УДК 004.42

А. В. Голигузов[1]*

Разработка кроссплатформенного программного обеспечения для хранения паролей

[1] Национальный исследовательский ядерный университет «МИФИ», г. Москва, Российская Федерация
* e-mail: gav503244@gmail.com

Аннотация. В статье исследуются уязвимости в работе базовых протоколов стека TCP/IP, уязвимости операционных систем UNIX/Linux, проблемы существующих менеджеров паролей, методы безопасного автоматизированного последовательного развертывания и интеграции. Разработана архитектура базы данных, используемой для локального хранения. В результате исследования реализовано кроссплатформенное программное решение, обеспечивающее информационную безопасность процесса хранения конфиденциальных данных с использованием наилучших практик по обеспечению комплексной безопасности разрабатываемого приложения. В качестве стека технологий использовался язык C++, пользовательский интерфейс написан на фреймворке Qt. Приложением поддерживается множественное шифрование и российские криптоалгоритмы. Основные компоненты были протестированы с помощью модульных тестов, написанных с использованием библиотеки GTest, во время разработки использовался статический анализатор кода PVS-Studio, а также автоматизированный конвейер сборки, тестирования и развертывания посредством GitHub Actions.

Ключевые слова: менеджер паролей, множественное шифрование, кроссплатформенная разработка, C++, KeePass

A. V. Goliguzov[1]*

Development of cross-platform password storing software

[1] National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Moscow, Russian Federation
* e-mail: gav503244@gmail.com

Abstract. The article explores vulnerabilities in the underlying protocols of the TCP/IP stack, vulnerabilities in UNIX/Linux operating systems, problems in existing password managers, methods for secure automated sequential deployment and integration. The architecture of the database used for local storage has been developed. As a result of the study, a cross-platform software solution was implemented that provides information security for the process of storing confidential data using the best practices for ensuring the integrated security of the application being developed. The C++ language was used as a technology stack, the user interface was written with the Qt framework. The application supports multi-layer encryption and Russian crypto algorithms. The main components were tested using unit tests written with GTest library; during development, the PVS-Studio static code analyzer was used, as well as an automated build, test, and deployment pipeline through GitHub Actions.

Keywords: password manager, multi-layer encryption, cross-platform development, C++, KeePass

Введение

В современном мире практически каждый человек имеет доступ в глобальную сеть Интернет. Общедоступность интернет-ресурсов привела, в частности, к созданию различных персонализированных сервисов, как-то социальные сети, почтовые сервисы и форумы. В них необходима аутентификация. Потребность в использовании таких сервисов (у достаточно продвинутого пользователя их количество может оцениваться сотнями) сопровождается необходимостью создавать на каждом из них новый аккаунт. Большинство людей используют одну и ту же связку логина и пароля, что является крайне небезопасным. Это дает возможность злоумышленнику, который получил доступ к одному из сервисов, получить доступ ко всем, используя те же самые аутентификационные данные. Создание надежных паролей для различных сервисов, а также запоминание их в голове является трудной задачей, здесь на помощь и приходят менеджеры паролей, которые отвечают за генерацию и обеспечение информационной безопасности процесса хранения конфиденциальных данных.

Актуальность данной работы связана с периодическими утечками различных персональных данных, в том числе и паролей. 16 декабря 2022 года пресс-служба Роскомнадзора заявила, что после начала специальной военной операции России на Украине в интернет попало порядка 600 млн записей о россиянах, за это время произошло не менее 140 утечек личных данных. Буквально за неделю до этого, 13 декабря 2022 года стало известно о том, что в открытый доступ попали данные миллионов пользователей «Московской электронной школы» (МЭШ). В конце августа 2022 года стало известно об утечке данных пользователей Start. По итогам первой половины 2022 года экспертно-аналитическим центром InfoWatch в мире зарегистрирована 2101 утечка информации ограниченного доступа, что почти в два раза (на 93,2%) больше, чем за аналогичный период прошлого года. Количество утечек в России за первое полугодие 2022 года составило 305 (+45,9% по сравнению с I полугодием 2021 года).

