УДК 004.056

СРАВНЕНИЕ МЕТОДОВ ХРАНЕНИЯ И ЗАЩИТЫ ПОЛЬЗОВАТЕЛЬСКИХ ПАРОЛЕЙ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

Фур М.А., студент гр. ИБт-231, III курс
Научный руководитель: Дементьева Ю.С., преподаватель кафедры ИБ
Кузбасский государственный технический университет
имени Т.Ф. Горбачева
г. Кемерово

Информационные системы (ИС) стали неотъемлемой частью жизни современного общества. Согласно исследованию Password Statistics (How Many Passwords Does An Average Person Have?) [1], средний пользователь интернета имеет более 100 учётных записей на различных онлайн-сервисах. Каждая из них требует регистрации и создания уникальных учётных данных — логина и пароля. Однако рост количества утечек персональных данных вызывает серьёзную обеспокоенность: по данным Центра ресурсов по борьбе с кражей личных данных [2], жертвами компрометации данных стали около 1 миллиарда человек. Любая организация, располагающая информационной системой, обязана обеспечивать защиту пользовательских аккаунтов от несанкционированного доступа. В связи с этим появляется актуальный вопрос: «Каким образом следует хранить и защищать пароли пользователей?» Целью данной работы является показать на практике, как разные способы хранения паролей влияют на безопасность информационной системы.

Для реализации поставленной цели были обозначены следующие задачи:

Задачи:

1. Разработать простую программу авторизации пользователя (ввод логина и пароля).
2. Реализовать несколько вариантов хранения паролей:
3. Реализовать защищённые методы хранения паролей:
4. Провести экспериментальное сравнение реализованных методов:
5. Сделать выводы о безопасности различных методов хранения паролей и обосновать необходимость использования криптографических методов защиты.

Для выполнения данной работы был определён ряд этапов, включающих изучение криптографической библиотеки для языка программирования C#, изучение видов хранения и защиты паролей.

Первым этапом стало исследование теоретических материалов, касающихся парольной безопасности. На данном этапе были изучены: виды и алгоритмы парольной аутентификации в различных ИС, методы защиты от несанкционированного доступа. Обобщены механизмы работы аутентификации независимо от метода хранения.

После сбора теоретических сведений приступили к практической части. Была создана тестовая программа, имитирующая работу аутентификации в ИС с вариантами выбора метода хранения и защиты пароля. На данном этапе была проведена работа по изучению криптографической библиотеки «Security.Cryptography» языка C#. Разработано приложение с графическим интерфейсом на платформе Windows Forms, позволяющее регистрировать пользователей и выполнять вход в систему с использованием трёх различных методов хранения паролей и трёх методов защиты.

Завершающим этапом работы стало сравнение методов хранения и защиты. На данном этапе проводился анализ уязвимостей каждого из реализованных способов. Для имитации несанкционированного доступа к паролям использовалась как заранее встроенная в программу система, так и простой доступ к файлам программы.

Хранение пароля может выполняться разными способами: прямо в программе, текстовом файле, базе данных. А также есть разные варианты защиты паролей, исследование проводится с отсутствующей защитой, хеширование и хеширование с солью.

Сформируем термины хеширования и соли. Хеширование – это математический алгоритм, который преобразовывает произвольный массив данных в состоящую из символов (букв и цифр) строку фиксированной длины. [3] Соль – это случайная строка, которая в свою очередь добавляется к паролю перед его хешированием. Она делает хеш различным для одинаковых паролей, что предотвращает использование радужных таблиц (инструмента для взлома хешей и паролей) и усложняет атаки методом полного перебора (когда злоумышленник выполняет перебор паролей до тех пор, пока не найдёт верный). [4]

Для выполнения исследования была создана простая программа с графическим интерфейсом (Рисунок 1), вверху экрана можно выбрать вариант теста, чуть ниже зарегистрироваться и пройти аутентификацию. Нижняя кнопка проводит эмуляцию взлома, а снизу выводятся данные по работе программы.

