**Автономная некоммерческая общеобразовательная организация "Физтех-лицей"
(АНОО «Физтех-лицей» им. П.Л. Капицы)**

XX научно-практическая конференция

«Старт в инновации»

Генератор временных паролей для персонального компьютера

Выполнили:
Емельянов Александр, 8 «Б» класс
Галинач Александр, 8 «Б» класс
Сырцев Филипп, 8 «Б» класс
Руководитель:
Мерзляков В.В.

Московская область, г. Долгопрудный

2021 г.

Содержание

Введение

Информационная безопасность – практика предотвращения несанкционированного доступа, использования, раскрытия, искажения, изменения, исследования, записи или уничтожения информации. В основе информационной безопасности лежит деятельность по обеспечению её конфиденциальности, доступности и целостности информации.

В связи с развитием информационных технологий одним из важнейших вопросов для любого пользователя становится обеспечение безопасности его учетных записей от потенциальной возможности неправомерного или случайного воздействия, приводящего к потере, искажению или разглашению конфиденциальных данных.

Для обеспечения дополнительной безопасности учетных записей пользователей, кроме логина и пароля, широко используется двухфакторная аутентификация. Работа таких аутентификаторов основана на генерации временных паролей в приложениях для смартфонов. Однако, в настоящее время отсутствуют сертифицированные программы, предназначенные для использования на персональном компьютере.

Таким образом, представленная работа относится к области информационной безопасности, основной целью которой является создание программного обеспечения для персонального компьютера, позволяющего генерировать временные пароли с целью авторизации пользователя на различных сайтах и в приложениях.

Достижение поставленной цели осуществляется путём решения следующих задач:

1. Изучение принципов работы существующих аутентификаторов для смартфонов – Google Authenticator и Яндекс.Ключ.
2. Создание серверной части для обработки запросов клиента, хранения и обработки информации в базе данных MySQL.
3. Разработка и программная реализация интерфейса аутентификатора.
4. Написание программного кода на языке C++ с использованием компонента CLR.

1. Анализ принципов работы существующих аутентификаторов

С ростом числа угроз персональным данным пользователей, становится все более необходимым обновлять стандарты безопасности сайтов и приложений для предотвращения несанкционированного доступа к их учетным записям.

Использовать для авторизации пользователя только логин и пароль опасно. Есть множество причин – от фишинга до вредоносных программ, по которым учетные данные могут оказаться в руках злоумышленников.

Для введения дополнительного уровня безопасности учетных записей используют двухфакторную аутентификацию – использование двух факторов проверки подлинности пользователя. Фактором в данном контексте называют способ аутентификации, т.е. проверки подлинности. Иногда используется сокращение 2FA (2-Factor Authentication). Также можно встретить синоним «мультифакторная аутентификация». По данным Google, она эффективно защищает пользователей от автоматических атак и фишинга практически в 100 % случаев и в 66 % при целевых атаках.

2FA – это метод идентификации пользователя в каком-либо сервисе (как правило, в интернете) при помощи запроса аутентификационных данных двух разных типов, что обеспечивает двухслойную, а значит, более эффективную защиту аккаунта от несанкционированного проникновения. Это означает, что после включения двухфакторной аутентификации пользователь должен пройти еще один шаг для успешного входа в систему.

На первом шаге входа в учетную запись вводятся логин и пароль Включение двухфакторной аутентификации добавляет, например, проверку одноразового кода, сгенерированного с помощью специального приложения.

Этот метод более безопасен, так как злоумышленник не может получить доступ к учетной записи пользователя, если у него нет доступа как к обычному паролю пользователя, так и к одноразовому паролю.

В настоящее время существует два широко используемых метода получения одноразового пароля:

1. На основе SMS. Каждый раз, когда пользователь входит в систему, он получает на указанный в учетной записи номер мобильного телефона или адрес электронной почты текстовое сообщение, которое содержит одноразовый пароль.

2. На основе TOTP («Time-based One-Time Password Algorithm» – «одноразовый пароль на основе времени»). При включении двухфакторной аутентификации пользователю предлагается отсканировать QR-код или ввести секретный код вручную с помощью специального приложения для смартфона, которое в дальнейшем постоянно генерирует одноразовый пароль для пользователя.

