Міністерство освіти і науки України

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
завідувач кафедри кібербезпеки
та захисту інформації
_____Наталія ЛУКОВА-ЧУЙКО
«14» червня 2022р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

дипломної роботи

бакалавра

(назва освітнього рівня)

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітня програма Кібербезпека
(назва освітньої програми)

на тему: «Механізм обміном даними між менеджерами паролей»

Виконавець: студент IV курсу, групи КБ-42

_____ **Микита МИРОВЕЦЬ**
(підпис) (прізвище ім'я по-батькові)

	Прізвище, ініціали	**Підпис**
Керівник	Микола БРАІЛОВСЬКИЙ.	

Нормоконтроль	Юрій ЩЕБЛАНІН.	

Київ 2022

Міністерство освіти і науки України

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри кібербезпеки
та захисту інформації
_____Наталія ЛУКОВА-ЧУЙКО
«01» листопада 2021 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студентові **КБ-42** **Мировцю Микиті Євгеновичу**
(група) (прізвище ім'я по-батькові)

Тема дипломної роботи «Механізм обміном даними між менеджерами паролей»

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема дипломної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №5 від 29.10.2021р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Засоби аутентифікації, види паролів, поради щодо складності паролю, схеми роботи програм, код програми

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Отримання доступу до персональних даних, засоби автентифікації, пароль, як спосіб захисту від несанкціонованого доступу, персональні дані, витоки, аналіз
Налаштування захисту локальних мереж, моніторингу та рекомендації щодо використання

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність розробка програмного забезпечення, що вирішує проблему безпечного експорту даних між менеджерами паролів

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 01.11. 2021 року

Завдання видав	_____	Микола БРАІЛОВСЬКИЙ.
	(підпис)	(ініціали, прізвище)
Завдання прийняв до виконання	_____	Микита МИРОВЕЦЬ
	(підпис)	(ініціали, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	01.11.2021 – 27.01.2022	*виконано*
2	Аналіз літератури	28.01.2022 – 11.02.2022	*виконано*
3	Розгляд структури локальних мереж	12.02.2022 – 24.02.2022	*виконано*
4	Дослідження основних вразливостей	25.02.2022 – 24.03.2022	*виконано*
5	Вибір методів захисту	25.03.2022 – 07.04.2022	*виконано*
6	Вибір методів моніторингу	08.04.2022 – 20.04.2022	*виконано*
7	Впровадження засобів моніторингу	21.04.2022 – 05.05.2022	*виконано*
8	Впровадження засобів захисту	06.05.2022 – 20.05.2022	*виконано*
9	Формування рекомендацій щодо використання	21.05.2022 – 04.06.2022	*виконано*
10	Оформлення пояснювальної записки	05.06.2022 – 08.06.2022	*виконано*
11	Підготовка до захисту	09.06.2022 – 10.06.2022	*виконано*

Завдання видав	_____	Микола БРАІЛОВСЬКИЙ
	(підпис)	(ініціали, прізвище)
Завдання прийняв до виконання	_____	Микита МИРОВЕЦЬ
	(підпис)	(ініціали, прізвище)

Термін подання дипломної роботи до ЕК 08 червня 2022 року

РЕФЕРАТ

Пояснювальна записка дипломної роботи складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел. Основний текст займає 55 сторінок, включає в себе зміст, вступ, три розділи дипломної роботи, висновки та список джерел. У пояснювальній записці дипломної роботи міститься 16 рисунків.

Метою роботи «Механізм обміну даними між менеджерами паролей» є отримання механізму безпечного обміну між менеджерами паролів.

Для досягнення зазначеної мети поставлено наступні завдання:

- дослідити існуючі засоби аутентифікації;
- дослідити скомпрометовані логіни та паролі;
- сформувати загальні рекомендації щодо створення надійного паролю;
- створити програму для вирішення проблеми експорту даних з менеджера паролів.

Об'єктом дослідження є процес експорту між менеджерами паролів.

Предметом дослідження є експорт даних між менеджерами паролів.

Практичною цінністю отриманих результатів *є* створене програмне забезпечення, що вирішує проблему безпечного експорту даних між менеджерами паролів.