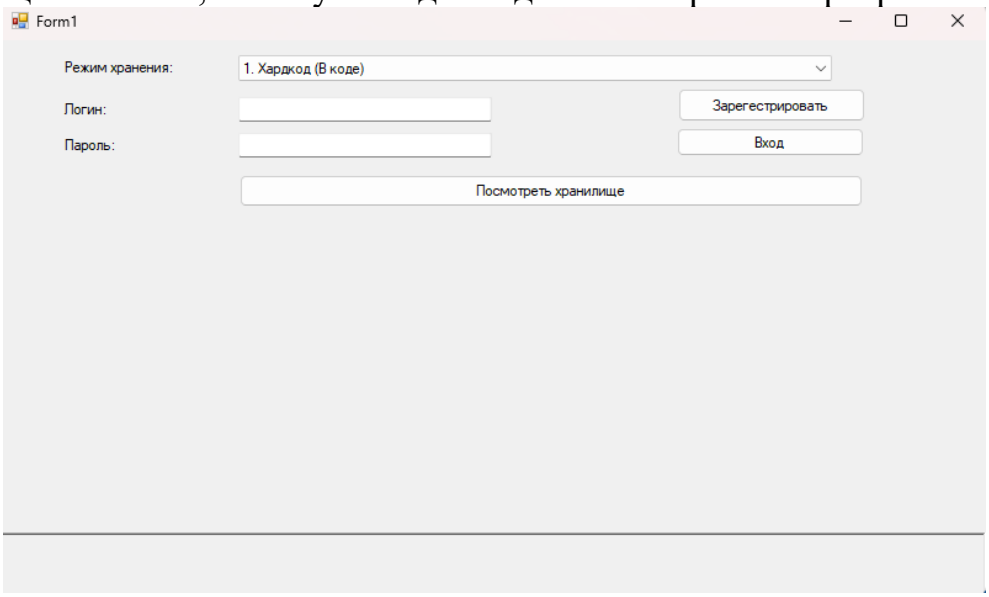

Рисунок 1 – Интерфейс программы

Первым вариантом у нас идёт проверка хранения пароля прямо в коде (Рисунок 2)

```
// Хардкод пароль (для демонстрации уязвимости)
private const string HardcodedPassword = "123";
```

Рисунок 2 – Пароль сохранён в коде

В случае если злоумышленник заполучит программу, или доступ к ней он сможет декомпилировать программу, и получить доступ к паролю. К тому же такой пароль может быть создан лишь единожды для одного аккаунта, так как данные хранятся напрямую в коде.

Далее была проведена проверка в хранении в текстовом файле (Рисунок 3). Данное хранение уже позволяет свободно регистрировать множество аккаунтов, однако открытое хранение не обеспечивает должной безопасности. Так злоумышленник легко получит доступ если сможет получить несанкционированный доступ к серверу ИС. Все пароли лежат в одном файле, помимо кражи самих паролей злоумышленник может и стереть информацию о всех аккаунтов нажатием нескольких кнопок.

