, Сатоши Накамото, впервые и ввёл этот термин. Blockchain — база данных, в криптовалютах она хранит последовательность всех денежных транзакций. Копия этой базы данных живёт одновременно на всех узлах системы (хотя для производительности некоторые узлы могут иметь урезанные права и не хранить базу в полном виде). Добавление к блокчейну новой транзакции возможно только на основе консенсуса, то есть совместной верификации несколькими узлами системы. [7] Впоследствии технология blockchain нашла применение во многих сферах, где требовалась надёжность, защищённость и отсутствие единой стороны, которой нужно доверять. В последнее время Blockchain начал использоваться и в области идентификации и аутентификации: Microsoft заявили о разработке распределённой системы идентификации совместно с двумя компаниями, специализирующимися на blockchain — Blockstack Labs и ConsenSys [3].

Решения этих двух компаний, Blockstack и uPort, являются достаточно перспективными разработками, однако также не лишены недостатков.

1.5.1 Blockstack

Blockstack изначально является системой бессерверных веб-приложений, которая позволяет создавать одностраничные сайты и бесплатно выкладывать их в блокчейн вместо традиционного хостинга. В процессе была разработана концепция Blockchain ID, с помощью которой можно создать в блокчейне публичный профиль и с помощью него аутентифицировать себя [4].

Система Blockstack использует уже существующую инфраструктуру Bitcoin. Каждая транзакция Bitcoin может содержать опциональное поле OP_RETURN, в котором хранится произвольная информация. Благодаря такой возможности, Blockstack строит на основе блокчейна Bitcoin виртуальный блокчейн (virtualchain), который состоит только из транзакций с непустыми полями OP_RETURN, содержащими сообщения нужного формата. Такие транзакции передают могут передавать информацию о следующих операциях:

- **Регистрация имени**. В ходе этой операции имя привязывается к зарегистрировавшему его биткоин-адресу.
- **Обновление профиля.** В профиле могут быть абсолютно произвольные поля.
- **Передачи имени другому адресу.**

Таким образом, виртуальный блокчейн Blockstack играет роль, похожую на роль DNS.

Достоинства:

- **Универсальность.** Зарегистрировав имя в системе Blockstack, пользователь может аутентифицироваться под этим именем на любых сайтах, на которых работает протокол аутентификации Blockstack.
- **Открытость.** Разработка системы ведётся по принципам open source.

- **Анонимность.** Информация о том, когда и куда аутентифицируется пользователь, доступна только пользователю и сайту, куда он аутентифицируется.
- **Отсутствие доверия к третьей стороне.** Система Blockstack работает на базе развитого блокчейна Bitcoin. Это значит, что:
    - **Систему практически невозможно взломать**: в такой системе вообще нет такого понятия, как взлом или утечка данных. Информация о зарегистрированных именах и так публична и доступна всем. Украсть можно только само имя, но благодаря криптографическим основам блокчейна это возможно только в том случае, если злоумышленник получит в руки приватный ключ пользователя. Поскольку приватный ключ хранится только на стороне пользователя, и ключ зашифрован с помощью его мастер-пароля, эта ситуация равноценна компрометации одновременно хранилища пользователя и его мастер-пароля, а не системы в целом.
    - **В системе нет единой точки отказа.** Система может выйти из строя только если выйдут из строя все узлы блокчейна Bitcoin.
    - **Никто не может воспользоваться именем пользователя кроме него самого.**
    - **Никто не может конфисковать или заблокировать личность пользователя.** Все действия в блокчейне принимаются на основе консенсуса нескольких узлов. Конфискация имени пользователя возможна только при одобрении значительного числа узлов сети, что равноценно компрометации нескольких тысяч серверов по всему миру.

Недостатки:

- **Новизна.** Протокол всё ещё находится в стадии разработки. До хоть сколько-нибудь рабочего состояния основные части системы аутентификации привели только к концу 2016 года. Система может работает нестабильно, с ошибками, отсутствует важная функциональность.
- **Неудобство для пользователя.** Хотя разработчики стремятся исправить эту проблему, прямо сейчас работа с протоколом достаточно неудобна. Пользователю необходимо скачать, запустить и настроить локально приложение-клиент, которое на момент написания работы в готовом виде существует только для операционной системы OS X; пользователи Windows и Linux вынуждены скачивать и настраивать части системы вручную. Пользователю также необходимо заказать себе хотя бы одно имя; в силу особенностей протокола, система требует от пользователя небольшую плату за каждую операцию с виртуальным блокчейном, в том числе за регистрацию имени. Это значит, что пользователю надо купить биткоины и передать их на свой адрес. Нет полноценных библиотек для встраивания аутентификации на веб-сайте.

1.5.2 uPort

uPort, в отличие от Blockstack, является специализированной системой, заточенной именно под идентификацию и аутентификацию. Создатели обещают возможность создавать отдельные личности для разных аспектов жизни человека (банковская деятельность, аккаунты в социальных сетях, профессиональная деятельность и т.д.). Личность можно будет использовать как

единый аккаунт для аутентификации, подписывать ею заявления, выставлять напоказ свой публичный профиль.

uPort всё ещё находится в разработке, создатели на данный момент выпустили White Paper и приглашают всех желающих поучаствовать в тесте альфа-версии. Система построена на основе умных контрактов на платформе Ethereum. Это даёт создателям богатые возможности по использованию уже существующего кода, встроенного в платформу.

Система состоит из [5]:

1. Мобильного клиентского приложения, хранящего приватный ключ пользователя.
2. Библиотек, с помощью которых разработчики могут подключить аутентификацию к своему сайту.
3. Умных контрактов Ethereum, содержащих логику приложения.

Хотя uPort выглядит достаточно многообещающей разработкой, сейчас она находится в самой начальной стадии. Главным недостатком является закрытый код мобильных приложений uPort, что не даёт полностью понять механизм работы, и стадия альфа-тестирования, и как следствие — нестабильная работа системы.

Достоинства:

- **Универсальность.**
- **Анонимность.**
- **Отсутствие доверия к третьей стороне.**
- **Удобство для пользователя.** Из всех систем аутентификации на основе blockchain uPort имеет наиболее удобное мобильное приложение-клиент.

Недостатки:

- **Новизна.** Система находится в стадии полузакрытой альфы: поучаствовать в тесте может любой желающий, но если тот зарегистрируется в программе тестирования. Мобильное приложение работает нестабильно и активно дорабатывается.
- **Частично закрытый код.** Репозитории клиентских приложений для iOS и Android закрытые, хотя разработчики обещают в скором времени опубликовать исходный код.

1.5.3 EMCSSL

Решение, основанное на криптовалюте EmerCoin. Эта криптовалюта продолжает идеи NameCoin — криптовалюты, на которой строится система альтернативных корневых DNS-серверов. EmerCoin имеет распределённое хранилище общего назначения с открытым интерфейсом: EmerCoin NVS. Это позволяет хранить в блокчейне информацию любого рода.

Одним из применений такого блокчейна является выпуск SSL-сертификатов. На них и строится аутентификация пользователей с помощью EmerCoin. В отличие от Blockstack и uPort, которые разрабатывают собственные клиентские протоколы аутентификации, написанные на Javascript и запущенные в браузере, EmerCoin используют существующую технологию аутентификации с помощью клиентских SSL-сертификатов.

Разница между обычными клиентскими сертификатами и сертификатами EMCSSL в том, что в качестве центра сертификации выступает не некоторая третья доверенная сторона, а сам блокчейн. Главный секрет SSL — приватный ключ центра сертификации — в EMCSSL является открытым, и валидация сертификата осуществляется не через подпись от CA, а через хэш в блокчейне.

Такая система не раскрывает секрет пользователя в процессе аутентификации серверу и использует децентрализованное хранение учётных записей. Как и у Blockstack и uPort, у EMCSSL есть возможность прикрепить к имени произвольную информацию в виде профиля [7].

Достоинства:

- **Универсальность.**
- **Открытость.**
- **Анонимность.**
- **Использование существующего протокола (аутентификация через клиентский SSL-сертификат).** Это серьёзно облегчает жизнь разработчикам сайтов, на которых будет использоваться такая модель аутентификации.
- **Отсутствие доверия к третьей стороне.**

Недостатки:

- **Направленность на специфическую целевую аудиторию технических специалистов.** Прямо сейчас у системы нет достаточно удобного способа для конечных пользователей зарегистрировать имя и использовать его как средство аутентификации. Пользователи вынуждены пройти процедуру генерации сертификата и загрузки его в браузер [8], и хотя разработчики предоставляют подробную инструкцию для совершения этих действий, она пока ещё направлена больше на людей хоть сколько-нибудь разбирающихся в криптографии.
- **Новизна.** Единственный сервис, который призван помочь конечному пользователю зарегистрироваться и использовать своё имя как средство аутентификации, Remme, находится в стадии закрытой беты [9].

1.6 Заключение

Большинство подходов к аутентификации, используемые на данный момент, можно разделить на два типа, если говорить об их преимуществах и недостатках:

1. Зрелые и надёжные решения, которые тем не менее страдают от недостатков, из-за которых такие протоколы нельзя использовать как действительно универсальное средство идентификации во всех сферах жизни. Главным недостатком таких решений является зависимость от третьей стороны. К таким решениям относятся хранение зашифрованных паролей на стороне поставщика услуг, OAuth, OpenID, Mozilla Persona.
2. Перспективные новые разработки на основе технологии blockchain. Они преодолевают главные недостатки первого типа решений: централизованность и зависимость от третьей стороны. Все решения о хранении имени и принадлежности его определённому человеку в таких системах принимаются консенсусом многих узлов сети. К таким решениям относятся Blockstack, uPort и EMCSLL. Однако эти системы находятся в стадии ранней разработки и пока ещё не могут удовлетворить все запросы пользователей.

Система, представленная в этой работе, использует в качестве основы протокол Blockstack и дополняет его специально разработанными в рамках этой работы библиотеками для разработчиков веб-приложений, с помощью которых они могут встроить в свои приложения аутентификацию через Blockstack. Протокол Blockstack был выбран по следующим причинам:

- Он работает на самом развитом блокчейне Bitcoin. uPort работает на Ethereum, EMCSSL — на EmerCoin.

- В отличие от uPort, альфа-версия которого была опубликована только в январе 2017 года, а мобильное приложение всё ещё находится в закрытом репозитории, разработка Blockstack изначально велась открыто, что позволило сразу оценить потенциал экосистемы Blockstack. Клиентское веб-приложение EMCSSL, Remme, тоже всё ещё находится в стадии закрытой беты.

Глава 2. Архитектура системы

2.1 Обзор

Система, представленная в рамках этой работы, выполняет следующие цели:

1. Обеспечивает безопасные и распределённые методы хранения, регистрации и передачи имён пользователей.
2. Обеспечивает локальное хранение приватного ключа пользователя, с помощью которого тот может доказать своё владение именем.
3. Предоставляет возможность пользователю подписывать своим приватным ключом запросы на аутентификацию и таким образом заходить под своим именем на сайты, поддерживающие данный протокол.
4. Содержит библиотеки, с помощью которых разработчики могут легко настроить аутентификацию пользователей на своём веб-сайте.

Система состоит из следующих компонентов:

1. Инфраструктура Blockstack:
    a. Virtualchain
    b. Blockstack Core
    c. Blockstack Portal
    d. Onename
2. Библиотека на Python, с помощью которой можно создавать и подписывать Authentication Request и Authentication Response объекты.
3. Приложение для фреймворка Django, реализующее аутентификацию с использованием протокола Blockstack на стороне сервера.

В рамках данной работы были реализованы вторая и третья части системы (Python-библиотека и Django-приложение). Первый компонент, то есть инфраструктура Blockstack, является открытой разработкой Blockstack Inc., и в рамках данной работы она была изучена и использована как часть системы. Также был написан тестовый сайт, использующий систему аутентификации Blockstack.

2.2 Инфраструктура Blockstack

2.2.1 Введение

Blockstack — это целый комплекс программного обеспечения, находящегося в разработке, конечная цель которой — создание нового децентрализованного интернета. Это стартап из Нью-Йорка, сотрудничающий с Microsoft и получивший инвестиции в размере 4 миллионов долларов в инвестиционном раунде Y Combinator [10].

Целью компании является построение в блокчейне отдельной системы имён, похожей на DNS. Это позволяет перевести такой важный и фундаментальный уровень всемирной паутины, как система имён, в децентрализованную область. Если перевести все имена в блокчейн, то устранятся единичные точки отказа и стороны, которым мы вынуждены доверять, такие как корневые DNS-узлы и Identity Providers. Имена Blockstack можно использовать как для адресации веб-сайтов, так и для аутентификации пользователей. Фактически, имя в блокчейне может быть как именем сайта, так и именем пользователя, или и тем, и другим.

Приложения Blockstack — приложения без сервера, одностраничные веб-сайты, состоящие из статических файлов (HTML, CSS, Javascript). Эти

статические файлы размещаются в личном облачном хранилище разработчика (Google Drive, Dropbox и т.п.) [11].