Ключові слова*:* засоби аутентифікації, паролі, менеджер паролів, шифрування, хеш-функції, персональні дані.

При виконанні даної бакалаврської роботи були задіяні наступні **методи дослідження**:

- спостереження;
- порівняння;
- аналіз;
- метод індукції.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

AES – Advanced Encryption Standard
RSA – Rivers, Shamir, Adleman
DES – Data Encryption Standard
SSL – Secure Socket Layer
ІБ – Інформаційна безпека
ІТ – Інформаційні технології
КІ – Конфіденційна інформація
ПД – Персональні дані

ЗМІСТ

ВСТУП

Актуальність. Станом на сьогодні, життєдіяльність кожної людини у тій чи іншій мірі пов’язана з інформаційними технологіями (далі - ІТ), а у деяких навіть залежить від них. Тому проблема захисту персональних даних (далі – ПД) є зараз актуальною, як ніколи. Але при цьому, чи достатньо уваги ми приділяємо безпеці у кіберпросторі? Чи повністю розуміємо ми ризики, нехтуючи своєю безпекою? Й справді зараз, згідно з сучасними трендами, не можна розділяти безпеку в кіберпросторі та безпеку у реальному житті. А тому потрібно забезпечити себе та свої ПД оптимальним рівнем захисту. Одним з базових елементів захисту є використання надійних паролів.

Середній користувач має приблизно 27 сервісів, якими він користується щодня. А тому, для кожного такого сервісу він має використовувати надійну пару логін-пароль. Звісно це складає деяку проблему, щоб створити та запам’ятати їх усі. Саме для таких випадків існують менеджери паролів.

Проте, навіть у таких сервісів є деякі недоліки. Одним з котрих є складність безпечно експортувати дані з одного вендору на інший.

Таким чином, створення механізму безпечного обміну даними між менеджерами паролів є **актуальною задачею**.

Метою дипломної роботи «Механізм обміну даними між менеджерами паролей» є отримання механізму безпечного обміну між менеджерами паролів.

Для досягнення зазначеної мети були поставлені наступні **завдання дипломної роботи:**

- дослідити існуючі засоби аутентифікації;
- дослідити скомпрометовані логіни та паролі;
- сформувати загальні рекомендації щодо створення надійного паролю;
- створити програму для вирішення проблеми експорту даних з менеджера паролів.

Об’єкт дослідження – процес експорту між менеджерами паролів.

Предмет дослідження – експорт даних між менеджерами паролів.

При виконанні даної бакалаврської роботи були задіяні наступні **методи дослідження**:

- спостереження;
- порівняння;
- аналіз;
- метод індукції.

Практичною цінністю отриманих результатів є створене програмне забезпечення, що вирішує проблему безпечного експорту даних між менеджерами паролів.

РОЗДІЛ 1
ПРОБЛЕМА БЕЗПЕКИ ПЕРСОНАЛЬНИХ ДАНИХ

1.1 Завдання розділу 1

ПД є нині дуже важливим ресурсом для сучасної людини. Тому треба докладати зусиль для того, щоб вони залишались конфіденційними. Є багато засобів для надання ІБ. Одним з таких є засоби аутентифікації.

Саме вони надають можливість користувачеві отримати доступ до ПД. А тому слід використовувати лише сучасні й надійні засоби.

У цьому розділі мають бути виконані наступні завдання:

- проаналізувати існуючи методи аутентифікації.
- виділити оптимальний засіб аутентифікації з точки зору звичайного користувача.
- проаналізувати скомпрометовані паролі.
- виділити загальні рекомендації щодо створення паролю, зважаючи на залежність складності до швидкості для підбору паролю.

1.2 Отримання доступу до персональних даних

У загальному випаду для отримання доступу до ПД, користувач повинен пройти 3 процедури:

1. Ідентифікація;
2. Автентифікація;
3. Авторизація.

Ідентифікація - процедура розпізнавання користувача в системі, як правило, за допомогою наперед визначеного імені (ідентифікатора) або іншої апріорної інформації про нього, яка сприймається системою. У більшості випадків мається на увазі логін, який перевіряється на наявність у системі [1].