Несмотря на свою простоту, метод на основе SMS имеет ряд проблем. Например, ожидание SMS при каждой попытке входа в систему, проблемы с безопасностью и т. д. Вследствие чего Национальный институт стандартов и технологий США (NIST) еще в 2016 году рекомендовала не использовать его в новых системах аутентификации. В связи с минусами метода на основе SMS, метод на основе TOTP становится популярным из-за его преимуществ.

Также, стоит отметить, что в настоящий момент есть некоторые разногласия о том, что именно считать шагами, а что факторами аутентификации. Общепризнанным является существование трех факторов:

– знание, например, пароль;

– обладание (в физическом смысле), например, смартфон;

– То, чем вы являетесь, например, биометрические данные

При этом шаги аутентификации являются субъединицами факторов, так, если для входа в систему необходимо введение двух паролей, то по сути мы используем только фактор знания. Если говорить о рассматриваемых нами методах, то метод на основе SMS

принято относить к двухшаговой, но однофакторной аутентификации, т. к. пароль для SMS генерируется на сервере, а TOTP к двухфакторной, поскольку для генерации пароля необходимо наличие определенного приложения на смартфоне, что усложняет задачу доступа злоумышленников к этой информации.

Используя для двухфакторной аутентификации метод на основе TOTP, мы создаем одноразовый пароль на стороне пользователя (а не на стороне сервера) через приложение для смартфона. Это означает, что у пользователя всегда есть доступ к своему одноразовому паролю. А также предотвращает отправку текстового сообщения сервером при каждой попытке войти в систему. Кроме того, сгенерированный пароль изменяется через определенный промежуток времени, что делает его, по сути, одноразовым.

Для реализации двухфакторной аутентификации с использованием TOTP необходимо учитывать основное требование – пароль должен создаваться на стороне пользователя, а также постоянно меняться.

Решение данной задачи может выглядеть следующим образом:

1. Внутренний сервер создает секретный ключ для этого конкретного пользователя.
2. Затем сервер передает этот секретный ключ телефонному приложению пользователя.
3. Телефонное приложение инициализирует счетчик.
4. Телефонное приложение генерирует одноразовый пароль, используя этот секретный ключ и счетчик.
5. Телефонное приложение изменяет счетчик через определенный интервал и восстанавливает одноразовый пароль, делая его динамическим.

Сейчас многие крупные компании, такие как Microsoft, Google, Dropbox, Facebook, Яндекс уже предоставляют возможность применения двухфакторной аутентификации. Причем для них всех можно использовать единое приложение-аутентификатор, соответствующее определённым стандартам, например, Microsoft Authenticator, Authy или FreeOTP, а также Google Authenticator и Яндекс.Ключ, являющиеся наиболее используемыми на территории Российской Федерации.

Google Authenticator – приложение для двухэтапной аутентификации с помощью Time-based One-time Password Algorithm (TOTP) и HMAC-based One-time Password Algorithm (HOTP) от Google LLC (рисунок 1).

Как правило, пользователи должны сначала установить приложение на своё мобильное устройство. Authenticator представляет 6- или 8-значный одноразовый цифровой пароль, который пользователь должен предоставить в дополнение к имени пользователя и пароля (ввести в специальное поле), чтобы войти в службы Google или других сервисов. Authenticator также может генерировать коды для сторонних приложений, такие как менеджеры паролей или услуг хостинга файлов.