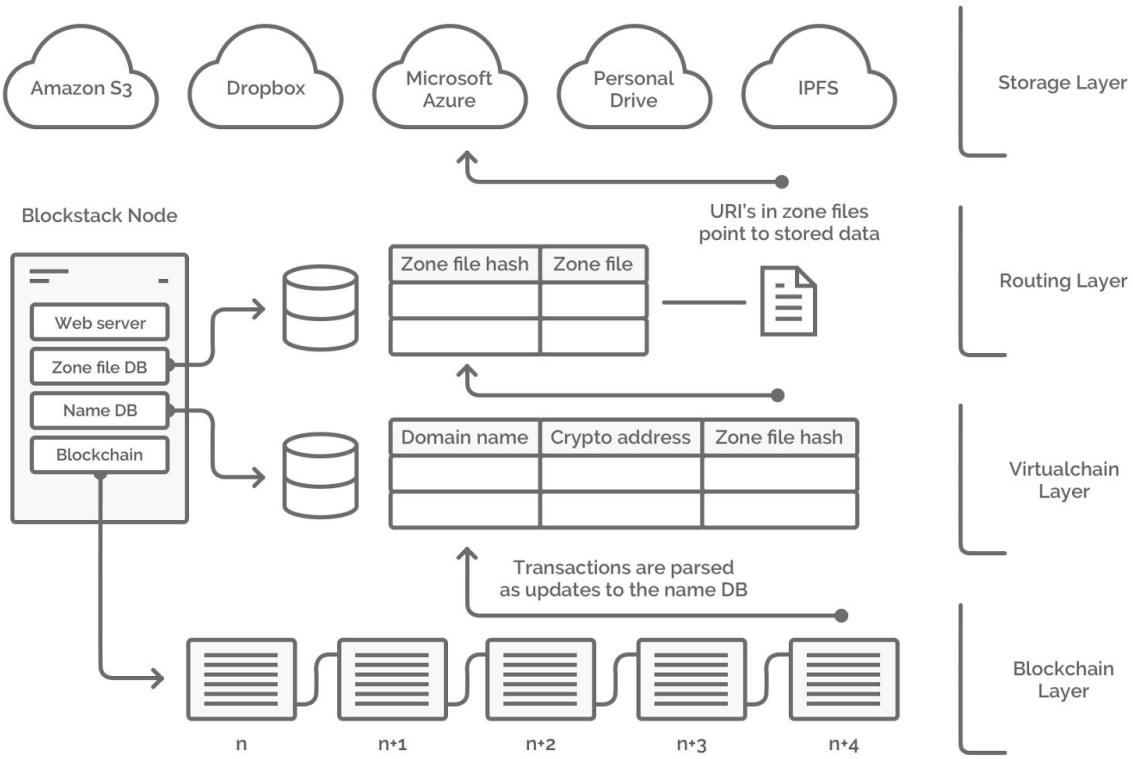

Рис. 1: Обзор архитектуры Blockstack.

Инфраструктура Blockstack разбивается на несколько слоёв [12].

Ключевой технологией инфраструктуры Blockstack является блокчейн Bitcoin, на котором строится система альтернативных корневых DNS-серверов. Blockchain Layer инфраструктуры Blockstack — это в чистом виде блокчейн Bitcoin; благодаря сильной криптографической основе, децентрализованной природе и механизмам консенсуса он обеспечивает распределённость данных, безопасность и тот факт, что у пользователя нельзя отобрать имя без компрометации его приватного ключа.

Virtualchain Layer — надстройка над блокчейном, которая позволяет хранить в блокчейне информацию, связанную с протоколом Blockstack: имена, их владельцев, операции над именами, информацию об именах.

К каждому имени прилагается Zone File в точно таком же формате, в котором он используется в DNS. Он содержит ссылки на информацию об имени, например, профиль (если это имя пользователя) или статические файлы [13][14]. Zone Files составляют Routing Layer.

На всех прежде упомятых трёх уровнях работает Blockstack Node. Это, во-первых, обычный майнер Bitcoin, то есть Bitcoin Full Node, который индексирует блокчейн. Во-вторых, такой сервер запускает у себя Blockstack Core: основную библиотеку для работы с Blockstack. Эта библиотека запускается как сервер, индексирующий Virtualchain и распознающий Zone File.

Ссылки в Zone File ведут на некоторое удалённое хранилище, которое может содержать как легковесные текстовые описания профилей, так и тяжеловесные статические файлы. Этот последний слой называется Storage Layer.

Наконец, Blockstack Portal — пользовательское приложение, которое отвечает за доступ пользователей к Blockstack-сайтам и аутентификацию.

На данный момент ещё не готова та часть системы, которая позволяет пользователю по имени сайта получить к нему доступ, то есть загрузить статические файлы по пути в Zone File. Однако уже сейчас можно опробовать ту часть системы, которая отвечает за индивидуальные аккаунты и их аутентификацию на любых сайтах, не только использующих Blockstack. Эта возможность и будет использована в данной работе для создания системы аутентификации в применении к обычным серверным сайтам [15].

Для простоты в дальнейшем процесс аутентификации через имя в системе Blockstack будет называться просто “протоколом Blockstack”, хотя сами разработчики не используют такой термин, так как название “Blockstack” относится к целому комплексу ПО, связанному не только с аутентификацией.

2.2.2 Virtualchain

Разработчики Blockstack ввели новое понятие виртуального блокчейна. Это слой программных компонентов поверх обычного блокчейна, который вводит новую функциональность без изменений блокчейна, лежащего в основе. Все новые операции вводятся на уровне виртуального блокчейна и кодируются в мета-данные транзакций обычного блокчейна. И хотя обычные узлы блокчейна тоже видят эту информацию, логическое значение она имеет только для узлов виртуального блокчейна [16].

Примерами операций являются предзаказ имени, регистрация имени, смена владельца, отзыв имени, продление регистрации имени и обновление Zone File.

В качестве конкретного блокчейна, на котором работает система имён Blockstack, был выбран блокчейн Bitcoin как самый развитый блокчейн сети. Это связано с тем, что маленькие блокчейны (с небольшим количеством майнеров) сильнее подвержены риску атаки: злоумышленнику необходимы меньшие вычислительные мощности по сравнению с крупным блокчейном. Кроме того, маленькие блокчейны (такие как Namecoin, который так же используется в качестве системы альтернативных DNS-серверов) уязвимы перед атакой 51%, когда один майнер или группа майнеров в сговоре контролируют более 51% вычислительной мощности блокчейна, что ведёт к захвату контроля над блокчейном одной стороной [17].

Информация об операциях Blockstack встраивается через поле OP_RETURN транзакций Bitcoin. Согласно сайту OP_RETURN Stats, собирающему информацию о кастомных протоколах на основе блокчейна Bitcoin, протокол Blockstack является крупнейшим по объёму транзакций среди нефинансовых приложений [18].

2.2.3 Blockstack Core

Это основное CLI-приложение, написанное на Python, которое осуществляет значительную часть работы с протоколом, осуществляемой на сервере.

Blockstack Core включает в себя:

1. Blockstack API. Предоставляет REST API для получения информации об инфраструктуре Blockstack: информация об именах, кошельках, профилях, узлах и т.п. Это приложение необходимо для пользователя, так как API используется в Blockstack Portal. Blockstack Core, запущенный в режиме API, не превращает локальную машину пользователя в полный узел Blockstack и таким образом не загружает локально весь блокчейн; вместо этого он полагается на полный узел, доступный по адресу node.blockstack.org
2. Blockstack Client. Консольная утилита, с помощью которой пользователь может управлять своим кошельком и привязанным к нему именами. К консольному клиенту привязаны два Bitcoin-кошелька: owner и payment. Owner-кошелёк владеет именами, именно его адрес указывается при передачи имени аккаунта пользователя. На payment-адрес начисляются биткоины, чтобы с помощью них можно было платить небольшую

техническую плату за каждую транзакцию (регистрация имени, обновление Zone File, передача имени).

3. Blockstack Registrar. Библиотека для Identity Provider'ов, с помощью которого можно легко регистрировать новые имена, заполнять профили пользователей и отправлять эти имена на их личные адреса.
4. Blockstack Core. Полный узел Blockstack, индексирующий транзакции блокчейна Bitcoin и виртуального блокчейна.

В этой работе в первую очередь нас будут интересовать API и Client, так как они необходимы для работы Blockstack Portal.

2.2.4 Blockstack Portal

Blockstack Portal — приложение-сервер, запускающееся локально на компьютере пользователя [20]. С помощью него пользователь может:

1. Завести себе новый owner-кошелёк.
2. Зайти со старого owner-кошелька с помощью backup phrase.
3. Приобрести себе имя.
4. Заполнить профиль для этого имени.
5. Принимать запросы на аутентификацию, логиниться на веб-сайтах.

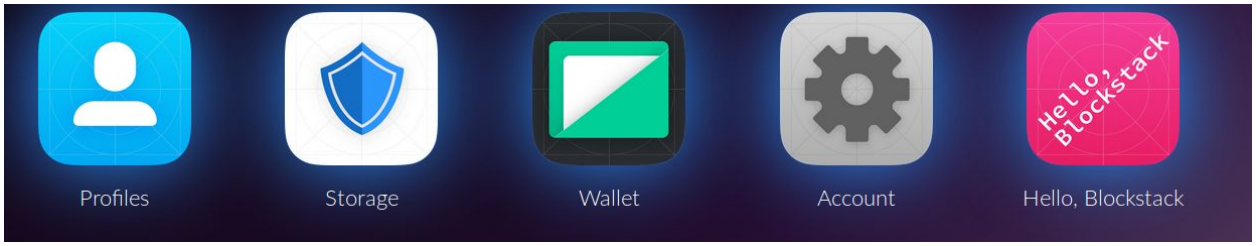

Рис. 2: домашняя страница Blockstack Portal.

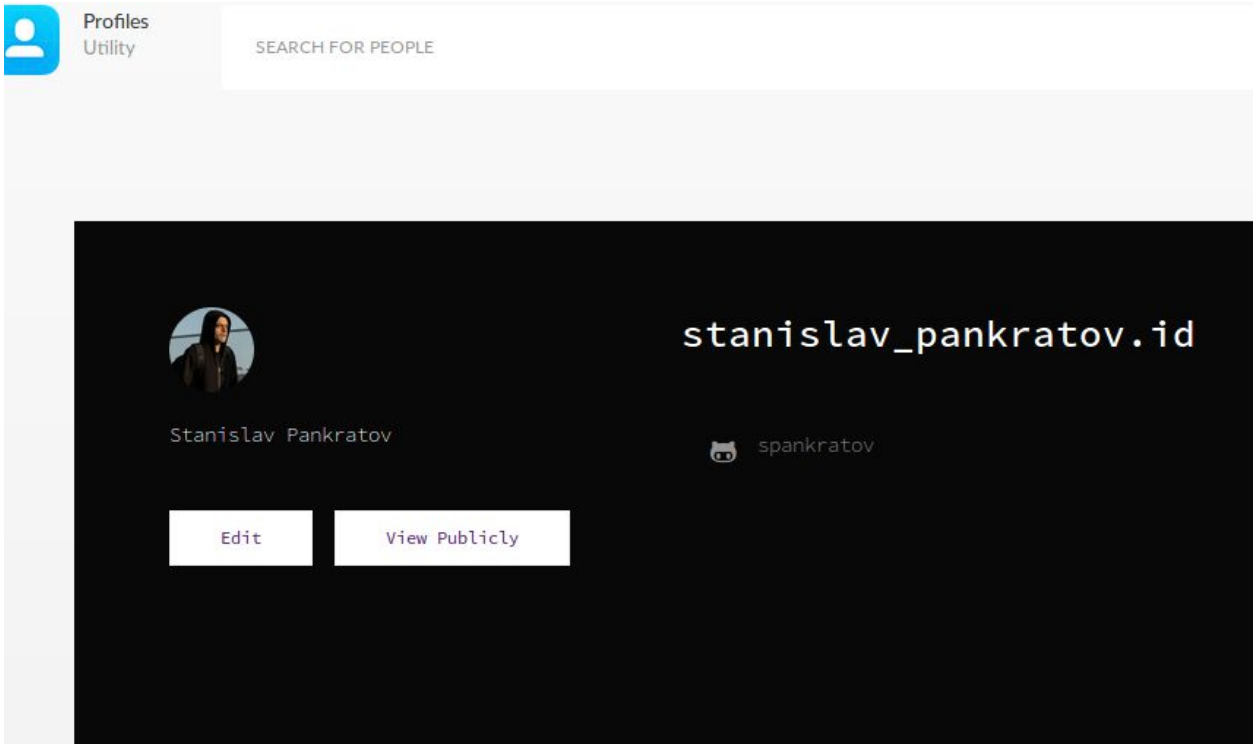

Рис. 3: страница профиля пользователя в Blockstack Portal.

2.2.5 Onename

Identity Provider от Blockstack, располагается по адресу https://onename.com

С помощью этого сайта можно зарегистрировать себе имя, редактировать профиль, верифицировать своё имя в Facebook, Twitter и Github, а также перенести созданноё имя на локальный owner-кошелёк. Любая плата за транзакции платится самим сайтом.

Из-за особенностей работы с блокчейном, имя может довольно долго находиться в процессе регистрации (до нескольких дней): оно будет отображаться на сайте Onename, но не будет фактически зафиксировано в блокчейне. Это связано с тем, что любые транзакции в блокчейне сперва должны быть объединены в блок и пройти процедуру подтверждения другими майнерами.

2.3 Python-библиотека

В отличие от ранее описанных технологий, Python-библиотека была разработана в рамках этой работы.