Исследование предметной области

Вероятные векторы атак могут находиться на различных уровнях: при передаче по сети, в операционной системе и пользовательском приложении.

В мире повсеместно используется стек протоколов TCP/IP для коммуникации. Первоначальная версия стека протоколов TCP/IP была разработана почти 50 лет назад, и за это время в нем было найдено много проблем, которые решались путем внедрения новых протоколов [1]. Для понимания векторов потенциальных проблем нужно иметь представление о строении стека протоколов. Такой стек состоит из четырёх уровней: прикладной, транспортный, межсетевой и сетевого доступа.

Самым нижним является уровень сетевых интерфейсов, он определяет соглашения по проведению взаимодействий внутри локальной сети (англ. LAN). Одной из такой атак является ARP-spoofing [2], так как в протоколе отсутствует проверка подлинности, а на сетевых интерфейсах возможен самопроизвольный

ARP, злоумышленник отсылает жертве ARP-ответ, содержащий неверные данные, что приводит к обновлению ARP-таблицы на стороне жертвы. Другим примером является атака MAC-spoofing, заключающаяся в подмене MAC-адреса, что приводит к обману жертвы или к нарушению логической связности сети [3].

На следующем уровне, сетевом, происходит маршрутизация в сети. Наиболее популярными атаками этого уровня являются IP address spoofing и Man-in-the-middle. Первая из них заключается в фальсификации отправляемых заголовков IP-пакетов, например, в корпоративных сетях, где внутренние системы доверяют друг другу. Злоумышленник может выдать себя за другого и получить авторизованный доступ к целевой машине без аутентификации [4]. Вторая является целым семейством атак и заключается в расположении злоумышленника между общающимися конечными узлами, за счет чего он может просматривать весь идущий трафик, а также при необходимости изменять его [5].

На транспортном уровне происходит передача данных через Интернет. Основным видом атак являются DoS-атаки, например, TCP SYN Flood, UDP Flood, RIPv1DDoS [6, 7].

На последнем, прикладном уровне, обеспечивающим взаимодействие программного обеспечения с сетью, наблюдается наибольшее количество атак. На этом уровне очень много различных протоколов (HTTP, SMTP, SNMP, FTP, SQL, DNS), определить атаку намного труднее, чем на более низких уровнях, так как на этом уровне существует наибольшее количество уязвимостей [8].

Таким образом, каждый из уровней стека TCP/IP предоставляет возможности для проведения атак. Во многих протоколах отсутствует шифрование и контроль аутентификации. Из этого следует, что необходимо разрабатывать дополнительное программное обеспечение, чтобы избежать уязвимостей, содержащихся в стеке протоколов TCP/IP. Одно из возможных решений представлено в [9]. Оно заключается в использовании комбинированного подхода к динамическому созданию инкапсулированных виртуальных сетевых туннелей с использованием луковой и чесночной маршрутизации, а также дополнительных уровней шифрования.

Операционные системы UNIX/Linux тоже не лишены недостатков. В них существует 2 пространства: пространство ядра (англ. kernel space) и пространство пользователя (англ. user space). Такое распределение было придумано для контроля действий пользователя. В случае необходимости исполнения кода пользователя в ядре используются системные вызовы (англ. system calls), являющиеся безопасным интерфейсом взаимодействия, но существуют варианты выполнения кода пользователя в ядре в обход системных вызовов – такие варианты являются уязвимостями [10].

В списке 25 самых опасных слабых мест программного обеспечения CWE 2021 года есть такие уязвимости, как (в скобках указан порядковый номер в списке) [11]:

– Out-of-Bounds write (1);
– Out-of-Bounds read (3);
– Improper Input Validation (4);

– Use after Free (7);
– Integer Overflow or Wraparound (12);
– NULL pointer dereference (15);
– Use of Hard-coded Credentials (16).