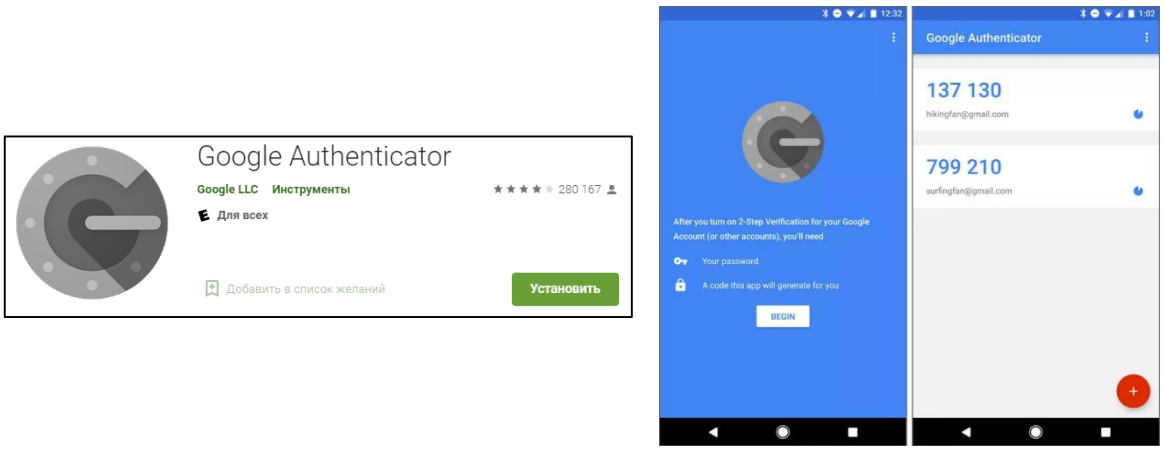


Рисунок 1 – Внешний вид приложения Google Authenticator

Приложение Яндекс.Ключ также использует двухфакторную аутентификацию на основе алгоритма TOTP, дополнительно защищает различные учетные записи от взлома, создавая одноразовые пароли для доступа к Почте и другим сервисам Яндекса (рисунок 2). Получать эти пароли может только пользователь, в своём мобильном телефоне – по уникальному пин-коду, который Яндекс выдает во время настройки Ключа. На веб-сервисы можно заходить, введя одноразовый пароль вручную, или по QR-коду, который формируется в форме авторизации.

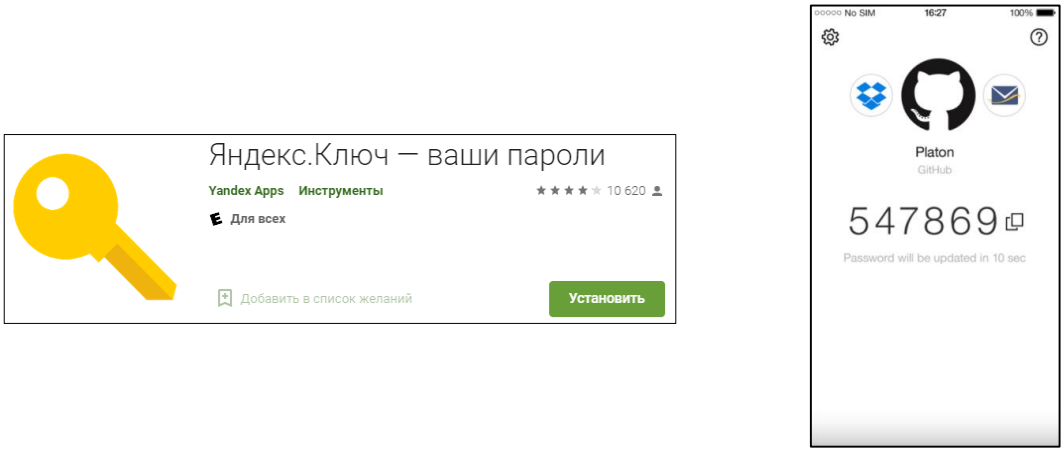


Рисунок 2 – Внешний вид приложения Яндекс.Ключ

Преимущество двухфакторной аутентификации через мобильное устройство:

– не нужны дополнительные токены, потому что мобильное устройство всегда под рукой;

– код подтверждения постоянно меняется, а это безопаснее, чем однофакторный логин-пароль.

Недостатки двухфакторной аутентификации через мобильное устройство:

– мобильный телефон должен ловить сеть, когда происходит аутентификация, иначе сообщение с паролем просто не дойдет;

– необходимо сообщить номер мобильного телефона, из-за чего, например, в будущем может приходить спам;

– текстовые сообщения (SMS), которые, поступая на мобильный телефон, могут быть перехвачены;

– текстовые сообщения приходят с некоторой задержкой, так как некоторое время уходит на проверку подлинности;

– современные смартфоны используются для получения как почты, так и SMS. Как правило, электронная почта на мобильном телефоне всегда включена. Таким образом, все аккаунты, для которых почта является ключом, могут быть взломаны;

– при потере аппаратного токена или смартфона с приложением для аутентификации, может быть утрачена возможность доступа к аккаунту без резервных способов входа.

Рассмотренные приложения для двухфакторной аутентификации используются только на смартфонах (планшетах).