Библиотека позволяет:

1. Сгенерировать запрос на аутентификацию со стороны сервера и ответ на этот запрос согласно заданному JSON-формату.
2. Подписать запрос ключом сервера.
3. Подписать ответ на запрос ключом пользователя.
4. Преобразовать запрос и ответ на запрос в форму JWT и обратно в JSON.
5. Верифицировать запрос на аутентификацию.
6. Верифицировать ответ на запрос аутентификации.

2.4 Django-приложение

Django-приложение также было разработано в рамках данной работы. Это удобный способ встроить на веб-сайт аутентификацию пользователей через протокол Blockstack. Используя возможности ранее написанной Python-библиотеки, приложение позволяет:

1. С помощью первого URL настроить процесс аутентификации: при переходе по этому URL, сервер с помощью Python-библиотеки генерирует запрос на аутентификацию и перенаправляет пользователя к Blockstack Portal.
2. С помощью второго URL (callback URL) приложение принимает ответ на запрос аутентификации, верифицирует его и в случае успеха аутентифицирует пользователя.
3. По имени пользователя получить его Blockstack-профиль.

Глава 3. Реализация протокола аутентификации Blockstack

3.1 Обзор

Реализация поставленных задач включает в себя:

1. Анализ открытого протокола аутентификации Blockstack, его инфраструктуры, выявление достоинств и недостатков, использование его для реализации библиотек разработчика blockchainauth и django-blockstack.
2. Разработку на основе протокола Blockstack Python-библиотеки blockchainauth.
3. Разработку Django-приложения django-blockstack, использующего blockchainauth.
4. Разработку тестового веб-сайта, использующего django-blockstack для аутентификации пользователей.

3.2 Термины

- **Имя** — строка, служащая для идентификации в сети Blockstack. Может использоваться как для идентификации пользователей, так и доменов, но в рамках данной работы рассматривается только идентификация пользователей.
- **Пользователь** — одна из сторон процесса аутентификации, некоторое лицо, во владении которого находится имя. Пользователь характеризуется парой закрытого и открытого ключа.

- **Аутентификация** — в данном случае процесс, при котором пользователь доказывает своё владение именем.
- **Закрытый ключ** — HEX-строка, главный криптографический секрет, который ассоциируется с пользователем. В случае Blockstack закрытый ключ является также закрытым ключом Bitcoin, то есть по нему же осуществляется доступ к Bitcoin-кошельку. Закрытым ключом пользователь подписывает все операции, связанные с его именем, в том числе аутентификационные операции.
- **Открытый ключ** — HEX-строка, публичная часть идентификации пользователя на уровне блокчейна. С помощью него приложение может верифицировать ответ на аутентификацию, то есть проверить, действительно ли ответ подписан закрытым ключом пользователя.
- **Адрес** — адрес Bitcoin, связанный с открытым ключом пользователя. Все зарегистрированные имена принадлежат какому-нибудь адресу.
- **Приложение** — одна из сторон аутентификации, некоторый сервис, чаще всего веб-сайт, для полноценной работы с которым пользователю необходимо аутентифицироваться. Здесь рассматриваются только те приложения, которые поддерживают протокол Blockstack.
- **Запрос на аутентификацию** — генерируемая на стороне приложения строка, в которой закодирована информация о приложении и некоторая мета-информация, необходимая пользователю для аутентификации. Для большей безопасности запрос может подписываться закрытым ключом самого приложения.
- **Ответ на запрос аутентификации** — генерируемая на стороне пользователя строка, в которой закодирована информация о пользователе и некоторая мета-информация, необходимая приложению для того, чтобы

проверить подлинность ответа и аутентифицировать пользователя. Ответ обязательно подписывается закрытым ключом пользователя.

- **Owner-кошелёк** — Bitcoin-кошелёк, которому принадлежат имена. Именно на адрес этого кошелька следует отправлять запросы на передачу имени от адреса к адресу.
- **Payment-кошелёк** — Bitcoin-кошелёк, с которого необходимо платить небольшую сервисную плату за любую операцию с именами, подразумевающую изменение блокчейна и новую транзакцию (регистрация, предзаказ, передачи имени, обновление Zone-файла). Именно на адрес этого кошелька следует отправлять биткоины. Пользователь всегда обладает как минимум одной парой owner-кошелька и payment-кошелька.

3.3 Процесс аутентификации

3.3.1 Со стороны пользователя

Клиентом для пользователя является Blockstack Portal, для работы которого также необходим Blockstack Core.

Инструкции по установке расположены в публичном репозитории на Github: https://github.com/blockstack/blockstack-portal

Чтобы запустить у себя Blockstack Portal, пользователю необходимо:

1. Установить Blockstack Core.
2. Настроить Blockstack Core, создав пару owner- и payment-кошельков. Из-за особенностей программного обеспечения Blockstack на данный момент owner-кошелёк из Blockstack Core и owner-кошелёк из Blockstack

Portal несовместимы, при настройке Blockstack Portal пользователю придётся сгенерировать новый owner-кошелёк.

3. Запустить Blockstack Core в режиме API, подключившись к полному узлу Blockstack.
4. Скачать Blockstack Portal.
5. Запустить CORS-прокси.
6. Запустить Blockstack Portal.
7. Завести в Blockstack Portal новый аккаунт (то есть сгенерировать owner-кошелёк) или ввести backup phrase из 24 слов, восстановив доступ к старому owner-кошельку.
8. Включить в настройках Blockstack Portal доступ к протоколу аутентификации.

Прежде, чем аутентифицироваться в любом приложении, пользователь должен приобрести себе имя. Сделать это можно одним из следующих способов:

1. Завести биткоины на свой payment-кошелёк, через интерфейс Blockstack Portal заказать себе имя.
2. Приобрести имя иным способом (через сайт onename.com или через консольный клиент Blockstack Core) и передать имя на owner-кошелёк в Blockstack Portal.

На данный момент процесс установки клиента и приобретения имени в системе Blockstack достаточно неудобен для конечного пользователя. В будущем разработчики планируют значительно его доработать.

После приобретения имени процесс аутентификации проходит для пользователя тривиальным образом:

1. С запущенным клиентом он заходит на сайт приложения и пытается аутентифицироваться (Рис. 4).
2. Приложение генерирует запрос на аутентификацию и перенаправляет пользователя в Blockstack Portal (Рис. 5).

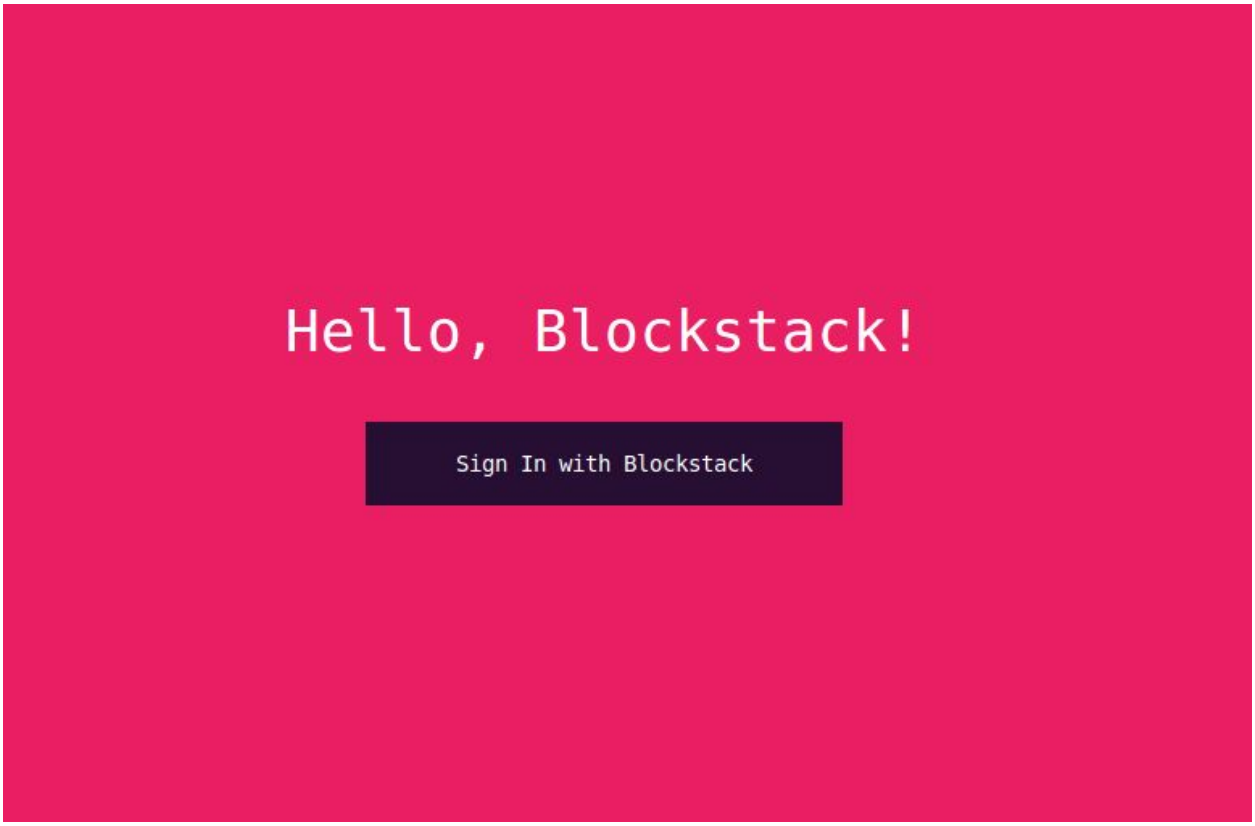

Рис. 4: домашняя страница тестового веб-сайта “Hello, Blockstack”, пользователь ещё не аутентифицирован.

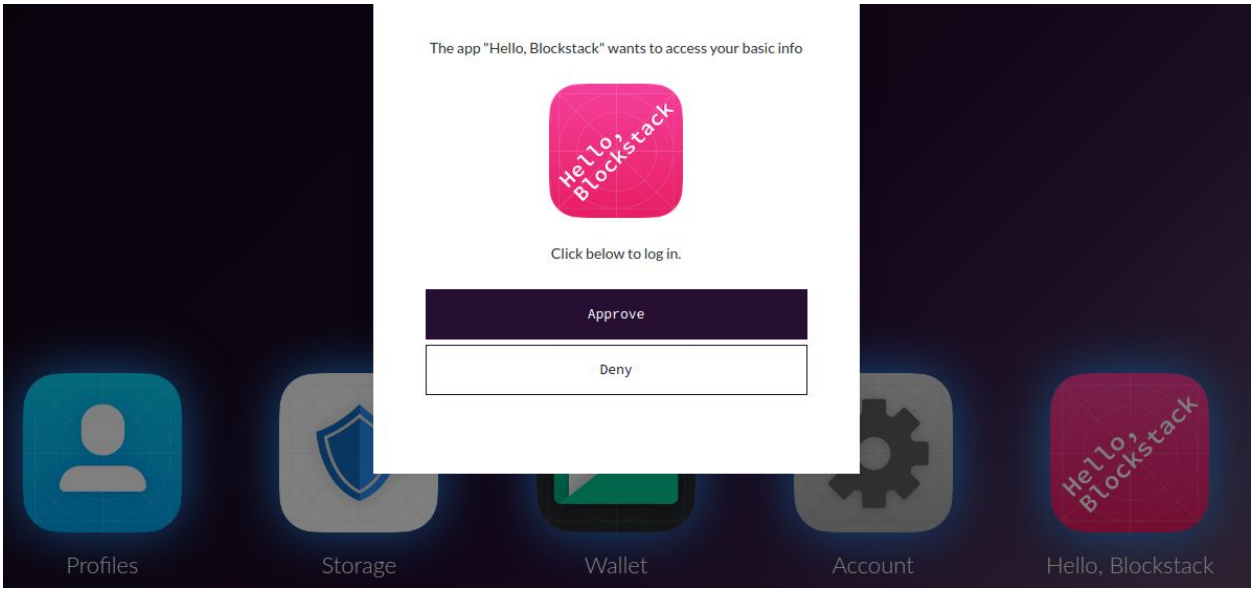

Рис. 5: интерфейс Blockstack Portal при входящем запросе на аутентификацию.

3. Пользователь кликом подтверждает запрос, Blockstack Portal генерирует ответ на запрос аутентификации и перенаправляет пользователя обратно в приложение. Приложение аутентифицирует пользователя (Рис. 6).