Так как ядро операционной системы написано на языке Си, все вышеприведенные уязвимости могут быть найдены в общедоступном коде (что и происходит на практике).

Для безопасного хранения данных существуют специализированные приложения – менеджеры паролей. Их разделяют на 3 типа: локальные, облачные и портативные. Каждый из типов тем или иным образом подвержен взлому. Ниже кратко рассматриваются наиболее популярные из них.

Программа LastPass, являющаяся самым популярным менеджером паролей на 2022 год, использует облачное хранение паролей. В августе 2022 года менеджер подвергся хакерской атаке, в результате которой была похищена часть исходного кода. В декабре 2022 года была проведена повторная атака, в ходе которой было взломано облачное хранилище и конфиденциальные данные клиентов были украдены, про компрометацию паролей неизвестно.

Другим примером менеджера паролей является программа KeePass. Это кроссплатформенная программа для хранения паролей. В ней используется локальное хранение данных в файлах с расширением *.kdbx. Это бинарный файл, хранящий зашифрованный XML-документ, содержащий пароли. Для него известна man-in-the-middle атака при использовании обновлений, основанная на использовании при обновлении HTTP и отсутствии верификации пакетов.

Еще одной популярной программой для хранения паролей является 1Password. Внутри него есть функция 1PasswordAnywhere, которая позволяет пользоваться сервисом где и когда угодно, не прибегая для этого к клиентскому ПО 1Password. Данная функция использует формат Agile Keychain, то есть пользовательские данные, фактически, хранятся в виде JavaScript-файлов. Если открыть его через HTTP, вы увидите серую страницу, иконку, изображающую замок и предложение ввести пароль. После ввода мастер-пароля можно разблокировать keychain и получить доступ к информации. Проблема в том, что метаданные пользователя не шифруются.

Постановка задачи

Целью настоящей работы является обеспечение информационной безопасности процесса хранения конфиденциальных данных.

Для решения задачи цель была декомпозирована на следующие подзадачи:

– исследование предметной области;

– проектирование архитектуры программного обеспечения (UML диаграмма классов);

– программная реализация предложенного решения (MVP);

– проведение тестирования разработанного программного обеспечения.

Предлагаемое решение

Предлагаемое решение основано на использовании многоуровневого шифрования. Файл, в котором будут храниться пароли, представляет собой бинарный файл с расширением *ssdb* (англ. *Safe Storage DataBase*). В начале файла содержится сигнатура – 8 байт, равные 0x12345678. После сигнатуры в файле располагается верхний слой. Слой состоит из набора заголовков и данных (рис. 1).