На данный момент для использования двухфакторного аутентификатора на персональном компьютере (ПК) существуют:

1. Эмуляторы операционной системы (ОС) Android, такие как Nox и BlueStacks и др., позволяющие установить и запустить необходимое приложение из официального магазина. Существенным недостатком таких программ является большое потребление внутренней памяти ПК, а также они обладают очень низкой скоростью запуска эмулируемой ОС (рисунок 3).

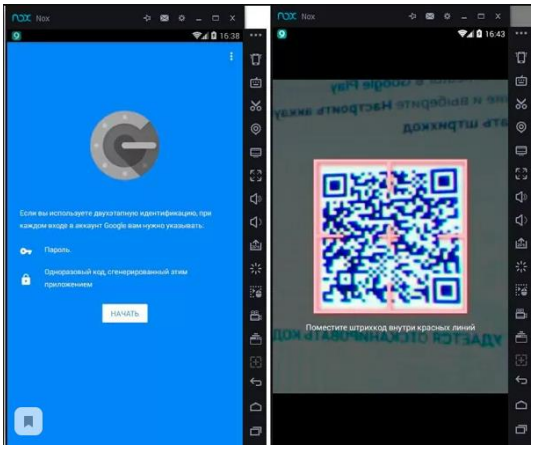


Рисунок 3 – Внешний вид приложения Google Authenticator в эмуляторе Nox

2. Расширения для популярных браузеров: Chrome, Edge, Firefox (рисунок 4). Главными недостатками подобных продуктов являются: отсутствие сертификации, необходимость запуска браузера перед началом работы, отсутствие возможности добавления нового сервиса без QR-кода.

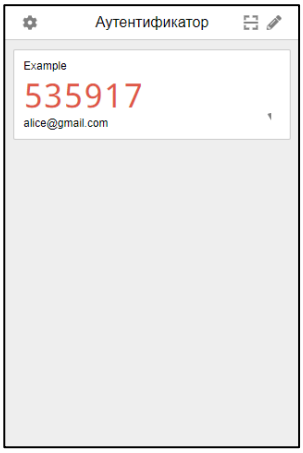


Рисунок 4 – Внешний вид расширения authenticator.cc для браузера

3. Сторонние пользовательские программные продукты (рисунок 5). Подобные разработки не имеют лицензии и сертификации, многие из них имеют закрытый исходный код, что не позволяет обеспечить безопасность от компьютерных вирусов.

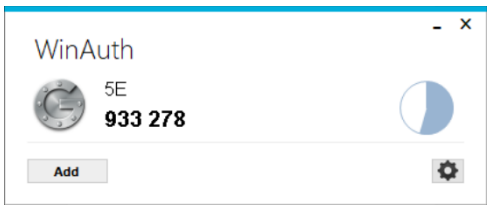


Рисунок 5 – Внешний вид программы WinAuth, взятой с GitHub

Таким образом, задача разработки версии генератора паролей двухфакторной аутентификации на основе алгоритма TOTP для персонального компьютера с открытым исходным кодом и устранением программных недостатков вышеуказанных продуктов, является актуальной.

2. Создание серверной части для обработки запросов клиента, хранения и обработки информации в базе данных MySQL

Для создания серверной части для обработки запросов клиента, хранения и обработки информации в базе данных MySQL был выбран язык программирования PHP.

Это позволяет проводить:

– авторизацию по логину и паролю,

– регистрацию новых пользователей,

– восстановление пароля,

– работу с почтой: (подтверждение, генерацию письма с подтверждением почты, реализацию рассылки информации),

– работу с GoogleAuthenticator:

– получение значения кода,

– добавление нового ключа,

– удаление ключа,

– изменение login'a у элемента,

– генерацию QR кода существующего элемента.

PHP – скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов. Для корректной работы серверной части приложения необходимо обновить версию PHP до версии 7.2+.

MySQL (Structured Query Language – язык структурированных запросов) – популярная свободная реляционная система управления базами данных с моделью «клиент-сервер». Ее также можно рассматривать как сервер базы данных. Он создан для обеспечения доступа к данным для других сервисов и приложений.

Принцип работы MySQL-сервера такой же, как в любых клиент-серверных моделях. Одно устройство делает запрос, а второе отвечает. Запрашивающих может быть больше одного, все зависит от сервера, сети и поставленных задач.

Технически немного иные, но по своей сути идентичные процессы происходят в среде MySQL:

– система создает базу данных для хранения информации (ее сортировки, идентификации и т.п.);

– клиенты (другие компьютеры в сети) подают запросы к базе с помощью специфичных для SQL команд;

– серверное приложение обрабатывает запрос и выдает ответ клиенту (выдает запрашиваемые данные).