Автентифікація - процедура встановлення належності користувачеві інформації в системі пред'явленого ним ідентифікатора. Іншими словами відбувається перевірка на відповідність паролю до логіну.

Також є доволі популярним засобом додаткового захисту двохфакторна автентифікація, яка полягає в тому, що при коректно введеному паролю, відбувається додаткова перевірка. Зазвичай додаткове підтвердження відбувається за рахунок номеру мобільного телефону, на який надсилається пароль [2].

Авторизація - це механізм захисту, який використовується для визначення привілеїв користувача / клієнта або рівнів доступу, пов'язаних із системними ресурсами, включаючи комп'ютерні програми, файли, послуги, дані та функції додатків [3].

1.3 Засоби автентифікації

В IT використовуються такі методи аутентифікації:

1. Однобічна аутентифікація, коли клієнт системи для доступу до інформації доводить свою аутентичність;

2. Двобічна аутентифікація, коли, крім клієнта, свою аутентичність повинна підтверджувати і система (наприклад, банк);

3. Трибічна аутентифікація, коли використовується так звана нотаріальна служба аутентифікації для підтвердження достовірності кожного з партнерів в обміні інформацією.

Методи аутентифікації також умовно можна поділити на однофакторні та двофакторні.

Однофакторні методи діляться на:

1. Логічні

2. Ідентифікаційні

3. Біометричні

Двофакторні методи аутентифікації отримуються у результаті комбінації двох різних однофакторних методів, частіше всього ідентифікаційного та логічного.

Наприклад: «пароль + дискета», «магнітна карта + PIN».

Кожен клас методів має свої переваги і недоліки. Майже всі методи аутентифікації страждають на один недолік - вони, насправді, аутентифікують не конкретного суб'єкта, а лише фіксують той факт, що аутентифікатор суб'єкта відповідає його ідентифікатору. Тобто всі відомі методи не захищені від компрометації аутентифікатора [4].

До логічних методів аутентифікації відносяться паролі та ключові фрази, які вводяться з клавіатури комп'ютера чи клавіатури спеціалізованого пристрою [4].

Ще не дуже давно логічна ідентифікація була чи ледве не єдиним способом визначення особистості користувача. Справа в тому, що вона є найбільш простою як у реалізації, так й у використанні.

Суть її зводиться до наступного: кожен зареєстрований користувач системи одержує набір персональних реквізитів (звичайно використаються пари: логін-пароль). Далі при кожній спробі входу людина повинна вказати свою інформацію. Оскільки вона унікальна для кожного користувача, то на цій підставі система й ідентифікує [5].

До головних переваг можна віднести низьку вартість такої реалізації та при правильному використанні, непогане відношення ціна/якість.

Недоліком парольної ідентифікації є значна залежність надійності ідентифікації від користувачів, точніше від обраних ними паролів.

Ідентифікаційні методи аутентифікації ґрунтується на визначенні особистості користувача за певним предметом, ключем, що перебуває в його ексклюзивному користуванні. Мова йде про спеціальні електронні ключі. На даний момент найбільше поширення одержали два типи пристроїв.

До першого відносяться карти. їх досить багато, і працюють вони за різними принципами. Так, наприклад, досить зручні у використанні безконтактні карти, які дозволяють користувачам проходити аутентифікацію як у комп'ютерних системах, так й у системах доступу приміщень.

Іншим типом ключів, які можуть використатися для апаратної ідентифікації, є так звані токени. Ці пристрої мають власну захищену пам'ять і підключаються

безпосередньо до одного з портів комп'ютера.

Головною перевагою застосування ідентифікаційної аутентифікації є досить висока надійність. Адже у пам’яті токенів може зберігатися досить складний ключ, який не вдасться швидко підібрати.

Недоліком є доволі висока ціна, адже потрібно кожному користувачу мати свій персональний токен для вдалої аутентифікації та подальшою авторизації [5].

Біометричні засоби аутентифікації полягають у використанні ідентифікації та аутентифікації людини за рахунок її унікальних, властивих тільки їй, біологічним ознакам. Причому для найпоширеніших з них (відбитки пальців і райдужна оболонка ока) існує безліч різних за принципом дії сканерів [4].