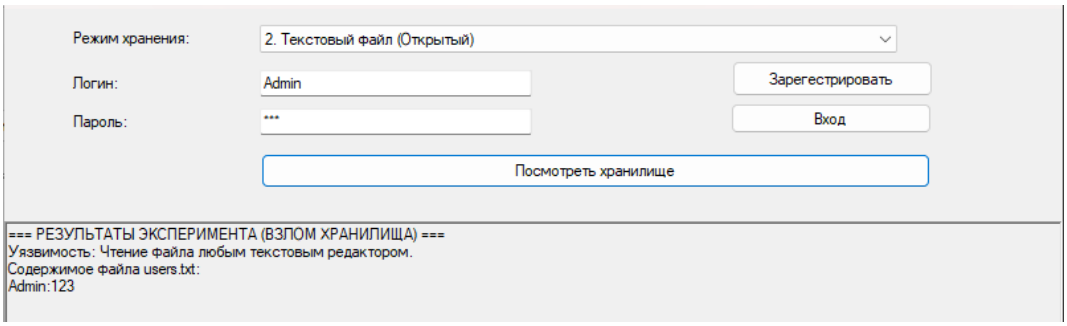

Рисунок 3 – Чтение текстового файла

Заключающим методом хранения является База данных (Рисунок 4). Пароль хранится всё так же без дополнительной защиты, однако теперь чтобы заполучить доступ к хранимым данным мало просто получить доступ к файлам, в большинстве ИС которые используют базы данных используется репликация и резервное копирование, благодаря этому злоумышленник не сможет безвозвратно удалить все данные. В свою очередь многие базы данных имеют встроенные защиты от несанкционированного доступа. Но открывают возможность для SQL-инъекций, хотя большинство современных решений уже умеют этому противодействовать.

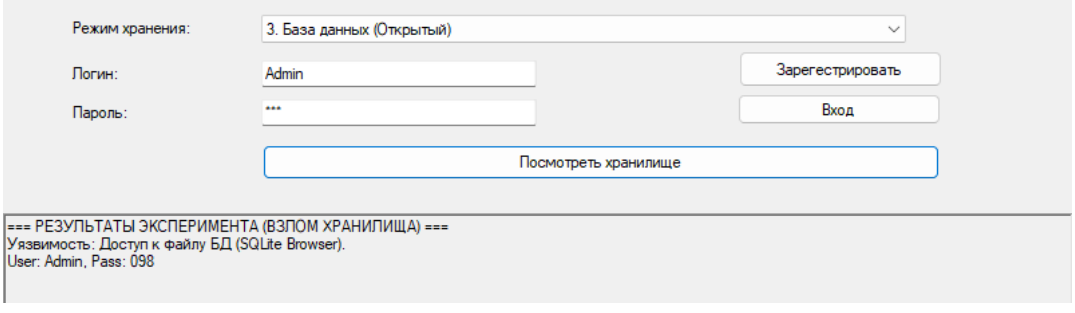

Рисунок 4 – Хранение в базе данных

Перейдём к методам защиты паролей. Здесь для удобства мы будем хранить пароли в текстовом файле, так как он не нагружает компьютер в отличии от базы данных.

Первым рассмотренным методом станет хеширование на примере SHA-256 (Рисунок 5 – 6)