Для функционирования серверной части разработанного приложения используется таблица базы данных MySQL, имеющая 5 столбцов, каждый из которых имеет определенный тип данных и назначение хранимой информации (рисунок 6).

Название колонны	Тип данных	Назначение
Id	INT	Id пользователя
Username	TEXT	Имя пользователя
Email	TEXT	Электронная почта
Password	TEXT	Пароль
SecretKeys	LONGTEXT	Информация о сервисах

Рисунок 6 – Формат таблицы базы данных, используемой на серверной части приложения

Для разработки страницы для генерации кодов приложений была использована библиотека GoogleAuthenticator, позволяющая генерировать QR-код, секретный ключ и получать значения временного пароля с помощью алгоритма TOTP.

С целью защиты базы данных от SQL инъекции, являющейся одним из наиболее распространенных методов взлома веб-страниц, был введен алгоритм проверки входных данных GET запроса на вредоносный код.

Для удобства процесса авторизации пользователя дополнительно введена возможность использования установленного адреса электронной почты в качестве логина.

3. Разработка интерфейса и программного кода приложения

3.1 Разработка интерфейса приложения

На основе анализа существующих двухфакторных аутентификаторов, выполненного в первой главе, для разрабатываемого приложения были взяты элементы графического интерфейса Google Authenticator и расширения authenticator.cc.

Первоначальный дизайн-макет интерфейса программы выполнялся в многофункциональном графическом редакторе Adobe Photoshop. Далее, необходимо было создать приложение с использованием оконного интерфейса, с помощью которого реализовать ранее созданный макет. В качестве основного языка программирования был выбран C++, средой разработки являлась Visual Studio 2019.

Существует несколько эффективных решений для программирования оконных приложений на языке C++. Например, кроссплатформенный фреймворк Qt, позволяющий создавать приложения на многих популярных языках программирования: данная платформа является основой среды KDE, используемой во многих версиях ОС Iinux. Основным преимуществом является возможность компиляции для разных версий операционных систем без изменения исходного кода, наличие всех необходимых классов для работы как с обычными приложениями, так и с сетью, базами данных. Однако, данная платформа имеет и недостатки, например, большой объем приложений после разработки.

Одной из альтернатив QT является среда Common Language Runtime (CLR), архитектура которой определена спецификацией CLI (Common Language Infrastructure). Данная среда отвечает за ряд задач, основными являются изоляция памяти приложений, проверка типов, преобразования IL (набор инструкций, не зависящих от платформы) в машинный код и так далее. Фактически, код приложения не компилируется в машинный код, но хранится в наборе инструкций на CIL, которые исполняются CLR для конкретной платформы. Таким образом, обеспечивается кроссплатформенность при меньшем объеме исполняемых файлов. Наглядно представление о роли среды CLR при компиляции.NET приложений продемонстрирована на рисунке 8.

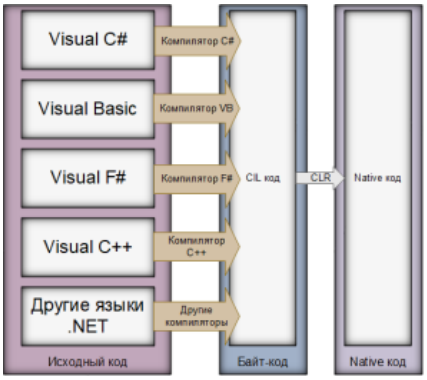


Рисунок 9 – роль среды CLR при компиляции.NET приложении

Для разработки программного интерфейса была выбрана именно CLR из-за меньшего количества необходимых для хранения файлов и хорошей кроссплатформенности. Также для создания приложения с использованием Qt необходим дополнительный редактор – Qt Editor, что является значительным минусом, так как изначально была выбрана среда разработки Visual Studio.

В процессе решения поставленной задачи был создан проект CLR, в который позже были добавлены формы Windows Forms, на которых создавались и программировались необходимые компоненты.

Windows Forms — интерфейс программирования приложений (API), отвечающий

за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. Создание ПО с использованием Windows Forms становится возможным в проекте C++ CLR и итоговый продукт не будет отличаться по сложности написания от написанного на C#, VB.NET, J# и других языков, поддерживающих .NET. Процесс создания программного интерфейса приложения в среде Visual Studio 2019 показан на рисунке 7.