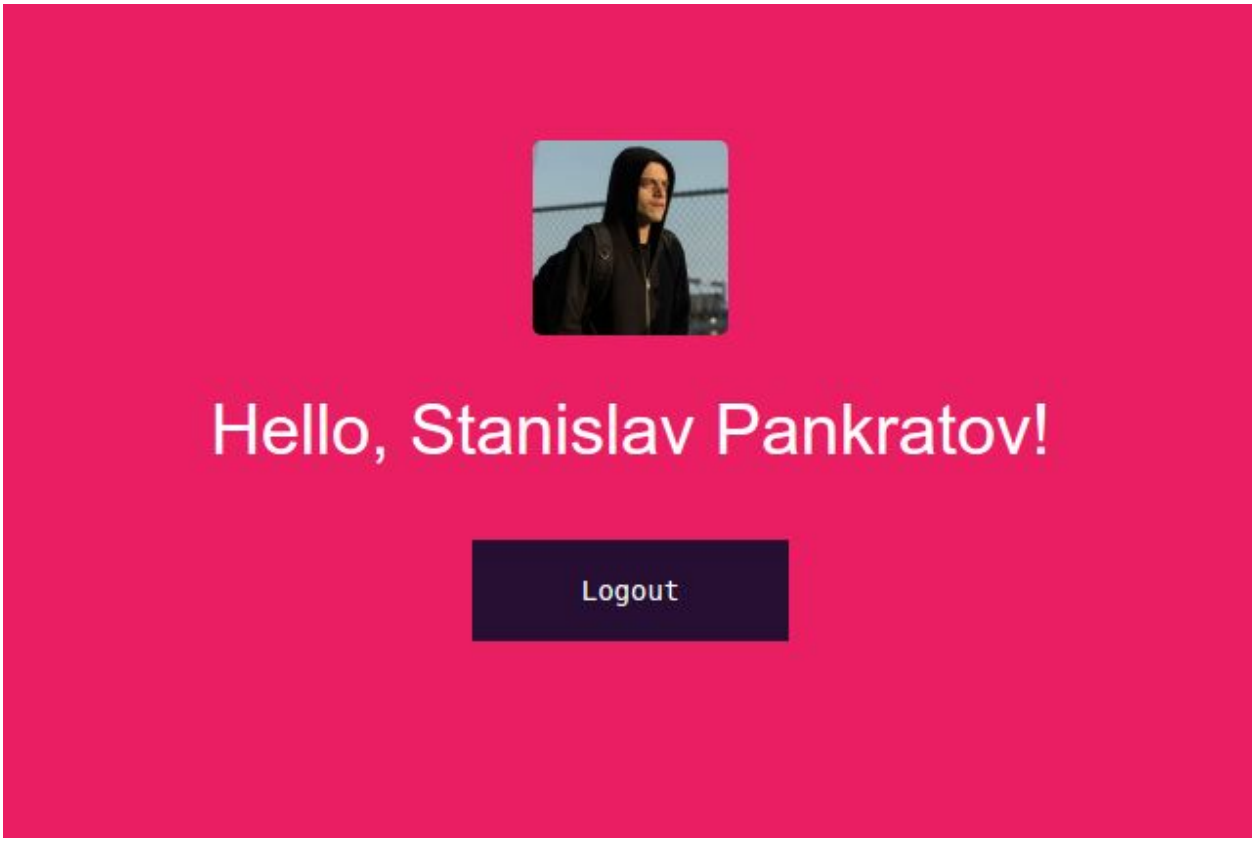

Рис. 6: домашняя страница тестового веб-сайта “Hello, Blockstack”, пользователь аутентифицирован.

3.3.2 Со стороны приложения

Чтобы встроить на свой сайт аутентификацию через Blockstack, разработчику необходимо подключить специальную библиотеку. Как только пользователь заходит на страницу аутентификации и пытается залогиниться, происходит следующее:

1. Библиотека на основе текущего времени и данных о самом приложении генерирует запрос на аутентификацию.
2. Запрос на аутентификацию подписывается закрытым ключом приложения и кодируется в виде одной строки (JSON Web Token).
3. Приложение перенаправляет пользователя в Blockstack Portal по специальной ссылке, содержащей ранее сгенерированную строку.
4. Blockstack Portal декодирует строку, верифицирует запрос и получает данные о приложении.
5. Как только пользователь подтверждает аутентификацию, Blockstack Portal перенаправляет его по адресу, указанному самим приложением в запросе на аутентификацию, указав в качестве GET-запроса закодированный ответ на запрос аутентификации.
6. Приложение верифицирует ответ на запрос и в случае успеха аутентифицирует пользователя и получает данные его профиля.

3.4 blockchainauth

3.4.1 Описание

Поскольку Blockstack разрабатывался как сеть для бессерверных приложений, основная библиотека для генерации запросов на аутентификацию и ответов на запрос аутентификации — библиотека на Javascript, код которой исполняется на стороне клиента: https://github.com/blockstack/blockstack.js

Однако в применении к традиционным серверным приложениям клиентская библиотека не подходит по соображениям безопасности. Добавив на страницу свой скрипт, злоумышленник может добиться неправильной работы приложения, подменив всю логику аутентификации. Таким образом можно, например, аутентифицироваться под чужим именем и получить доступ к чужим данным. Чтобы такого не случилось, была разработана библиотека на языке Python, исполняющаяся на уровне сервера, отвечающая за аутентификацию пользователей, blockchainauth:

https://github.com/spankratov/blockstack-auth-python

По окончанию разработки изменения, сделанные в рамках данной работы, были приняты разработчиками Blockstack в репозиторий проекта:

https://github.com/blockstack/blockstack-auth-python

3.4.2 Запрос на аутентификацию

Запросы на аутентификацию в рамках библиотеки представлены в виде объектов AuthRequest. AuthRequest хранит следующую информацию:

1. Закрытый ключ приложения (private_key).
2. Доменное имя приложения (domain_name).

3. Ссылка, по которой можно получить файл с информацией о приложении (manifest_uri). Значение по умолчанию: доменное имя + “/manifest.json”. По ссылке должен возвращаться JSON в следующем формате:

```
{
  "name": "Hello, Blockstack",
  "start_url": "localhost:5000",
  "description": "A simple demo of Blockstack Auth",
  "icons": [{
    "src":
"https://helloblockstack.com/icon-192x192.png",
    "sizes": "192x192",
    "type": "image/png"
  }]
}
```

4. Ссылка, на которую клиентское приложение отправляет ответы на запросы аутентификации (redirect_uri). Значение по умолчанию — доменное имя.
5. Список разрешений (scopes) — временно неиспользуемая часть протокола. Значение по умолчанию — пустой массив.
6. Unix-время, через которое истекает срок действия этого запроса (expires_at). Значение по умолчанию: через час.

Генерируемый на основе этих данных запрос имеет следующий JSON-формат:

```
{
    "header": {"typ": "JWT", "alg": "ES256K"},
    "payload": {
    "domain_name": "http://localhost:5000",
    "exp": 1493412486,
```

```
    "iat": 1493408886,
    "iss":
"did:btc-addr:1NZNxhoxobqwsNvTb16pdeiqvFvce3Yg8U",
    "jti": "75719c8a-3679-45b7-9551-21b6dfc28444",
    "manifest_uri":
"http://localhost:5000/manifest.json",
    "public_keys":
["027d28f9951ce46538951e3697c62588a87f1f1f295de4a14fdd4
c780fc52cfe69"],
    "redirect_uri": "http://localhost:5000",
    "scopes": []
    },
    "signature":
"HBwhcgPj7hrKg_IOGyaMJ9L-U_kwE5EweK8H54E2fuNONeWElfJg-h
10LJvbwrvf_3TcgzQRbqdxGSmro8Ey6A"
}
```


Формат запроса подобран специально для того, чтобы закодировать его в виде JSON Web Token [21].

1. Первая часть запроса, header, содержит информацию о том, что это JWT, а также алгоритм подписи (ES256K — синоним для SECP256k1, основная эллиптическая кривая, используемая в Bitcoin).
2. Вторая часть, payload, содержит всю основную информацию о запросе:
    a. domain_name — доменное имя.
    b. exp — Unix-время, в которое истекает срок действия запроса.
    c. iat — Unix-время, в которое был создан запрос.
    d. iss — строка в формате либо “did:btc-addr:<bitcoin address>”, либо “did:ecdsa-pub:<public key>”, которая обозначает способ адресации к приложению в блокчейне (адрес или открытый ключ).
    e. jti — уникальный идентификатор запроса (UUID).
    f. manifest_uri — ссылка на manifest.json приложения.

g. public_keys — список открытых ключей приложения (на данный момент поддерживается только один открытый ключ).
h. redirect_uri — ссылка, по которой приложение принимает ответы на запрос аутентификации.
i. scopes — список разрешений, на данный момент не используется.

3. Третья часть, signature, является криптографической подписью отправляемых данных. Подпись делается по алгоритму ECDSA с хэш-функцией SHA256.

Согласно формату JWT, все три эти части кодируются с помощью Base64 и записываются последовательно с точкой-разделителем. Пример использования библиотеки для генерации токена:

```
>>> from blockchainauth import AuthRequest
>>> private_key =
"a5c61c6ca7b3e7e55edee68566aeab22e4da26baa285c7bd10e8d2
218aa3b22901"
>>> auth_request = AuthRequest(private_key,
"localhost:5000")
>>> auth_request_token = auth_request.token()
>>> print auth_request_token
eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.eyJzY29wZXMiOltdLC
JyZWRpcmVjdF91cmkiOiJsb2NhbGhvc3Q6NTAwMCIsImp0aSI6IjI3M
WYxNGI5LTc3YmEtNDZiMy1iNmRjLTI2YzZjNjAyYWY5YiIsImV4cCI6
IjE0OTM1NTI0MDQuMDYiLCJwdWJsaWNfa2V5cyI6WyIwMjdkMjhmOTk
1MWNlNDY1Mzg5NTFlMzY5N2M2MjU4OGE4N2YxZjFmMjk1ZGU0YTE0Zm
RkNGM3ODBmYzUyY2ZlNjkiXSwiaXNzIjoiZGlkOmJ0Yy1hZGRyOjFOW
k54aG94b2Jxd3NOdlRiMTZwZGVpcXZGdmNlM1lnOFUiLCJpYXQiOiIx
NDkzNTQ4ODA3Ljk2IiwibWFuaWZlc3RfdXJpIjoibG9jYWxob3N0OjU
wMDAvbWFuaWZlc3QuanNvbiIsImRvbWFpbl9uYW1lIjoibG9jYWxob3
N0OjUwMDAifQ.EotiYm1yGOJt3qqUWKzB0FQqkk8onxEB_rBgIllIUZ
5l8gujREPPX1osWy8Wm6q8p0q81k41w1K8YyAAIgWucg
```

С помощью библиотеки запросы можно верифицировать. Запрос проходит верификацию, если:

1. Он содержит правильную подпись, сгенерированную на основе данных из payload открытым ключом из списка public_keys.
2. Адрес поля iss — производный открытого ключа из public_keys.
3. Время iat уже прошло.
4. Время exp ещё не настало.

```
>>> AuthRequest.verify(auth_request_token)
True
```

3.4.3 Ответ на запрос аутентификации

Запросы на аутентификацию в рамках библиотеки представлены в виде объектов AuthResponse. AuthResponse хранит следующую информацию:

1. Закрытый ключ приложения (private_key).
2. Профиль пользователя в формате JSON (profile).
3. Имя, по которому пользователь аутентифицируется (username).
4. Unix-время, через которое истекает срок действия этого запроса (expires_at). Значение по умолчанию: через месяц.

Генерируемый на основе этих данных ответ на запрос имеет следующий JSON-формат, во многом схожий с форматом AuthRequest:

```
{
    "header": {"typ": "JWT", "alg": "ES256K"},
    "payload": {
    "exp": 1496141298.31,
    "iat": 1493549308.76,
```

```
    "iss":
"did:btc-addr:1NZNxhoxobqwsNvTb16pdeiqvFvce3Yg8U",
    "jti": "b483adf1-1c3d-435c-be5e-bb88000bb2b4",
    "profile": {...},
    "public_keys":
["027d28f9951ce46538951e3697c62588a87f1f1f295de4a14fdd4
c780fc52cfe69"],
    "username": "stanislav_pankratov.id"
    },
    "signature":
"CoOAxoYEOAYd0ajaa31AWxFzvBhSxzdEUa-tm2LDlkICj_sNIgF-jH
JY-6WPGfk4OmLLZ-8Uos5HdWes1PviRQ"
}
```

Работа с объектами AuthResponse также схожа с объектами AuthRequest:

```
>>> from blockchainauth import AuthResponse
>>> STANISLAV_PANKRATOV_PROFILE = {...}
>>> private_key =
"a5c61c6ca7b3e7e55edee68566aeab22e4da26baa285c7bd10e8d2
218aa3b22901"
>>> username = "stanislav_pankratov.id"
>>> from blockchainauth import AuthResponse
>>> auth_response = AuthResponse(private_key,
STANISLAV_PANKRATOV_PROFILE, username)
>>> auth_response_token = auth_response.token()
>>> print auth_response_token
eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.eyJw...
```

С помощью библиотеки ответы на запросы можно верифицировать. Ответ проходит верификацию, если выполняются все те же условия, что и для самих запросов, а также если ответ проходит дополнительную верификацию: адрес из поля iss должен совпадать с публичным адресом владельца этого имени. Для проверки владельца имени используется либо локально запущенный Blockstack Core API, либо публичный, запущенный авторами Blockstack.

```
>>> AuthResponse.verify(auth_request_token)
True
```

3.5 django-blockstack

3.5.1 Описание

Чтобы разработчику не приходилось вручную проделывать на своём веб-сервере все описанные шаги, в рамках этой работы было разработано Django-приложение для лёгкой интеграции:

https://github.com/spankratov/django-blockstack

Django — самый популярный веб-фреймворк общего назначения на Python и пятый по популярности веб-фреймворк в целом [22]. Он имеет встроенную систему аутентификации и модель пользователя, отображающуюся в базу данных. Django требует минимальной настройки, чтобы установить стандартную систему хранения паролей в зашифрованном виде на стороне сервера. Но Django поддерживает множество других систем аутентификации, включая OAuth и OpenID. Для подключения дополнительной функциональности используются Django-приложения — библиотеки с интеграцией с Django.