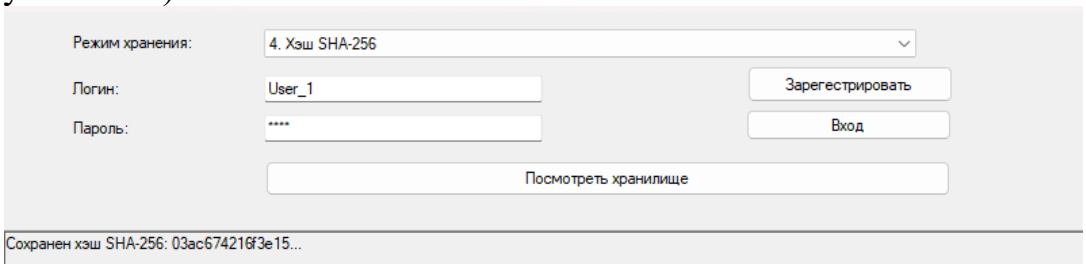

Рисунок 5 - Регистрация с хешем

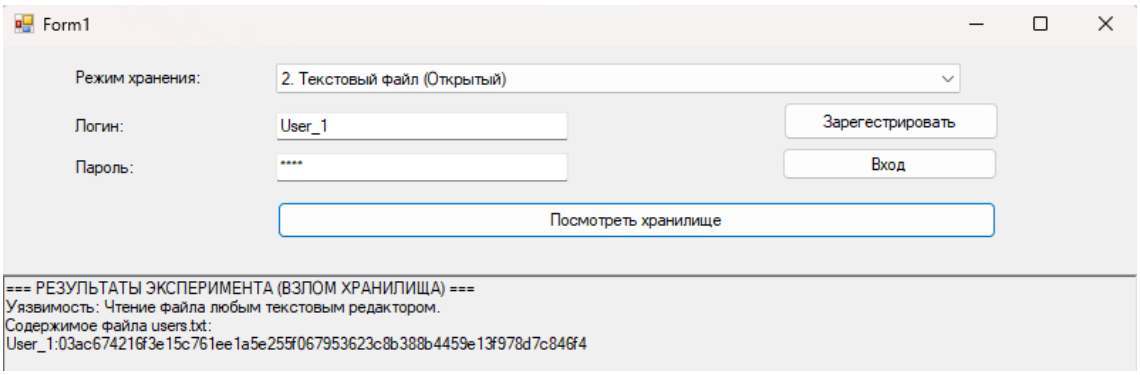

Рисунок 6 – Получение доступа к файлу с Хешем

Хеш долгое время являлся хорошим способом защиты паролей, пока не начали появляться радужные таблицы. Радужная таблица – это массив данных, содержащий заранее просчитанный хеш для определенного количества распространенных паролей. [2] Благодаря нему можно быстро выискивать простые пароли, находящиеся в реестре паролей. Для демонстрации был создан ещё один аккаунт с паролем 1234 (Рисунок 7). Такие простые пароли всегда есть в радужных таблицах, благодаря чему злоумышленнику достаточно просто найти этот хеш в файле.

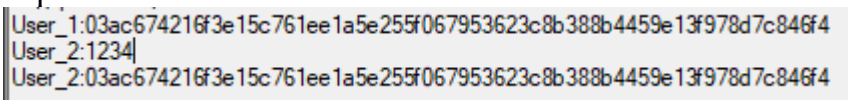

Рисунок 7 – Одинаковый хеш

Следующим методом является хеш с солью (Рисунок 8). Этот метод решает проблему радужных таблиц. При вновь одинаковых паролях хеш будет различен, так как для каждого пароля генерируется своя соль, которая добавляется к паролю. Даже если злоумышленник получит соль ему не получится так же быстро получить доступ к ИС. Так же для улучшения безопасности соль не просто встаёт в начало или конец строки, она может вставляться после определенного количества символов, или, например для паролей с чётным количеством знаков в середину, а с нечётным после третьего символа. Что в совокупности делает пароли почти полностью защищёнными при прямом доступе к хранилищу. Однако простые и распространённые пароли все так же являются

уязвимостью для такой защиты. Злоумышленник просто будет перебирать при помощи компьютера возможные варианты, но это займёт время в которое администрация ИС может успеть заметить открывшуюся уязвимость.

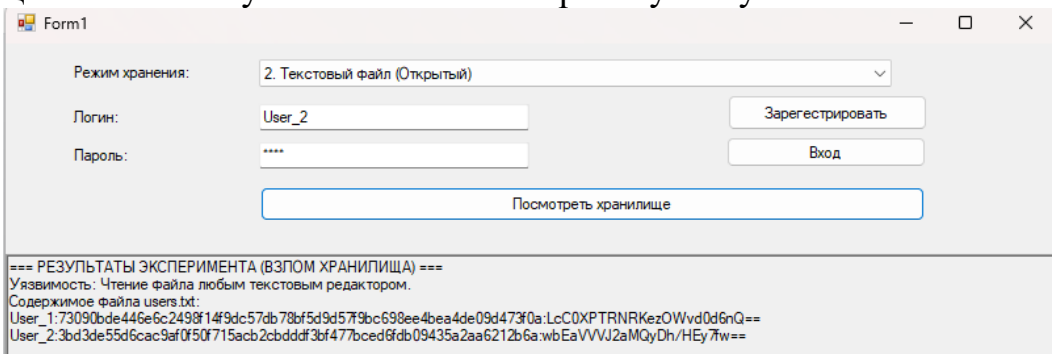

Рисунок 8 – Хеш с солью

На последнем этапе для максимализации защищённости наших паролей мы будем использовать базу данных. Сам же метод защиты - PBKDF2. Создадим четыре аккаунта с паролем: «123456789» (Рисунок 9). Пароли одинаковы, а вот хеш разный как при использовании обычной соли. Отличие от просто использования соли является множественность итерации. Хеширование пароля происходит многократно, в примере используется десять тысяч итераций. Такое многократное хеширование полностью исключает возможность использования радужных таблиц, а также в несколько раз увеличивает время перебора при помощи компьютера.