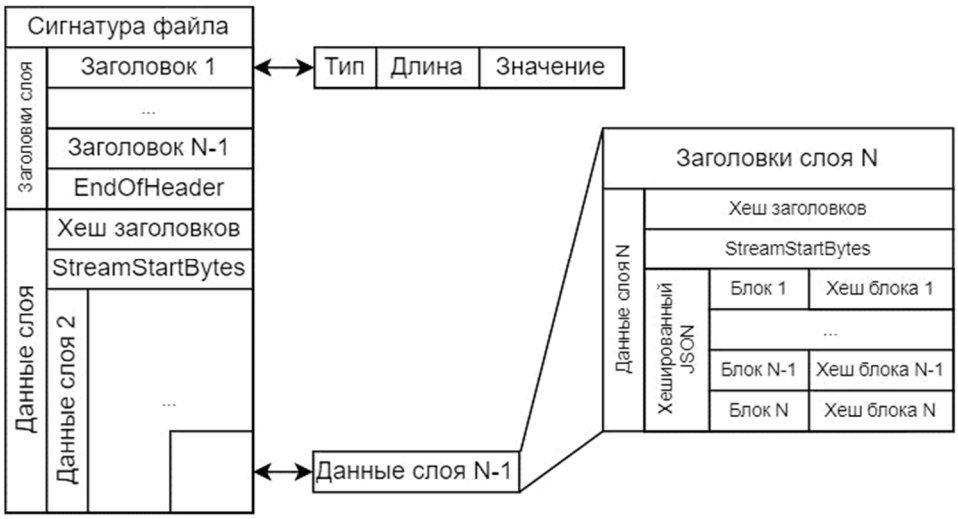

Рис. 1. Структура файла базы данных

Набором заголовков является последовательность Тип-Длина-Значение, оканчивающаяся заголовком, обозначающим конец. Тип заголовка является обязательным и занимает 1 байт, после чего следует также обязательная длина – 2 байта, после которого хранится значение, занимаемое количество байт которого зависит от длины (в случае, если длина равна нулю, значение заголовка может отсутствовать, например, заголовок EndOfHeader).

После заголовков следуют данные. В начале данных содержатся 64 байта, используемых для проверки целостности слоя. Первые 32 являются SHA256 хешем заголовка, остальные 32 – байты из заголовка StreamStartBytes. Для промежуточных слоев (при множественном шифровании) данными являются зашифрованные нижележащие слои. Алгоритм расшифровки приведен в виде блок-схемы на рис. 2.

Для последнего слоя данными является зашифрованный документ в формате JSON, захешированный блоками и, опционально, сжатый выбранным методом сжатия. Размер захешированного блока составляет 16 Кб. Сам JSON содержит объект, внутри которого идут необходимые пользовательские записи вида. Опционально эти записи могут быть обфусцированы.

Внутренняя архитектура приложения представляет из себя статическую библиотеку Core, прилинкованную к исполняемому файлу, отвечающему за графическую визуализацию. Такой подход позволяет удобно использовать библиотеку с различными UI фронтэндами: графическим (англ. GUI) или командным

(англ. CLI), вне зависимости от их реализации, делая код в библиотеке независимым, инкапсулируя всю логику в изолированный модуль. Диаграмма классов библиотеки представлена на рис. 3. Устройство библиотеки можно логически разделить на 4 части.