3.5.2 Части приложения

django-blockstack зависит от следующих пакетов: blockchainauth (то есть разработанной в рамках этой работы Python-библиотеки), самого django и blockstack-profiles (библиотека с открытым исходным кодом для работы с Blockstack-профилями).

Одна из главных составляющих приложения — BlockstackAuthBackend. В терминах Django это Authentication Backend. Каждый раз, когда пользователь пытается аутентифицироваться и предоставляет некоторую информацию о себе (например, логин и пароль), Django вызывает метод authenticate у всех перечисленных в настройках Authentication Backend объектов. Пользователь аутентифицируется, если хоть один из них возвращает объект User.

```
class BlockstackAuthBackend(object):
  def authenticate(self, request,
auth_response_token=None):
    ...
```

Поскольку в протоколе Blockstack пользователь аутентифицируется по токену, сгенерированному его клиентом, метод authenticate объекта BlockstackAuthBackend принимает аргумент auth_response_token вместо традиционных username и password.

С помощью ранее созданной библиотеки blockchainauth токен верифицируется, в случае успеха пользователь ищется в базе или создаётся по Blockstack-имени в качестве username. По имени ищется профиль, из которого в базу записывается информация об имени пользователя.

Специально для поиска профиля есть функция fetch_profile. Сперва она пытается получить профиль с локально запущенного Blockstack Core API. Если сервер API не запущен или профиль по иным причинам не удалось получить, функция пытается получить его с удалённого Blockstack Core API.

django-blockstack добавляет три ссылки на сайт. Одна из них отвечает за генерацию объекта AuthRequest и перенаправление пользователя к его клиенту. В качестве redirect_uri для AuthRequest указывается вторая ссылка, которая

принимает токен AuthResponse и аутентифицирует пользователя. По третьей ссылке Blockstack Portal может получить файл manifest.json сайта.

За логику, стоящую за этими ссылками, отвечают функции views в терминах Django:

```
# django-blockstack/views.py
def blockstack_request(request):
  ...


def blockstack_response(request):
  ...


def manifest(request):
  ...
```

И, наконец, функции views ассоциируются с конкретными URL:

```
# django-blockstack/urls.py
urlpatterns = [
  url(r'^blockstack/request', views.blockstack_request,
        name='request'),
  url(r'^blockstack/response',
views.blockstack_response,
        name='response'),
  url(r'^blockstack/manifest.json', views.manifest,
        name='manifest')
]
```

3.5.3 Установка

Как и любые пакеты Python, django-blockstack устанавливается одной командой через pip:

```
pip install django-blockstack
```

Дальнейшие шаги описаны в репозитории на Github. Чтобы интегрировать аутентификацию Blockstack на своём сайте, разработчику нужно:

1. Добавить “django_blockstack” в список установленных приложений:

```
INSTALLED_APPS = [
    ...
    'django_blockstack',
]
```

2. Добавить “django_blockstack.backends.BlockstackAuthBackend” в настройку AUTHENTICATION_BACKENDS:

```
AUTHENTICATION_BACKENDS = [
    'django_blockstack.backends.BlockstackAuthBackend',
    ...
]
```

3. Конфигурировать приложение через настройку BLOCKSTACK_AUTH. Все данные settings используются для инициализации запроса на аутентификацию, это поле должно обязательно включать значение domain_name. Поле manifest содержит обязательную информацию о сайте. blockstack_api — опциональная настройка, используется, если на сервере запущен Blockstack Core в режиме API на нестандартном порте.

```
BLOCKSTACK_AUTH = {
    'settings': {
        'private_key':
'a5c61c6ca7b3e7e55edee68566aeab22e4da26baa285c7bd10e8d2
218aa3b22901',
        'domain_name': 'http://localhost:8000'
    },
    'manifest': {
        'name': 'Blockstack Auth Test',
```

```
        'start_url': 'http://localhost:8000',
        'description': 'A simle demo of Blockstack
Auth',
        'icons': [{
                'src':
'http://localhost:8000/static/icon.png',
                'sizes': '420x420',
                'type': 'image/png'
        }]
    },
    'blockstack_api': 'http://localhost:6270'
}
```

4. Подключить URL Blockstack в главном файле проекта urls.py:

```
url(r'', include('django_blockstack.urls')),
```

5. При попытке пользователя аутентифицироваться через Blockstack сайт просто перенаправляет его на ссылку /blockstack/request:

```
def login(request):
    …
    return redirect('blockstack:request')
```

6. Когда сайту необходимо получить доступ к профилю пользователя, он вызывает метод fetch_profile:

```
>>> from blockstack_auth.profile import fetch_profile
>>> fetch_profile(user.username)
```

3.6 Сайт

Для тестирования протокола был написан простой сайт, который отображает основную информацию из профиля пользователя Blockstack: имя,

аватар, фоновое изображение и ссылки на подтверждённые аккаунты в Github, Facebook и Twitter. Сайт написан на Django и использует приложение django-blockstack: https://github.com/spankratov/django-blockstack-test

При первом посещении сайт предлагает пользователю аутентифицироваться (Рис. 7).

При нажатии на кнопку пользователя перенаправляют на Blockstack Portal, где отображается основная информация о приложении (имя, иконка) (Рис. 8).

Как только пользователь нажимает кнопку “Approve”, его перенаправляют обратно на сайт, где он аутентифицируется. Видна основная информация из его профиля (Рис. 9).

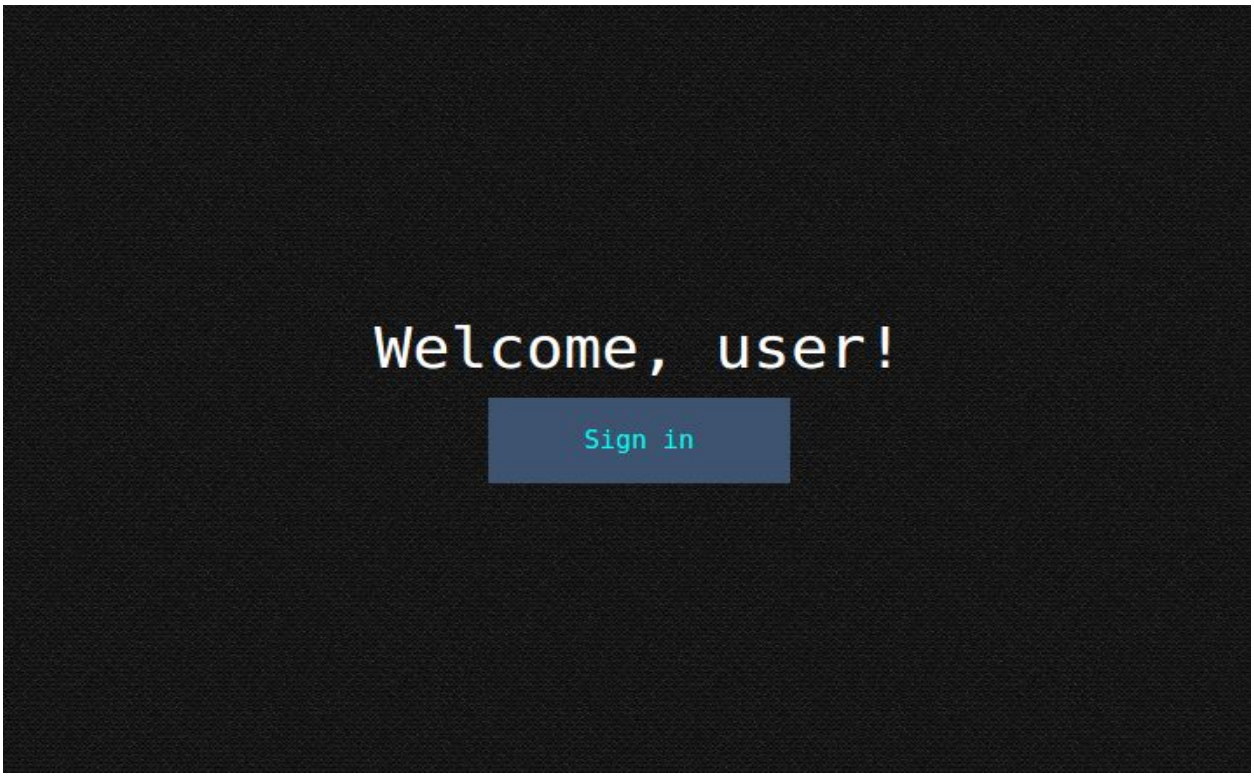

Рис. 7: домашняя страница тестового сайта. Пользователь не аутентифицирован.

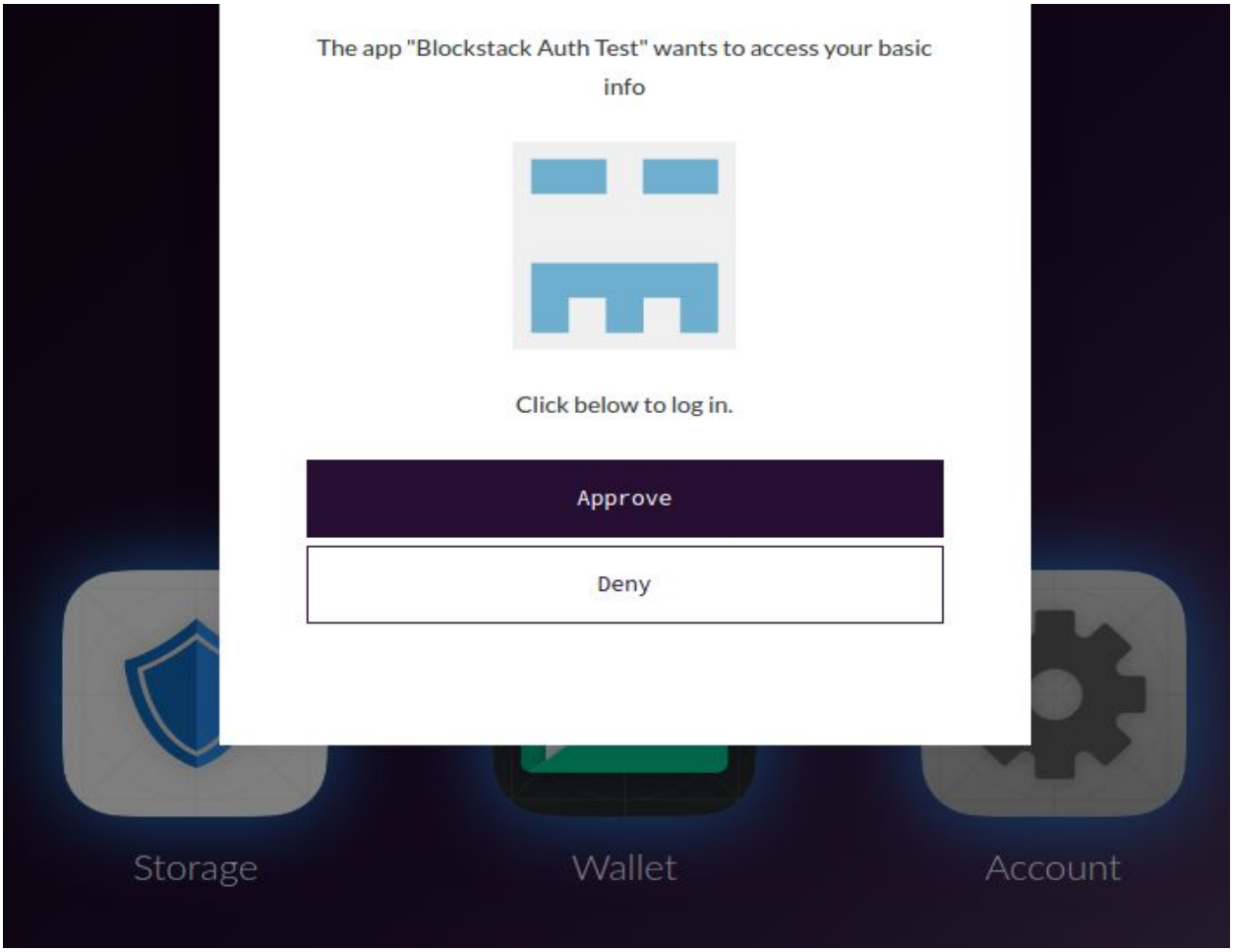

Рис. 8: интерфейс Blockstack Portal, запрос на аутентификацию от тестового сайта.