Головною перевагою біометричних технологій є найвища надійність. І дійсно, усі знають, що двох людей з однаковими відбитками пальців у природі просто не існує.

Основним недоліком біометричної ідентифікації є вартість устаткування. Адже для кожного комп’ютера, що входять до цієї системи, необхідно придбати власний сканер.

Двохфакторна аутентифікація – гібрид логічної аутентифікації та ідентифікаційної, адже після вводу коректного паролю, що відповідає логіну, система безпеки задля покращення рівня захищеності запитує доступ до пристрою, доступ до якого може бути лише у користувача з таким логіном та паролем. В більшості випадків мова йде про мобільний телефон.

Отже, кожен з методів аутентифікації має свої переваги та недоліки, проте неможливо точно сказати, що один з них є найкращим за всіма показниками. Інакше існував би лише один з них. Тому визначення кращого можна зробити лише за умови використання їх у специфічних умовах, де, наприклад, недоліки одного нівелюються і тим самим він є безперечним варіантом для використання.

Однак, якщо розглядати захист ПД з точки зору користувача, то можна зробити висновок, що логічна аутентифікація є кращою. Адже така реалізація дешева у реалізації у порівнянні з іншими двома. Та за рахунок правильного використання здобувається високий рівень ІБ.

1.4. Пароль, як спосіб захисту від несанкціонованого доступу.

Пароль – це довільний набір знаків, що складається з літер, цифр та інших символів, що призначена для підтвердження особи або її прав. Він використовується для захисту інформації від несанкціонованого доступу, а саме від доступу осіб, що не мають права доступу [6].

Його іноді ще називають парольною фразою, коли в паролі використовується більше одного слова, або кодом, коли в паролі використовуються лише цифри, такі як персональний ідентифікаційний номер (PIN).

Пароль – це простий засіб автентифікації, що працює, як виклик -відповідь. Для задоволення запиту на виклик використовується усний, письмовий або набраний код. Складність та надійність визначається за рахунок порядку та різноманітності символів.

Статичний пароль - пароль, що використовується для ідентифікації та подальшої авторизації. Як вже зазначено у самій назві, то його суть полягає в тому, що він з часом не змінюється, а значить, що його можна використовувати багато разів. Зазвичай він створюється самим користувачем.

Виходячи з вищезгаданих пунктів, при створенні та зберіганні потрібно надати доволі багато уваги його безпеці, бо від цього залежить конфіденційність інформації. Очевидним слабким місцем у порівняні з динамічними паролями – є багаторазове використання, що значно зменшує захищеність.

Очевидні слабкі місця систем статичного пароля з незашифрованою передачею. Простого списку деяких найбільш очевидних прикладів достатньо, щоб показати легкість, з якою зловмисник може отримати пароль.

Ці методи включають:

- атака за словником: це використовується для отримання "слабких" паролів. Це паролі, які користувачі можуть легко запам'ятати, і з цієї причини вони широко використовуються. Ця форма нападу враховує незначні зміни в режимах словника, такі як додавання цифри або великої літери посередині слова;

- атака соціальної інженерії: зловмисник намагається вгадати пароль на основі

особистої інформації користувача (дата народження, імена дітей чи домашніх тварин, улюблені види спорту тощо);

- підслуховування: "прослуховуючи" мережу, зловмисник отримує пароль користувача, а потім може встановити особу жертви, заповнивши форму автентифікації;

- фішинг - цей метод полягає у надсиланні шахрайських електронних листів, що імітують визнану установу чи компанію, наприклад банк, із запрошенням користувачеві надати дані для входу через веб-сайт зловмисника (який також імітує веб-сайт компанії), нібито для повторної активації або розблокування особистого облікового запису.

Задля протидії подібним атакам на більшості сучасних веб-сайтів використовуються незначні контрзаходи від деяких з цих атак. Для захисту від атаки за словником сайтами використовується обмежена кількість спроб аутентифікації за певний проміжок часу.