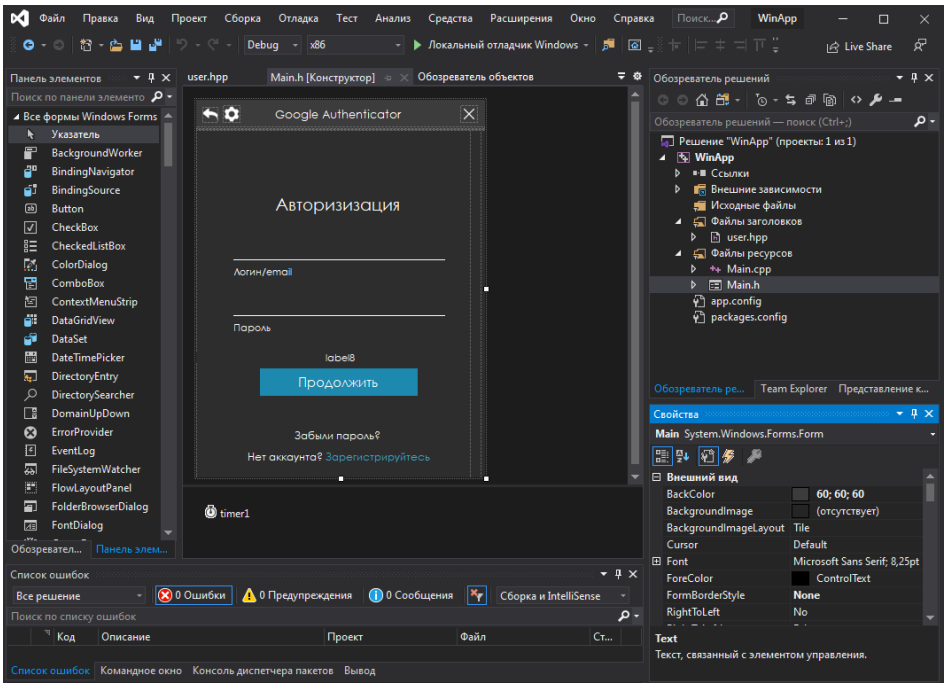
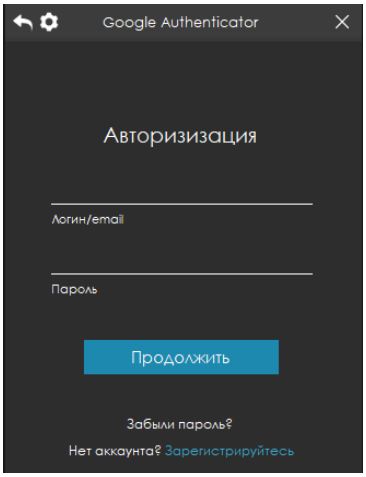
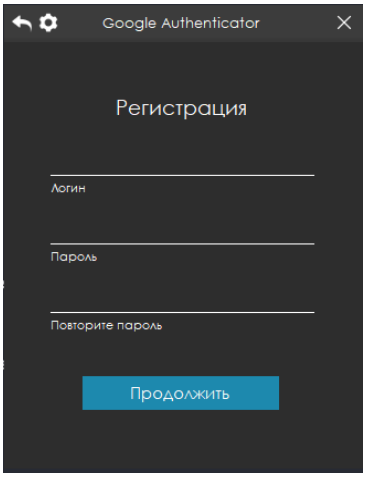


Рисунок 7 – Разработка окна авторизации пользователя в Visual Studio 2019

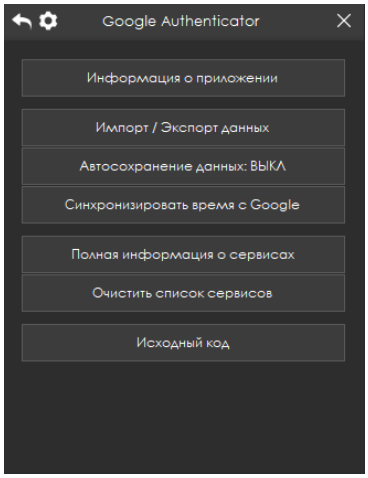
В результате проделанной работы был создан и запрограммирован на обработку различных событий компонентов графический интерфейс приложения, представленный на рисунке 9.