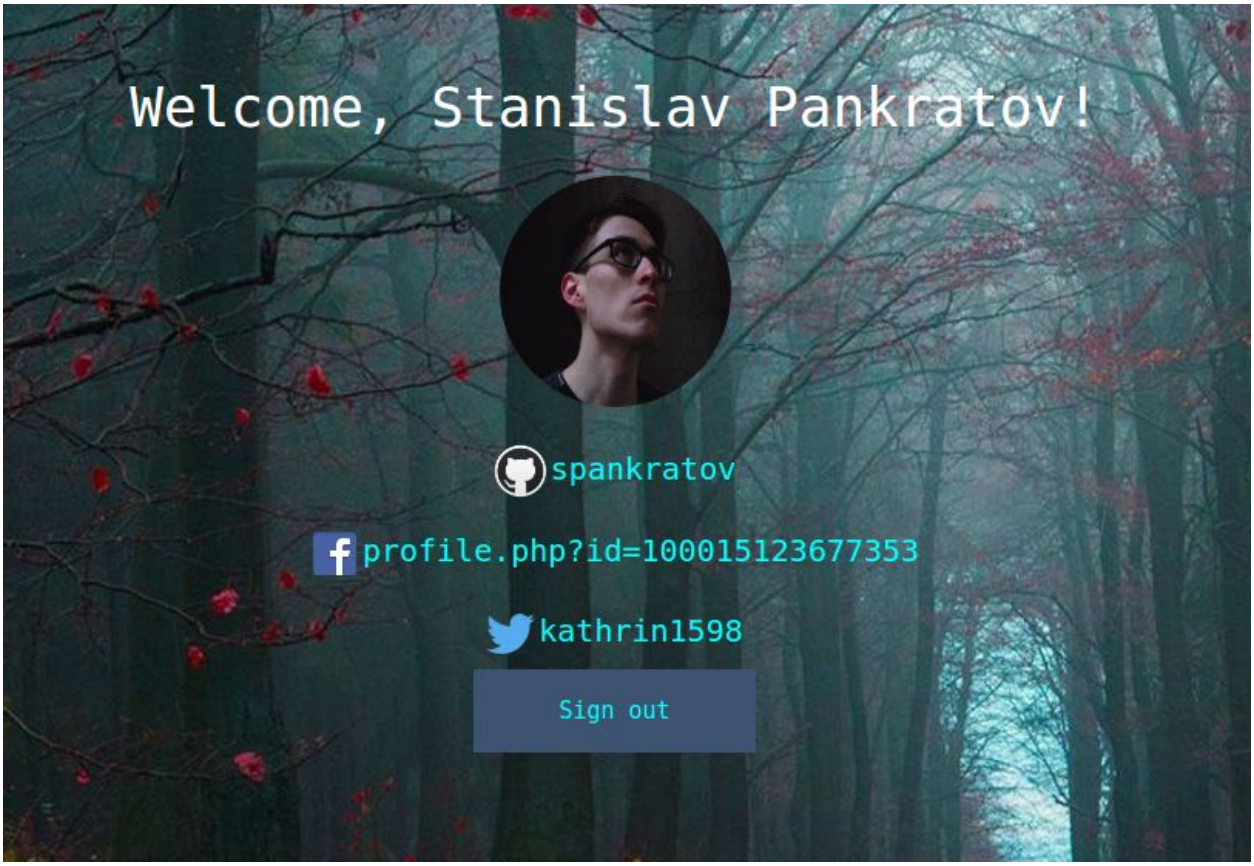

Рис. 9: домашняя страница тестового сайта, пользователь аутентифицирован.

Показывается основная информация о пользователе из его профиля.

Глава 4. Анализ решения

4.1 Введение

В этой главе будут подробно рассмотрены основные проблемы, которые решает протокол Blockstack. Будут описаны достоинства для пользователей в отношении удобства хранения паролей. Будут изучены векторы атаки на централизованные системы хранения пользовательских аккаунтов и почему подобные атаки невозможны при использовании протокола Blockstack. Также необходимо затронуть вопрос нарушений со стороны централизованного провайдера имён. В конце будут приведены недостатки, которые существуют в протоколе на данный момент, и как данная работа помогла их частично преодолеть.

4.2 Универсальность

Одним из неоспоримых преимуществ протокола является единый вход от одного и того же имени на множество сервисов. В мире с огромным количеством веб-сервисов это избавляет от необходимости создавать множество разных учётных записей для этих сервисов, каждый раз заново заполнять всю необходимую информацию, хранить пароли отдельно для каждого. Имя в системе Blockstack может использоваться на всех сайтах, которые подключили этот протокол, что можно легко сделать с помощью библиотек, разработанных в рамках этой работы. Таким образом, вкладом данной работы является упрощение применения протокола Blockstack как протокола единого входа.

Однако технологии единого входа не новы. Аккаунты в таких сервисах, как Google или Facebook используются для аутентификации на множестве

вебсайтов, и они так же удобны для пользователей тем, что позволяют не создавать отдельные аккаунты для каждого сайта.

В следующих разделах сравнение протокола Blockstack будет производиться только с сервисами, предоставляющими механизмы Single Sign On.

4.3 Безопасность

Самой большой проблемой, связанной с централизованными хранилищами учётных записей, является безопасность. Поскольку аккаунты пользователей хранятся не у самих пользователей, они не могут контролировать меры безопасности, предпринимаемые компаниями-поставщиками имён. Серверы компаний могут быть взломаны, и через них злоумышленники могут получить доступ к аккаунтам пользователей.

Взлом учётной записи на одном из сервисов не всегда означает компрометацию всех учётных записей пользователя, но на практике пользователи часто используют один и тот же пароль для нескольких сервисов. Если хакеры узнают пароль от аккаунта на каком-нибудь малозначимом сервисе, а тот же самый пароль используется, например, для электронной почты, то злоумышленник получит доступ сразу и к почте, и ко всем привязанным к ней аккаунтам.

Особенно опасны массовые взломы поставщиков имён, предоставляющими различными протоколы Single Sign On (OAuth, OpenID), или взломы провайдеров электронной почты, так как к электронной почте очень часто привязываются другие аккаунты. Это создаёт принципиально очень опасную ситуацию, при которой крупные поставщики имён (Google, Facebook, Twitter, Microsoft, Yahoo, Yandex, VK) являются гигантской точкой отказа всей

сети: взлом одного такого провайдера вызывает утечку колоссального количества аккаунтов, что в свою очередь влечёт компрометацию аккаунтов на других сервисах, которые используют аутентификацию через взломанного провайдера.

Чтобы оценить масштабы такого взлома, необходимо оценить долю аккаунтов всемирной паутины, в том или ином виде привязанных к поставщикам личностей. И хотя полноценной статистики по этим вопросам нет, можно всё равно проследить общие тенденции по косвенным данным.

Рассмотрим отчёт 2016 года, проведённый компанией-разработчиком решений аутентификации LoginRadius [28]. Согласно анализу 160000 веб-сайтов с функцией Social Login (вход через социальные сети), 93% пользователей этих сайтов предпочитали вход через социальные сети обычной регистрации по электронной почте (Рис. 10).

При этом подавляющее большинство пользователей, выбравших Social Login, являются пользователями двух крупных поставщиков имён: Facebook и Google (Рис. 11).

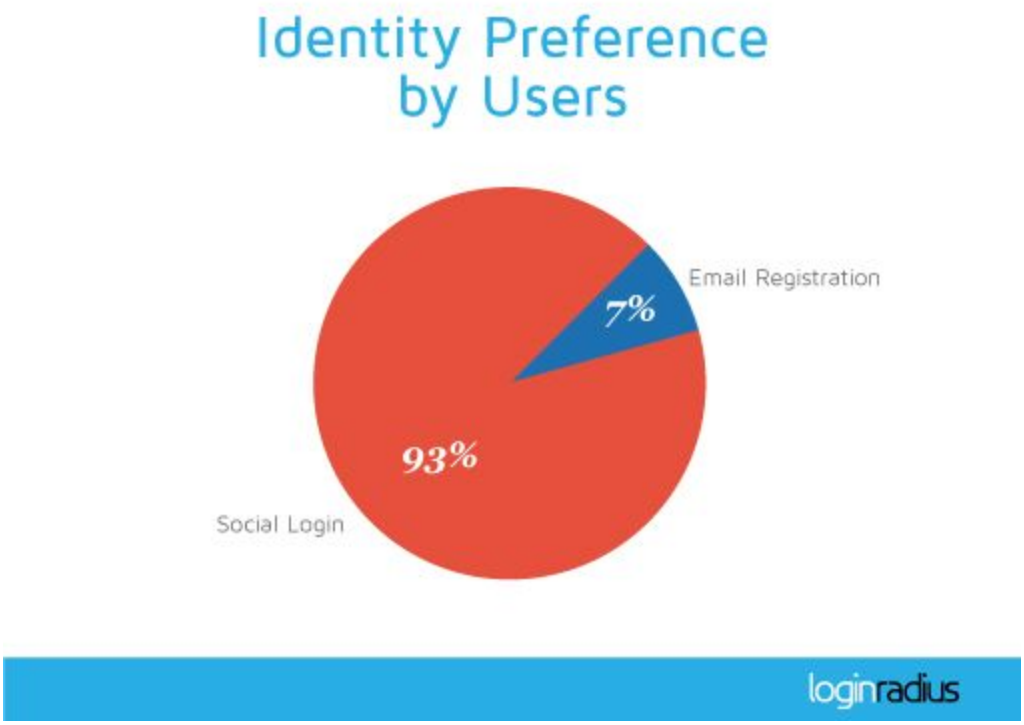

Рис. 10: предпочтения пользователей в методах аутентификации. Вход через социальные сети против стандартной регистрации через почту.

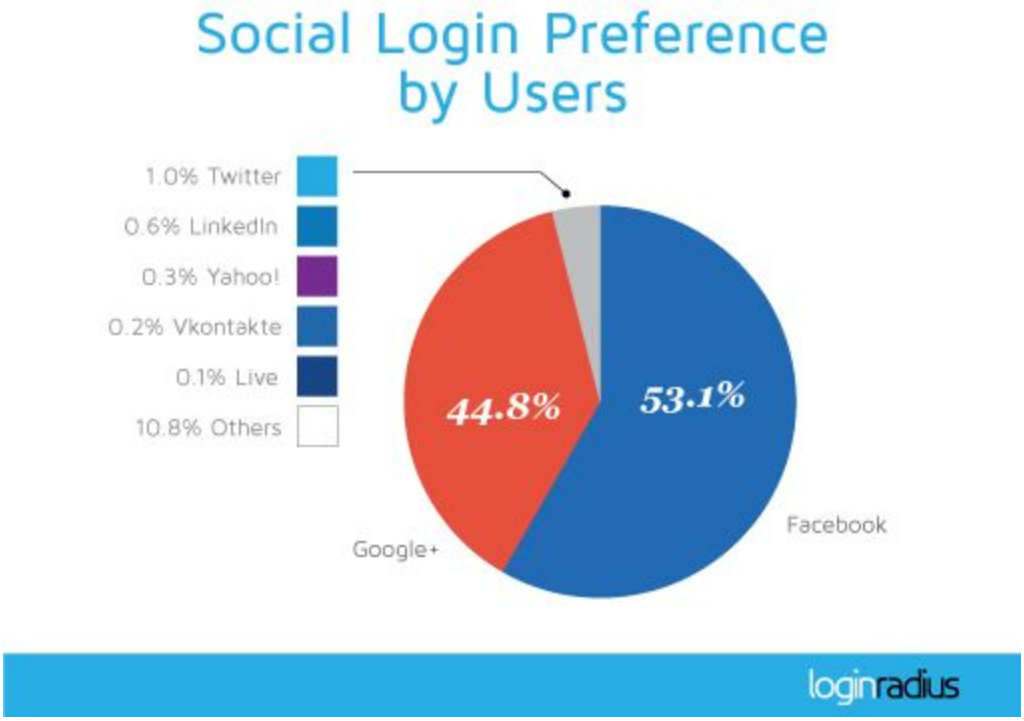

Рис. 11: распределение пользователей по сервисам Social Login.

Из этого можно сделать вывод, что все пользователи, воспользовавшиеся возможностью аутентифицироваться через социальные сети, находятся под угрозой кражи их аккаунта в случае взлома их поставщика личности, причём почти 98% пользователей зависят всего от двух сервисов, которые являются крупными точками отказа. Все пользователи этих двух сервисов (почти 2 миллиарда активных пользователей Facebook [29] и более миллиарда активных пользователей Google [30]) могут воспользоваться услугами единого входа и попасть в зависимость от этих поставщиков имён.

Владельцы множества сайтов позволяют своим пользователям аутентифицироваться через аккаунты крупных поставщиков имён. Так, через Facebook можно аутентифицироваться примерно на 14,5 миллионах сайтов [31].

Не все веб-сайты, однако, предоставляют возможность аутентифицироваться через поставщиков имён. Тем не менее, если веб-сайт не предоставляет этой возможности, то практически всегда в целях защиты от роботов и от забывчивости пользователя он требует при регистрации электронную почту, через которую пользователь впоследствии может восстановить доступ к аккаунту. Взлом поставщика электронной почты может повлечь кражу всех аккаунтов, привязанных к почте пользователей.

Как и в случае с социальными сетями, большинство аккаунтов привязаны к нескольким крупным поставщикам услуг. Согласно отчёту MailChimp, крупного сервиса рассылки электронной почты, исследовавшего 81,69 миллиарда писем, отправленных с их серверов, большинство пользователей пользуются почтой от пяти больших компаний: Google, Hotmail, Yahoo, AOL и Comcast [32].

Хотя и не располагая какими-то точными оценками из-за отсутствия нужной статистики, можно сделать вывод, что с точки зрения безопасности

пользователи-владельцы аккаунтов и имён находятся в слишком большой зависимости от нескольких крупных поставщиков услуг и подвержены риску компрометации своих аккаунтов при компрометации этих сервисов.