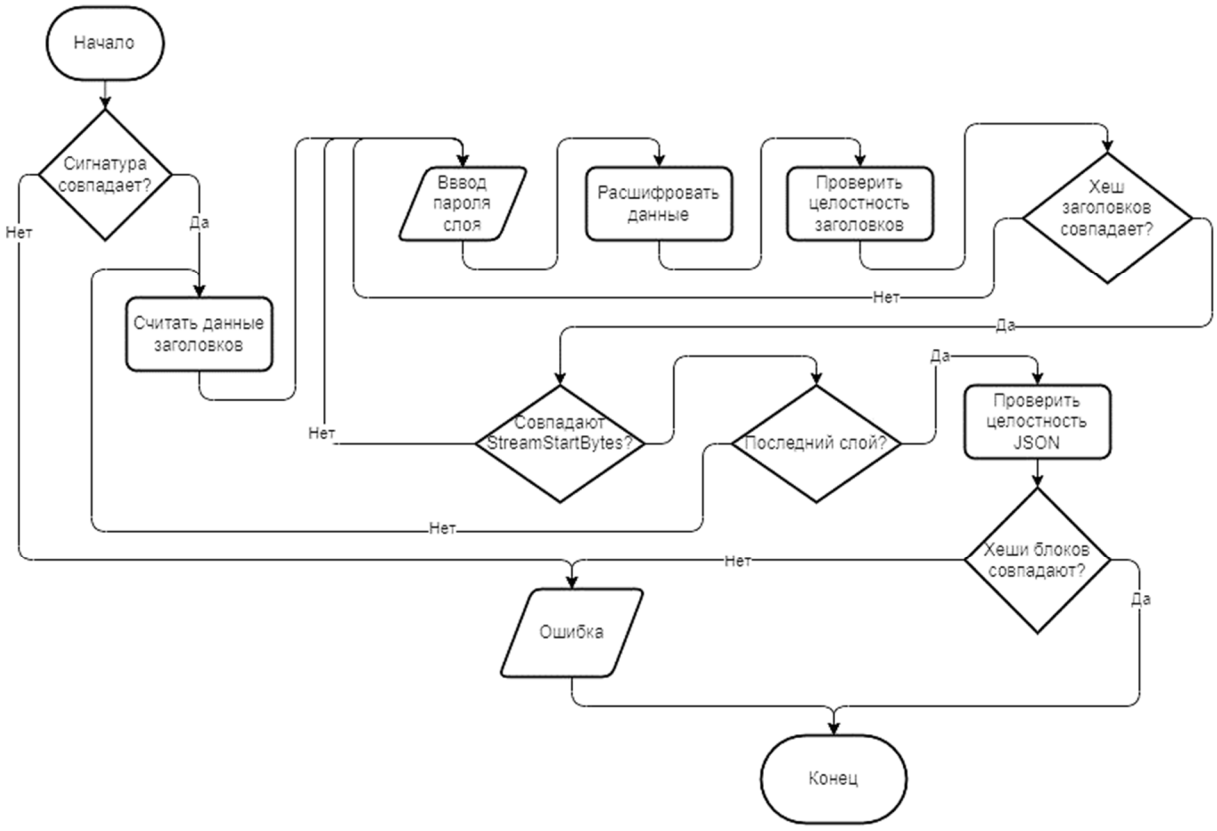

Рис. 2. Блок-схема алгоритма расшифрования БД

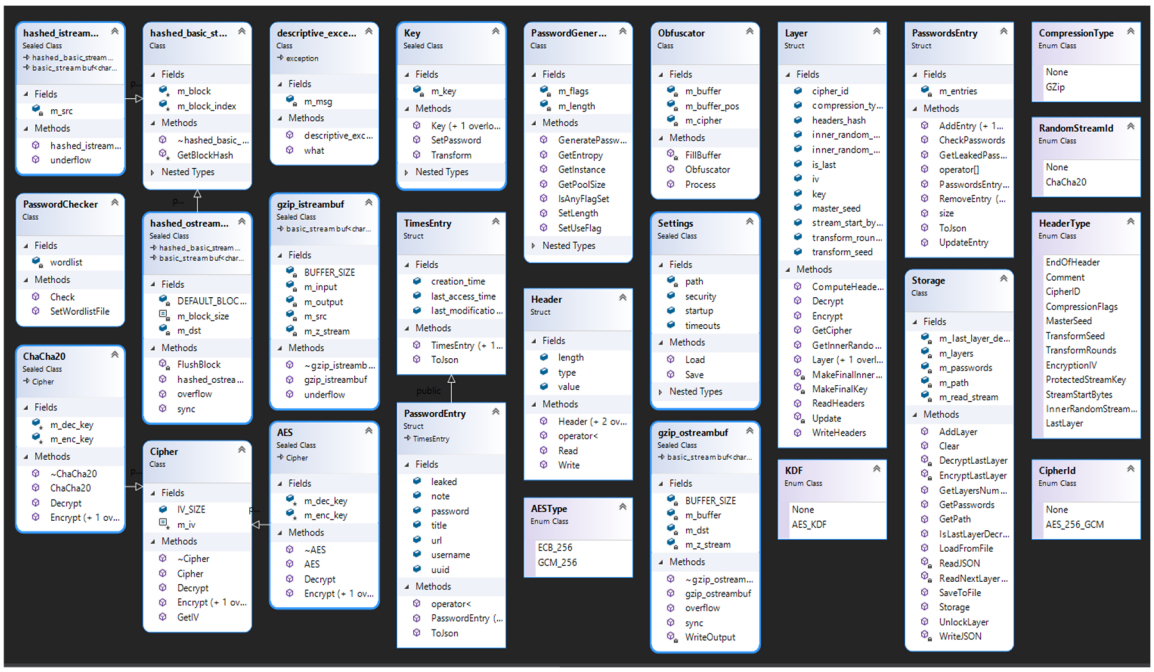

Рис. 3. Диаграмма классов библиотеки Core.