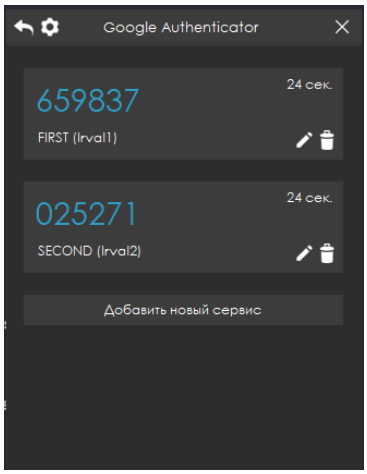
а)



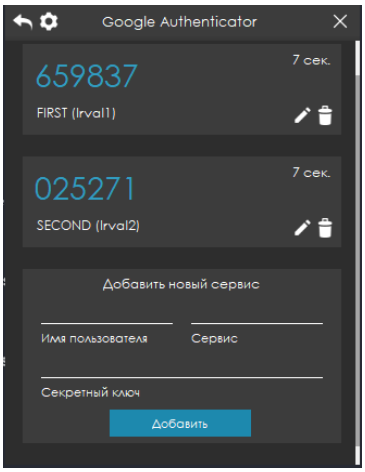
б)



в)



г)



д)

Рисунок 9 – Графический интерфейс разработанного приложения:
а) окно авторизации пользователя; б) окно регистрации нового пользователя;
в) окно настроек приложения; г) окно с кодами двухфакторной аутентификации;
д) панель добавления нового сервиса

В отличие от существующих аутентификаторов разработанный интерфейс приложения совместно с программным кодом позволяют:

– производить синхронизацию данных пользователя (авторизацию, регистрацию, подтверждение почты, восстановление пароля) с удаленным сервером;

– производить генерацию кодов, как на сервере, так и на клиенте в офлайн режиме;

– добавлять новый сервис с заданными значениями без сканирования QR-кода;

– сканировать QR-код непосредственно с экрана компьютера, путем выделения необходимой области;

– удалять всю сохраненную информацию;

– работать без автосохранения настроек на сервере и в папке AppData.

3.2 Разработка программного кода приложения

Для написания кода приложения был использован язык программирования C++ с компонентом .NET Framework CLR.

В разработанном ПО генератор одноразовых кодов для двухфакторной аутентификации построен на основе алгоритма TOTP, в котором в качестве значения счетчика подставляется величина, зависящая от времени. Обозначим:

- T – дискретное значение времени, используемое в качестве параметра.
- X – интервал времени, в течение которого действителен пароль. (По умолчанию 30 сек.)
- T_0 – начальное время, необходимое для синхронизации сторон.
- K – разделяемый секрет.
- $CurrentTime$ – текущее время.

Тогда:

$$T = (CurrentTime\ -\ T_0)/X$$

$$HOTP(K,T) = Truncate\big(HMAC\text{-}SHA\text{-}1(K,T)\big)$$

$$TOTP = HOTP(K,T)$$

Где:

$HMAC\text{-}SHA\text{-}1(K,T)$ – генерация 20-ти байт на основе секретного ключа и времени с помощью хеш-функции.
$Truncate$ – функция выбора определенным способом 4 байт:
Обозначим $String$ – результат $HMAC\text{-}SHA\text{-}1(K,T)$; $OffsetBits$ — младшие 4 бита строки $String$; $Offset = StringToNumber(OffsetBits)$ и результатом $Truncate$ будет строка из четырёх символов — $String[Offset]$ … $String[Offset + 3]$.

Данный алгоритм реализуется в библиотеках GoogleAutheticator, используемых в разработке офлайн режима, а также на удаленном сервере.

Концепция одноразовых паролей в совокупности с современными криптографическими методами может использоваться для реализации надежных систем удаленной аутентификации. TOTP достаточно устойчив к криптографическим атакам, однако вероятности взлома есть.

Прослушивая трафик клиента, злоумышленник может перехватить посланный логин и одноразовый пароль (или хеш от него). Затем ему достаточно блокировать компьютер «жертвы» и отправить аутентификационные данные от собственного имени. Если он успеет это сделать за промежуток времени X, то ему удастся получить доступ. Именно поэтому X стоит делать небольшим. Но если время действия пароля сделать слишком маленьким, то в случае небольшой рассинхронизации клиент не сможет получить доступ.