Можно было бы сказать, что крупные компании нанимают достаточно квалифицированный персонал и защищают данные своих пользователей настолько надёжно, насколько это возможно. Однако от ошибок не застрахован никто, и как показывает история, перед взломами уязвимы даже гигантские корпорации. Несколько самых значимых примеров:

- Взлом серверов Sony в 2011 году. Пароли хранились в нехэшированном виде, а доступ к ним был получен обычной SQL-инъекцией [23].
- Шесть с половиной миллионов паролей пользователей LinkedIn были украдены в 2012 году. Пароли были захэшированы, но без криптографической соли. Это позволило хакерам легко получить пароли из украденных хэшей [24].
- Около миллиарда аккаунтов пользователей Yahoo в 2013 году были скомпрометированы через поддельные cookie-файлы [25].
- Фишинговая атака на электронные ящики Google в 2017 году, были взломаны около миллиона аккаунтов [26]. Аккаунты были взломаны из-за уязвимости в протоколе OAuth, о которой исследователи предупреждали ещё в 2011 году [27].

Опасность может крыться и не на стороне самой компании, а где-то среди используемых ею технологиях. Появление таких уязвимостей сложно контролировать и невозможно предугадать. Так, одна из самых известных уязвимостей последних лет, Heartbleed, была найдена в OpenSSL — программном обеспечении, лежащем в основе современного Интернета. Оказались уязвимы серверы таких компаний, как Amazon, GitHub, Pinterest,

Reddit, Tumblr, Wikimedia, Yahoo. Уязвимость присутствовала в коде OpenSSL на протяжении двух лет [33].

На примере Heartbleed можно увидеть, как легко злоумышленнику, зная конкретную уязвимость, получить доступ к аккаунтам пользователей. Для анализа веб-сайта достаточно свободно распространяемых программ [34]. Вот как выглядит анализ содержимого памяти сервера с помощью программы Metasploit Framework:

```
root@kali:~# msfconsole
msf > use auxiliary/scanner/ssl/openssl_heartbleed
msf auxiliary(openssl_heartbleed) > set RHOSTS
192.168.154.137
RHOSTS => 192.168.154.133
msf auxiliary(openssl_heartbleed) > run
[*] 192.168.154.137:443 - Sending Client Hello...
[*] 192.168.154.137:443 - Sending Heartbeat...
[*] 192.168.154.137:443 - Heartbeat response, checking
if there is data leaked...
[+] 192.168.154.137:443 - Heartbeat response with leak
[*] 192.168.154.137:443 - Printable info leaked:
@SKO0'94wiW*G):[f"!98532ED/Ait/537.36 (KHTML, like
Gecko)
Chrome/34.0.1847.116 Safari/537.36Accept-Encoding:
gzip,deflate,sdchAccept-Language: pt-BR,pt;q=0.8,en-
US;q=0.6,en;q=0.4Cookie: nessus-session=false]5gOZ
pt-BR,pt;q=0.8,en-
US;q=0.6,en;q=0.4Cookie: nessus-session=falseql|Z4EKT
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

По оценкам исследователей, совершив всего 3 таких запроса в минуту, злоумышленник может получить 30% всей памяти, читаемой в данный момент сервером [35]. Если злоумышленник ищет в памяти пары “логин-пароль”, то

довольно скоро он наткнётся на данные такого вида, пришедшие от пользователя веб-приложения в виде POST-запроса на аутентификацию:

```
[*] 192.168.154.137:443 - Printable info leaked:
…
login_username=user123&password=jh345sfhds32
…
```

Используя тестовый фреймворк RUBiS, имитирующий сайт интернет-аукциона, исследователи смогли при всего 100 активных пользователях за 10 минут получить около 7000 строк с парами “логин-пароль”, эксплуатируя уязвимость Heartbleed [36].

Конечно, разработанная в рамках данной работы система не решает глобальный вопрос уязвимостей в программной обеспечении. Даже при использовании протокола Blockstack злоумышленники могут предпринять следующие атаки:

1. Атака на компьютер пользователя. Хотя приватные ключи и зашифрованы локально мастер-паролем пользователя, в зависимости от полученных прав доступа злоумышленник всё равно может украсть аккаунт пользователя (например, выудив у пользователя пароль кейлоггером или представившись на сайте под именем пользователя через существующую сессию в Blockstack Portal).
2. Атака на сервер приложения, использующего протокол аутентификации Blockstack. Злоумышленник может получить доступ ко всей пользовательский информации, относящейся к этому приложению (например, к личной переписке в социальной сети).

Однако при использовании протокола Blockstack принципиально невозможен тот сценарий развития событий, когда взлом одного сервиса влечёт за собой кражу аккаунтов на другом. Взлом одной стороны аутентификационного процесса выливается либо в компрометацию одного аккаунта (в случае взлома локального компьютера пользователя), либо в компрометацию многих аккаунтов, связанных только лишь с этой стороной (в случае взлома сервера приложения). Поставщик имён в данном случае заменяется блокчейном, и взлом хранилища имён равноценен компрометации гигантского блокчейна Bitcoin и контролю более 51% узлов сети.

В рамках данной работы в качестве протокола аутентификации сознательно был выбран протокол Blockstack из-за своих сильных характеристик безопасности. При активном использовании этого протокола в сети может исчезнуть проблема крупных точек отказа в виде поставщиков личностей.

4.4 Децентрализованность

При классическом сценарии хранения имён пользователей, когда хранение осуществляется на какой-то одной стороне (компании-поставщике имён), фактически имя контролируется не пользователем, а самой компанией. Компания-поставщик по своему усмотрению может:

1. Полностью удалить имя пользователя.
2. Изменить его данные.
3. Просматривать его данные и передавать третьим сторонам. Пример: PRISM.
4. Использовать данные пользователя в коммерческих целях Пример: таргетированная реклама.

Потому и нельзя говорить, что пользователь в полной мере владеет именем. Именем владеет компания, которая даёт его пользователю в бессрочную аренду на своих условиях.

Централизованное хранение имён пользователей может быть опасно ещё и тем, что компания-поставщик может попросту прекратить существование. В обозримом будущем это маловероятный исход для крупнейших компаний (Google, Facebook), но есть и множество более мелких и менее стабильных поставщиков имён, от которых зависят пользователи.

Протокол Blockstack радикально решает проблему, используя блокчейн — децентрализованную базу данных. Ни один узел сети не может полностью контролировать систему, новые блоки добавляются в блокчейн на основе консенсуса множества узлов. Вышеперечисленные проблемы в блокчейне могут возникнуть только при нарушении работы более 51% узлов блокчейна Bitcoin. В мае 2017 года количество полных узлов сети Bitcoin — около 7000 [37]. Это значит, что для компрометации пользовательских имён злоумышленникам пришлось бы контролировать более 3500 полных узлов сети, что представляется практически невозможным.

4.5 Недостатки и возможности их преодоления

Как и любая технология, протокол Blockstack и разработанная вокруг него система имеет свои недостатки.

Одним из главных, непреодолимых недостатков является невысокая скорость работы системы в целом. Это проистекает из принципиальных особенностей протокола. Поскольку все имена хранятся в блокчейне Bitcoin, все операции так или иначе связанные с изменением этого хранилища (регистрация имени, обновление zone-файла, передача имени) проходят верификацию и

принимаются или отвергаются на основе консенсуса нескольких узлов сети. Такая транзакция может проходить от нескольких минут до пары часов.

Из-за своей новизны протокол находится в ещё очень сыром состоянии и не совсем удобен для обычных пользователей:

1. За каждую операцию, связанную с владением именем и изменением блокчейна пользователю приходится платить небольшую сумму в биткоинах. Это проистекает из того, что каждая такая операция — это транзакция Bitcoin, которую необходимо подписать закрытым ключом пользователя. А все транзакции Bitcoin подразумевают передачи хотя бы небольшой суммы.
2. Чтобы воспользоваться протоколом, пользователь вынужден ставить дополнительное программное обеспечение (Blockstack Core, Blockstack Portal), которое находится на ранней стадии разработки и часто даже не готово для удобной установки. Так, готовый установочный пакет существует только для операционной системы OS X. На данный момент пользователи Windows и Linux вынуждены вручную собирать и запускать все части ПО, хотя разработчики активно работают над исправлением этой проблемы.
3. Пока существует очень малое количество сайтов, где подключен протокол и где пользователь может попробовать аутентификацию через Blockstack.

В силу того, что протокол находится на ранней стадии развития, он пока не получил широкого применения. Тем не менее он крайне важен хотя бы как доказательство того, что такая важная часть жизни, как идентификация человека может управляться без контроля третьими сторонами. Кроме того, у протокола есть все шансы распространиться по всему миру благодаря

поддержке Microsoft, с которыми сотрудничают разработчики Blockstack, чтобы создать децентрализованную систему идентификации [3].

Главным вкладом данной работы является частичное преодоление неудобства, связанного с работой с протоколом. Так, у разработчиков традиционных серверных приложений не было возможности легко встроить аутентификацию пользователей на сервере. В инфраструктуре Blockstack есть библиотека на Javascript, blockstack.js, которая может генерировать запросы на аутентификацию и ответы на них, но эта библиотека предназначена в первую очередь для бессерверных приложений. Такой подход подразумевает прохождение аутентификации на стороне клиента, что может привести к злоупотреблениям (например, подмене кода, попытке выдать себя за других людей). Благодаря разработанным библиотекам все веб-сайты, использующие Python и Django, могут встроить аутентификацию пользователей через Blockstack, что способствует более широкому принятию протокола.

Разработчики Blockstack работают над преодолением перечисленных недостатков, и есть все основания полагать, что вскоре протокол аутентификации будет более удобен для широкого применения.

Выводы

В данной работе были проанализированы подходы к аутентификации, применяющиеся в современном мире. Были выявлены критические недостатки, такие как централизованность и зависимость процесса аутентификации от третьих сторон. Анализ крупных утечек информации и злоупотреблений со стороны поставщиков имён показал серьёзность этих недостатков.

В качестве отправной точки был выбран перспективный протокол аутентификации децентрализованной системы имён Blockstack. За счёт хранения имён в блокчейне он сам по себе решал приведённые выше недостатки, однако в силу своей новизны был слишком неудобен как для обычных пользователей, так и для разработчиков веб-приложений, которые могли бы встроить аутентификацию через Blockstack в своё приложение.

Вклад данной работы — упрощение установки системы аутентификации Blockstack для разработчиков веб-приложений.

Python-библиотека blockchainauth предназначена для удобной генерации запросов на аутентификацию и ответов на запрос аутентификации. Эта библиотека может использоваться как сама по себе, для ручного управления процессом аутентификации со стороны разработчика, так и в составе Django-приложения django-blockstack, позволяющему с помощью одной команды установки пакета и нескольких строк конфигурационных файлов встроить аутентификацию через Blockstack на веб-сайт.

Разработка тестового веб-сайта показала, что использование этих библиотек удобно и не вынуждает разработчика лишний раз задумываться о настройке аутентификации.

Заключение

В рамках данной работы были получены следующие результаты:

1. Проанализированы существующие протоколы аутентификации, их преимущества и недостатки. Был выделен класс наиболее перспективных протоколов на базе технологии блокчейн.
2. Был выбран для изучения один конкретный протокол из системы Blockstack, использующий ранее разработанную инфраструктуру: хранилище личностей в блокчейне, узлы сети, приложение-клиент. Были изучены его возможности, достоинства и недостатки, подробно рассмотрено его применение в аутентификации пользователей.
3. Были разработаны две библиотеки на языке Python: blockchainauth и django-blockstack. Обе библиотеки пригодны для лёгкой интеграции системы аутентификации Blockstack на веб-сайты.
4. Был разработан тестовый сайт, в полной мере использующий библиотеки blockchainauth и django-blockstack.
5. Система была подробно изучена, были рассмотрены проблемы существующих систем аутентификации и как она их преодолевает. Были изучены недостатки самого протокола Blockstack и как они частично решаются с помощью разработанных библиотек.

Сформулированные задачи были выполнены в полном объеме. Поставленные цели были достигнуты.

Приложение

Фрагменты кода blockchainauth

Класс AuthRequest

```
class AuthRequest(AuthMessage):
     """ Interface for creating signed auth request tokens, as
well as decoding
     and verifying them.
     """

     verify_methods = [
     is_expiration_date_valid,
     is_issuance_date_valid,
     do_signatures_match_public_keys,
     do_public_keys_match_issuer
     ]

     def __init__(self, private_key, domain_name,
manifest_uri=None, redirect_uri=None, scopes=None,
expires_at=None, crypto_backend=default_backend()):
     """ private_key should be provided in HEX, WIF or binary
format
          domain_name should be a valid domain
          manifest_uri should be a valid URI
          redirect_uri should be a valid URI
          scopes should be a list
          expires_at should be a float number of seconds since
the epoch
     """
     if not manifest_uri:
          manifest_uri = domain_name + '/manifest.json'

     if not redirect_uri:
          redirect_uri = domain_name

     if not scopes:
          scopes = []

     if not expires_at:
          expires_at = time.time() + 3600  # next hour
```

```
    self.private_key = private_key
    self.domain_name = domain_name
    self.manifest_uri = manifest_uri
    self.redirect_uri = redirect_uri
    self.scopes = scopes
    self.expires_at = expires_at
    self.tokenizer = Tokenizer(crypto_backend=crypto_backend)

    def _payload(self):
        now = time.time()
        payload = {
            'jti': str(uuid.uuid4()),
            'iat': str(now),
            'exp': str(now + self.expires_after),
            'iss': None,
            'public_keys': [],
            'domain_name': self.domain_name,
            'manifest_uri': self.manifest_uri,
            'redirect_uri': self.redirect_uri,
            'scopes': self.scopes
        }
        if self.private_key:
            public_key =
BitcoinPrivateKey(self.private_key).public_key()
            address = public_key.address()
            payload['public_keys'] = [public_key.to_hex()]
            payload['iss'] = make_did_from_address(address)
        return payload