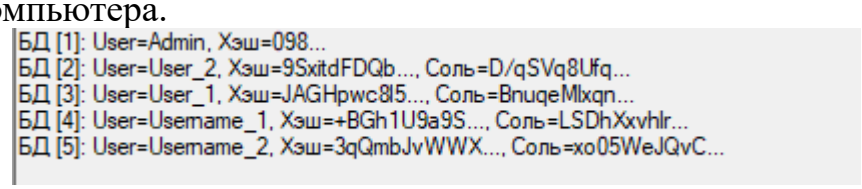

Рисунок 9 – Самый защищённый способ хранения

Подводя итог проведённого исследования, следует отметить, что выбор метода хранения паролей является критически важным фактором безопасности ИС. В ходе работы были решены все поставленные задачи: разработана программа авторизации, реализованы шесть вариантов хранения паролей (три небезопасных и три защищённых), проведено экспериментальное сравнение и сформулированы выводы об эффективности различных подходов.

В результате исследования выявлено, что хранение паролей в открытом виде создаёт критические уязвимости. При хранении в программном коде пароль может быть получен путём декомпиляции исполняемого файла за время менее одной минуты. Хранение в текстовом файле позволяет злоумышленнику получить доступ ко всем учётным записям при наличии доступа к файловой системе сервера. Хранение в базе данных без защиты хотя и обеспечивает сохранность данных благодаря механизмам репликации и резервного копирования, всё же оставляет пароли уязвимыми для SQL-инъекций и прямого чтения из таблиц.

Защищённые методы хранения демонстрируют существенно более высокий уровень безопасности. Хэширование по алгоритму SHA-256 защищает от

прямого чтения паролей, однако остаётся уязвимым для атак с использованием радужных таблиц, что позволяет восстанавливать простые пароли за время от нескольких секунд до одного часа. Добавление уникальной соли к каждому паролю полностью исключает возможность применения радужных таблиц и вынуждает злоумышленника применять метод полного перебора, что увеличивает время компрометации до нескольких часов или дней.

Наиболее эффективным методом защиты признано использование специализированного алгоритма PBKDF2. Многократное хэширование (в данной работе использовано 10 000 итераций) увеличивает время подбора пароля в тысячи раз, делая атаку экономически нецелесообразной. Даже при наличии одинаковых паролей у разных пользователей, хэши будут различными благодаря уникальной соли, что дополнительно затрудняет анализ утечки.

Список литературы:

1. Silkalns A. Password Statistics (How Many Passwords Does An Average Person Have?) / Aigars Silkalns [Электронный ресурс] // colorlib. : [сайт]. — URL: https://colorlib.com/wp/password-statistics/ (дата обращения: 22.03.2026).

2. Орбелиани М. Взломать за 60 секунд / Орбелиани М. [Электронный ресурс] // Коммерсантъ : [сайт]. — URL: https://www.kommersant.ru/doc/7737707 (дата обращения: 22.03.2026).

3. Donohue B. Чудеса хеширования / Donohue B. [Электронный ресурс] // kaspersky : [сайт]. — URL: https://www.kaspersky.ru/blog/the-wonders-of-hashing/3633/ (дата обращения: 22.03.2026).

4. Хеширование и хеш-функция / [Электронный ресурс] // Сбербанк : [сайт]. — URL: https://www.sberbank.ru/ru/person/kibrary/vocabulary/kheshirovanie-i-khesh-funkciya (дата обращения: 22.03.2026).

5. Bwall , Drone Надлежащее хэширование паролей Подробнее: https://www.securitylab.ru/analytics/427930.php / Bwall , Drone [Электронный ресурс] // SecurityLab : [сайт]. — URL: https://www.securitylab.ru/analytics/427930.php (дата обращения: 22.03.2026).