Приложение имеет два режима работы. При включении клиент-серверного режима, абсолютно все вычисления производятся на сайте и результат передается клиенту с помощью GET запросов, вся информация также хранится на удаленном сервере в базе данных MySQL.

Для отправки GET запроса на сервер и получения ответа в виде строки типа System::String^ используется функция sendRequest:

```
System::String^ sendRequest(System::String^ uri) {
	System::String^ resp;
	HttpWebRequest^ request = dynamic_cast<HttpWebRequest^>(WebRequest::Create(uri));
	request->MaximumAutomaticRedirections = 4;
	request->MaximumResponseHeadersLength = 4;
	request->Credentials = CredentialCache::DefaultCredentials;
	HttpWebResponse^ response = dynamic_cast<HttpWebResponse^>(request-
>GetResponse());
	Stream^ receiveStream = response->GetResponseStream();
	StreamReader^ readStream = gcnew StreamReader(receiveStream, Encoding::UTF8);
	resp = readStream->ReadToEnd();
	response->Close();
	readStream->Close();
	return resp;
}
```

Полученный ответ от сервера десериализируется с помощью библиотеки Newtonsoft.JSON в экземпляр класса User:

```
ref class App {
public:
	String^ QR;
	String^ Login;
	String^ Secret;
	String^ Site;
	String^ Code;
};
ref class User {
public:
	String^ Id;
	String^ Username;
```

```
        String^ Email;
        String^ Password;
        array<App^>^ Applications;
};
```

Вся полученная информация обрабатывается и отображается в отдельных компонентах на главной форме приложения.

Также в программе имеется возможность включения офлайн режима, используемого для увеличения скорости получения одноразовых кодов, за счет полной автономности программы от удаленного сервера. В случае активации данной функции, для генерации кодов требуется библиотека Google.Authenticator, а вся сохраненная информация находится в директории AppData и при желании может импортироваться/экспортироваться (данная функция также доступна в клиент-серверном режиме).

4. Заключение

В ходе проведенной работы был создан программный продукт, позволяющий получать пароли для двухфакторной аутентификации и имеющий определенные преимущества перед уже существующими приложениями. ПО может быть запущено на компьютере без установки дополнительных программ, для работы требуется только один из самых популярных фреймворков – .NET Framework. При разработке интерфейса приложения были учтены недостатки в удобстве и простоте работы с Google Authenticator, authenticator.cc, Яндекс.Ключ. Итоговый продукт обладает рядом отличительных возможностей, позволяющих:

– производить синхронизацию данных пользователя (авторизацию, регистрацию, подтверждение почты, восстановление пароля) с удаленным сервером;

– производить генерацию кодов, как на сервере, так и на клиенте в офлайн режиме;

– добавлять новый сервис с заданными значениями без сканирования QR-кода;

– сканировать QR-код непосредственно с экрана компьютера, путем выделения необходимой области;

– удалять всю сохраненную информацию;

– работать без автосохранения настроек на сервере и в папке AppData.

Для разработки серверной части приложения использовался язык программирования PHP и база данных MySQL, это позволяет использовать сохраненные клиентом данные и получать результаты серверных вычислений с любого устройства, имеющего доступ к интернету. В этом состоит одно из основных преимуществ решения перед работы с сокетами.

Клиентское приложение под Windows создавалось в среде разработки Visual Studio 2019 с использованием технологии Windows Forms, позволяющей упростить процесс создания графического интерфейса, путем использования стандартных компонентов системы. В качестве основного языка программирования использовался C++ с применением среды выполнения .NET CLR.

5. Список использованных источников

1. Средства визуального программирования на языке C++. Среда CLR https://moluch.ru/archive/205/50253/
2. Многофакторная аутентификация http://ru.wikipedia.org/wiki/ Многофакторная аутентификация
3. Немного о 2FA: Двухфакторная аутентификация http://habr.com/ru/company/1cloud/blog/277901/
4. Time-based One-time Password Algorithm https://ru.wikipedia.org/wiki/Time-based_One-time_Password_Algorithm
5. TOTP (Time-based one-time Password algorithm) https://habr.com/ru/post/534064/
6. Google Authenticator https://ru.wikipedia.org/wiki/Google_Authenticator
7. 3 лучших приложения для двухфакторной аутентификации для Android и iOS https://levashove.ru/3-applications-for-two-step-authentication/