    @classmethod
    def fetch_app_manifest(cls, token):
        # decode the token
        try:
            decoded_token = cls.decode(token)
        except DecodeError:
            return None

        try:
            return
requests.get(decoded_token['payload']['manifest_uri']).json()
        except (requests.exceptions.RequestException,
ValueError):
            # ValueError for non-json responses
            return None
```

```
    def redirect_url(self):
        return 'blockstack:' + self.token()
```

Класс AuthResponse

```
class AuthResponse(AuthMessage):
    """ Interface for creating signed auth response tokens, as
well as decoding
    and verifying them.
    """

    verify_methods = [
    is_expiration_date_valid,
    is_issuance_date_valid,
    do_signatures_match_public_keys,
    do_public_keys_match_issuer,
    do_public_keys_match_username
    ]

    def __init__(self, private_key, profile=None, username=None,
                 expires_at=None,
crypto_backend=default_backend()):
    """ private_key should be provided in HEX, WIF or binary
format
         profile should be a dictionary
         username should be a string
         expires_at should be a float number of seconds since
the epoch
    """
    if not private_key:
         raise ValueError('Private key is missing')

    if not profile:
         profile = {}

    if not expires_at:
         expires_at = time.time() + 30 * 24 * 3600  # next month

    self.private_key = private_key
    self.public_key =
BitcoinPrivateKey(self.private_key).public_key()
    self.address = self.public_key.address()
    self.profile = profile
    self.username = username
    self.expires_at = expires_at
```

```
    self.tokenizer = Tokenizer(crypto_backend=crypto_backend)

    def _payload(self):
        now = time.time()
        return {
            'jti': str(uuid.uuid4()),
            'iat': str(now),
            'exp': str(now + self.expires_after),
            'iss': make_did_from_address(self.address),
            'public_keys': [self.public_key.to_hex()],
            'profile': self.profile,
            'username': self.username
        }
```

Фрагменты кода django-blockstack

Authentication Backend

```
class BlockstackAuthBackend(object):
  def authenticate(self, request, auth_response_token=None):
    try:
      verified = AuthResponse.verify(auth_response_token)
    except (ValueError, DecodeError):
      return None
    if verified:
      username =
AuthResponse.decode(auth_response_token)['payload']['username']
      user, _ = User.objects.get_or_create(
           username=username
      )
      try:
        profile = fetch_profile(username)
      except:
        pass
      else:
        claim = profile.get('claim', {})
        name = claim.get('name', None)
        if name:
          name = name.split(' ', 1)
          user.first_name = name[0]
          if len(name) == 2:
            user.last_name = name[1]
```

```
            user.save()
        return user
```

Функция fetch_profile

```
def fetch_profile(name):
  profile = {}
  try:
    # try local blockstack api first
    blockstack_api_url = getattr(settings, 'BLOCKSTACK_AUTH',
{}).get('blockstack_api', 'http://localhost:6270')
      response = urllib2.urlopen(blockstack_api_url +
NAME_LOOKUP_URL + name)
      response = json.loads(response.read())
      if name in response and 'profile' in response[name]:
        profile = response[name]['profile']
  except urllib2.URLError:
    pass

  if not profile:
    try:
      # try external blockstack core api
      response = urllib2.urlopen(BLOCKSTACK_CORE_URL +
NAME_LOOKUP_URL + name)
      response = json.loads(response.read())
      if name in response and 'profile' in response[name]:
        profile = response[name]['profile']
    except urllib2.URLError:
      pass

  return profile
```

Views

```
# django-blockstack/views.py
def blockstack_request(request):
```

```
  blockstack_auth_settings = \
    settings.BLOCKSTACK_AUTH['settings']
  blockstack_auth_settings['private_key'] = \
    blockstack_auth_settings.get('private_key', None)
  blockstack_auth_settings['manifest_uri'] = \
    blockstack_auth_settings['domain_name'] + \
    reverse('blockstack:manifest')
  blockstack_auth_settings['redirect_uri'] = \
    blockstack_auth_settings['domain_name'] + \
    reverse('blockstack:response')
  ar = AuthRequest(**blockstack_auth_settings)
  if not AuthRequest.verify(ar.token()):
    return HttpResponse('Can\'t generate Blockstack
authentication request token', status=500)
  else:
    response = HttpResponse("", status=302)
    response['Location'] = ar.redirect_url()
    return response


def blockstack_response(request):
  auth_response_token = request.GET.get('authResponse')
  if not auth_response_token:
    return HttpResponseBadRequest('No authResponse parameter')
  user = authenticate(request,
                      auth_response_token=auth_response_token
  )
  if user is not None:
    login(request, user)
    response = HttpResponse("", status=302)
    response['Location'] = \
      '{scheme}://{host}'.format(scheme=request.scheme,
                                 host=request.get_host())
    return response


def manifest(request):
  result = JsonResponse(settings.BLOCKSTACK_AUTH['manifest'])
  result['Access-Control-Allow-Origin'] = '*'
  return result
```

Формат профиля пользователя


```
[
  {
     "decodedToken": {
     "header": {
     "alg": "ES256K",
     "typ": "JWT"
     },
     "payload": {
     "issuedAt": "2017-04-20T20:27:23.518036",
     "claim": {
          "account": [
          {
               "proofType": "http",
               "identifier": "profile.php?id=100015123677353",
               "proofUrl":
"https://www.facebook.com/permalink.php?story_fbid=191989997981788&id=100015123677353",
               "service": "facebook",
               "@type": "Account"
          },
          {
               "proofType": "http",
               "identifier": "spankratov",
               "proofUrl":
"https://gist.github.com/spankratov/4e1730dfd405c113cd708b341dd42c58",
               "service": "github",
               "@type": "Account"
          },
          {
               "proofType": "http",
               "identifier": "kathrin1598",
               "@type": "Account",
               "service": "twitter",
               "proofUrl":
"https://twitter.com/kathrin1598/status/855134802135199744"
          }
          ],
          "name": "Stanislav Pankratov",
          "image": [
          {
```

```
                "contentUrl":
"https://s3.amazonaws.com/dx3/stanislav_pankratov",
                "name": "cover",
                "@type": "ImageObject"
           },
           {
                "contentUrl":
"https://s3.amazonaws.com/kd4/stanislav_pankratov",
                "@type": "ImageObject",
                "name": "avatar"
           }
           ],
           "@type": "Person"
      },
      "expiresAt": "2018-04-20T20:27:23.518036",
      "subject": {
           "publicKey":
"03d55e4b8cf4b298ffab9638f182ea15d528ad78a0577eab1b09d93470775686
86"
      },
      "issuer": {
           "publicKey":
"03d55e4b8cf4b298ffab9638f182ea15d528ad78a0577eab1b09d93470775686
86"
      }
      },
      "signature":
"NWVAe3IU2UdxKiGj3nOv4oayelvrM3N-Qaw-l0mLk7gK8qg47fINIyayvLrcKGAB
brKvdsNPswCtEyhpr8b8GQ"
      },
      "token":
"eyJhbGciOiJFUzI1NksiLCJ0eXAiOiJKV1QifQ.eyJpc3N1ZWRBdCI6IjIwMTctM
DQtMjBUMjA6Mjc6MjMuNTE4MDM2IiwiY2xhaW0iOn...",
      "parentPublicKey":
"03d55e4b8cf4b298ffab9638f182ea15d528ad78a0577eab1b09d93470775686
86",
      "encrypted": false
   }
]
```

Список литературы

[1] Официальный веб-сайт ID2020. — URL: http://id2020.org/

[2] B. Kaliski. PKCS #5: Password-Based Cryptography Specification Version 2.0 // RFC 2898. — 2000. — С. 9—11

[3] Microsoft Building Open Blockchain-Based Identity System With Blockstack, ConsenSys, Bitcoin Magazine. — URL: https://bitcoinmagazine.com/articles/microsoft-building-open-blockchain-based-identity-system-with-blockstack-consensys-1464968713

[4] Introducing the Blockstack Identity System, Blockstack blog. — URL: https://blockstack.org/blog/introducing-the-blockstack-identity-system

[5] Dr. Christian Lundkvist, Rouven Heck, Joel Torstensson, Zac Mitton, Michael Sena. UPORT: A PLATFORM FOR SELF-SOVEREIGN IDENTITY. — URL: https://whitepaper.uport.me/uPort_whitepaper_DRAFT20170221.pdf — С. 2

[6] Andreas M. Antonopoulos. Mastering Bitcoin: Unlocking digital currencies. — "O'Reilly Media, Inc." — 2014.

[7] Погружение в технологию блокчейн: Децентрализованная беспарольная система безопасности. Хабрахабр, блог компании Microsoft. — URL: https://habrahabr.ru/company/microsoft/blog/316864/

[8] EMCSSL – Система идентификации пользователей WWW на основе подсистемы NVS криптовалюты EmerCoin и децентрализованных клиентских SSL-сертификатов. Хабрахабр. — URL: https://habrahabr.ru/post/257605/

[9] Официальный сайт Remme. — URL: http://remme.io/

[10] Blockstack | AgenlList. — URL: https://angel.co/blockstack

[11] Blockstack - Intro, официальный сайт Blockstack. — URL: https://blockstack.org/intro

[12] Muneeb Ali, Jude Nelson, Ryan Shea, Michael J. Freedman. Bootstrapping Trust in Distributed Systems with Blockchains // ;login: — VOL. 41, NO. 3 — 2016. — C. 56

[13] Blockstack, Onename, and future applications - Identity - Blockstack Forum. URL — https://forum.blockstack.org/t/blockstack-onename-and-future-applications/529

[14] Muneeb Ali, Jude Nelson, Ryan Shea, Michael J. Freedman. Blockstack: A Global Naming and Storage System Secured by Blockchains // 2016 USENIX Annual Technical Conference. — 2016. — C. 188

[15] Diploma project about Blockstack authentication protocol, Blockstack forum. — URL: https://forum.blockstack.org/t/diploma-project-about-blockstack-authentication-protocol/850

[16] Library for scanning blockchains and running Blockstack state engines, Github. — URL: https://github.com/blockstack/virtualchain

[17] Muneeb Ali, Jude Nelson, Ryan Shea, Michael J. Freedman. Blockstack: A Global Naming and Storage System Secured by Blockchains // 2016 USENIX Annual Technical Conference. — 2016. — C. 184—186

[18] OP_RETURN Stats. — URL: http://opreturn.org

[19] The reference implementation of Blockstack, Github. — URL: https://github.com/blockstack/blockstack-core

[20] The Blockstack Browser Portal, Github. — URL: https://github.com/blockstack/blockstack-portal

[21] JSON Web Tokens, jwt.io — URL: https://jwt.io

[22] Web framework rankings, HotFrameworks — URL: http://hotframeworks.com/

[23] Sony hacked yet again, plaintext passwords, e-mails, DOB posted, Ars Technica — URL: https://arstechnica.com/tech-policy/2011/06/sony-hacked-yet-again-plaintext-passwords-posted/

[24] LinkedIn suffers data breach - security experts, Reuters — URL: http://in.reuters.com/article/linkedin-breach-idINDEE8550EN20120606

[25] Yahoo Hack: 1bn accounts compromised by biggest data breach in history, The Guardian — URL: https://www.theguardian.com/technology/2016/dec/14/yahoo-hack-security-of-one-billion-accounts-breached

[26] 1 Million Gmail Users Impacted by Google Docs Phishing Attack, Treatpost — URL: https://threatpost.com/1-million-gmail-users-impacted-by-google-docs-phishing-attack/125436/

[27] Причиной недавней фишинговой атаки «Google Docs» стала уязвимость в OAuth, обнаруженная 6 лет назад, Tproger — URL: https://tproger.ru/news/google-docs-phishing-foretold-oauth/

[28] Customer Identity Preference Trends Q2 2016, LoginRadius — URL: https://blog.loginradius.com/2016/08/customer-identity-preference-trends-q2-2016/

[29] Number of Facebook users worldwide 2008-2017, Statista — URL: https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/

[30] Gmail Now Has More Than 1B Monthly Active Users, TechCrunch — URL: https://techcrunch.com/2016/02/01/gmail-now-has-more-than-1b-monthly-active-users/

[31] Facebook Connect Market Share and Web Usage Statistics, SimilarTech — URL: https://www.similartech.com/technologies/facebook-connect

[32] Email Providers: Gmail the Clear Winner, Mailchimp blog — URL: https://blog.mailchimp.com/major-email-provider-trends-in-2015-gmail-takes-a-really-big-lead/

[33] Heartbleed, Wikipedia — URL: https://en.wikipedia.org/wiki/Heartbleed

[34] Alexandre Borges. How to perform a Heartbleed Attack — URL: https://alexandreborgesbrazil.files.wordpress.com/2014/04/hearbleed_attack_version_a_1.pdf

[35] Jun Wang, Mingyi Zhao, Qiang Zeng, Dinghao Wu, Peng Liu. Risk Assessment of Buffer “Heartbleed” Over-read Vulnerabilities // 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. — 2015. — C. 557

[36] Jun Wang, Mingyi Zhao, Qiang Zeng, Dinghao Wu, Peng Liu. Risk Assessment of Buffer “Heartbleed” Over-read Vulnerabilities. C. 559-560

[37] Global Bitcoin Nodes Distribution, Bitnodes — URL: https://bitnodes.21.co/