Первая из них – Streams, в этой части присутствуют надстройки над стандартными стримами для ввода и вывода – классы hashed_istreambuf и hashed_ostreambuf для хешированного, gzip_istreambuf и gzip_ostreambuf для сжатого ввода/вывода. Вторая – Crypto, в ней реализованы надстройки над библиотекой openssl. Это базовый интерфейс Cipher, представляющий возможности шифрования/расшифрования некоторых данных, и его наследники – классы AES, являющиеся фабрикой для алгоритмов шифрования семейства AES, Grasshopper – реализация российского алгоритма Кузнечик, ChaCha20, реализующий соответствующий алгоритм шифрования, используемый в качестве обфускатора пароля в записи БД, сам Obfuscator, занимающийся обфускацией, и класс Key – билдер мастер-ключа. Третьей логической составляющей является Struct – части структурного представления БД. К ней относятся TimesEntry, PasswordEntry, PasswordsEntry – классы, представляющие внутреннее строение JSON-на (для работы с ним в памяти и сериализации/десериализации); класс Header, то есть заголовок уровня; класс Layer, представляющий из себя слой зашифрованных данных; класс Storage, реализующий последовательное чтение/запись файла базы данных, а также хранилище текущих данных. К оставшейся части относятся вспомогательные классы-синглтоны, занимающиеся конкретными задачами: PasswordGenerator отвечает за генерацию паролей, PasswordChecker проверяет пароли на нахождение во вшитом вордлисте пароля, попавшего в общий доступ, Settings – класс для загрузки/сохранения и оперирования текущими настройками приложения.

Тестирование разработанного решения

Во время разработки программного продукта использовалась методология DevSecOps. Для автоматизации процесса непрерывной интеграции и непрерывной поставки (CI/CD), статического анализа, а также проведения автоматического тестирования был разработан пайплайн для Github Actions, приведенный в виде BPMN-диаграммы на рис. 4, написанный на человекочитаемом языке сериализации данных YAML.

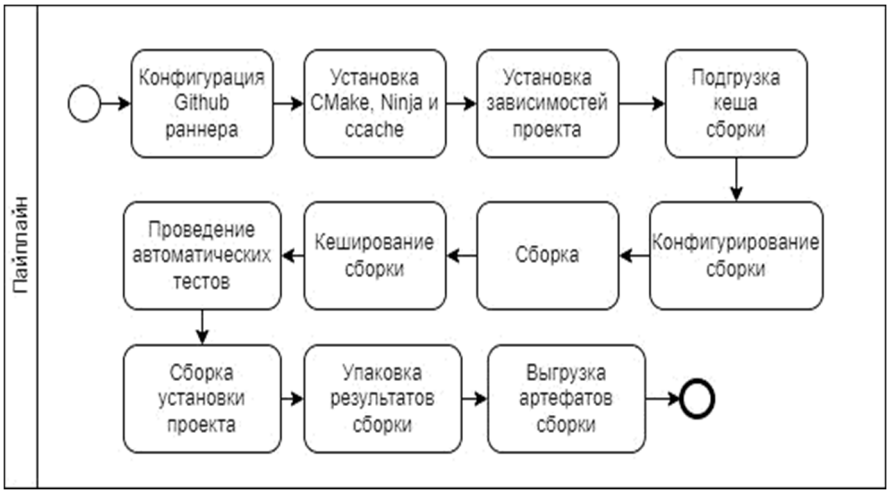

Рис. 4. BPMN-диаграмма пайплайна

Во время разработки активно использовался статический анализатор PVS-Studio. Для него был выбран режим MODE GA:1,2,3 OP:1,2,3 64:1,2, что обозначает использование диагностики общего плана General (GA), основной набор диагностических правил PVS-Studio с номерами уровней достоверности предупреждений (1, 2, 3), использования диагностик микрооптимизаций Optimization (OP), содержащий указания по повышению эффективности и безопасности кода, а также диагностики, позволяющие выявлять специфические ошибки, связанные с разработкой 64-битных приложений или переносом кода с 32-битной на 64-битную платформу 64-bit (64).

Наряду с использованием статического анализатора кода были написаны модульные тесты с использованием кроссплатформенного фреймворка GoogleTest. Они отвечают за тестирование корректности проведения операций шифрования и расшифрования.

В качестве еще одного подхода к тестированию реализованного программного обеспечения использовалось ручное тестирование. С помощью него были проведены тестирование взаимодействия пользователя с графическим функционалом.

Заключение

В данной работе было спроектировано, разработано и протестировано кроссплатформенное программное обеспечение для хранения паролей.

Главным итогом работы является собственная программная реализация менеджера паролей. Плюсами данной программной реализации являются:

– возможность использования разработанного менеджера паролей на различных видах операционных систем (Windows, Linux, macOS);

– свойство локального хранения зашифрованной информации, обеспечивающее постоянный доступ к ней;

– использование различных методов улучшения выработки мастер-ключа для увеличения безопасности зашифрованных данных к атаке полного перебора значений ключа криптографического алгоритма;

– возможность проверки паролей и уведомления, хранящихся в базе данных, на наличие утечек по встроенному списку;

– поддержка множественного шифрования базы данных;

– поддержка российских криптоалгоритмов;

– различные варианты настройки используемых алгоритмов шифрования, дополнительной обфускации и компрессии данных.

В ходе работы были получены следующие результаты:

– исследована предметная область;

– спроектирована архитектура программного обеспечения (диаграмма классов);

– создан минимально жизнеспособный продукт (MVP);

– проведено тестирование разработанного программного обеспечения.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Finding and Exploiting Vulnerabilities in Embedded TCP/IP Stacks / Wang, Wenhui. // MS thesis, University of Twente. - 2021.

2. D-ARP: An Efficient Scheme to Detect and Prevent ARP Spoofing / Morsy, Sabah M., Dalia Nashat. // IEEE Access. - 2022. – Vol. 10, – P. 49142-49153.

3. Reliable Monitoring Security System to Prevent MAC Spoofing in Ubiquitous Wireless Network / Ullas, S. U., J. Sandeep // Advances in Big Data and Cloud Computing: Proceedings of ICBDCC18. Springer Singapore. - 2019.

4. Classification of DHCP spoofing and effectiveness of DHCP Snooping / Akashi, Shigeo, Yao Tong. // Proceedings on 2018 International Conference on Advances in Computer Technology, Information Science and Communication, edited by Wen-Bing Horng and Yong Yue. - 2019. – Vol. 233.

5. Man-in-the-middle-attack: Understanding in simple words / Mallik, Avijit. // Cyberspace: Jurnal Pendidikan Teknologi Informasi. - 2019. – Vol. 2, iss. 2. – P. 109-134.

6. Deep learning binary fruit fly algorithm for identifying SYN flood attack from TCP/IP / Nagaraju V., [et al.]. // Materials Today: Proceedings. – 2021.

7. Mitigating SYN Flooding and UDP Flooding in P4-based SDN / Shen, Zi-Yang, [et al.]. // 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE. - 2021.

8. Hierarchical anomaly-based detection of distributed DNS attacks on enterprise networks / Lyu, Minzhao, [et al.]. // IEEE Transactions on Network and Service Management. - 2021. – Vol. 18, iss. 1. – P. 1031-1048.

9. System of a self-organizing virtual secure communication channel based on stochastic multi-layer encryption and overlay technologies / E. A. Basinya, Z. B. Akhayeva, D. H. Omarkhanova, G. B. Tolegenova [et al.]. – Text : direct // Journal of Theoretical and Applied Information Technology. - 2022. – Vol. 100, iss. 16. – P. 4918-4927.

10. Analysis and study of security mechanisms inside Linux Kernel / Zhai, Gaoshou, Yaodong Li. // International Conference on Security Technology. - 2008

11. CWE Top 25 Most Dangerous Software Weaknesses / Summers, Alec, [et al.]. // - 2022.