Нажаль, у такої контрміри є невеличкий недолік: при багаторазових невдалих спробах аутентифікації користувач не може аутентифікувтися, навіть якщо невдалими були не його спроби.

Крім того, сучасні веб-сайти більш-менш точно вимірюють рівень ентропії паролю користувача, тобто рівень складності паролю, яку складає зловмисникам для взлому пароля методом грубої сили. Паролі, що вважаються простими, як правило, не приймаються.

Однак користувачам нелегко запам’ятати складні паролі. З цієї причини користувачі часто зберігають паролі у своєму браузері, який є незахищеним середовищем. Це також може спричинити інші проблеми: наприклад, коли користувач змінює обладнання, він може втратити доступ до одного або кількох облікових записів, для яких він не зміг запам’ятати пароль.

Взагалі кажучи, єдиною перевагою аутентифікації статичного пароля є передбачувана простота реалізації. Це вірно лише частково; хоча технічно він простий у реалізації, певна складність передається користувачеві, який відповідає за безпеку своїх паролів особливо, якщо їх доволі багато [7].

Динамічний пароль – пароль, який має обмеження на кількість використань для аутентифікації, зазвичай лише один раз. Його функціонування зазвичай обмежується певною кількістю часу, загалом це 24 години.

Використання динамічного паролю обумовлене підвищенням рівня захищеності, адже його неможливо використовувати повторно, тому навіть за умови, що зловмисник його перехопить, то це не дає йому майже ніяких можливостей для успішного використання при подальших спробах аутентифікації. Але це справедливо лише за умови, за якої зловмисник є другою за рахунком персоною, що його використовує.

Логічним недоліком такого паролю є необхідність у постійній його зміні на новий. Постійна зміна призводить до незручностей для користувача, адже йому кожен раз потрібно запам’ятовувати оновлений пароль після кожної успішної аутентифікації.

Динамічний пароль у порівнянні з статичним є більш безпечним, але у той же час значно менш зручніший у повсякденному використанні, тому його основний недолік було виправлено за рахунок зміни способу використання. Більш того, динамічні паролі використовуються буквально кожен день рядовими користувачами, наприклад, при двохфакторній аутентифікації.

Прикладом може служити спроба аутентифікуватися на сайті чи додатку майже будь-якого банку. Після успішного введення логіну та паролю на номер мобільного телефону, який зареєстрований, як телефон особи, що намагається увійти до свого профілю, висилається код з 4-6 символів. Цей код підтверджує вашу особу та являє собою динамічний пароль, адже він є одноразовим.

Динамічні паролі дійсно створюють велику перешкоду для зловмисника для заполучення вашої конфіденційної інформації (далі – КІ).

Слід зазначити, що зараз двохфакторна аутентифікація використовується майже на всіх сервісах, що можуть зберігати важливу інформацію користувача

Тому, зважаючи на простоту використання та рівень ІБ, що надає двохфакторна аутентифікація, вона має бути використана всюди, де конфіденційність інформації є ключовим.

Основна відмінність динамічного паролю від статичного – неможливість повторного використання динамічного. Цей параметр їх відрізняє і тим самим дає деякі переваги та створює незручності для користувача.

Як було зазначено вище, динамічний пароль вважається безпечнішим, оскільки на кожну спробу аутентифікації надається лише один пароль, а це значить, що навіть, якщо зловмисник отримає цей пароль після того, як користувач його використав, то це не дає йому ніяких подальших переваг, адже використати він його не може. А обрахувати яким буде наступним – не тривіальна задача, адже у більшості випадків одноразові паролі формуються з випадкових символів сервісом або програмою, і рішення, яке працювало з модифікованими паролями, які створює людина, швидше за все не спрацює.

Не дивлячись на те, що такий режим роботи є більш безпечнішим, він створює невеликі перешкоди для користувача. Адже запам’ятовувати такий пароль немає ніякого сенсу, тому кожен раз потрібно вводити новий, який користувач бачить перший раз. Також слід зазначити, що у більшості випадків користувач отримує такий пароль через мережу Інтернет. То з цього випливає наступний недолік – неможливість аутентифікації при відсутньому з’єднанні.

Тому можна зробити висновок, що одноразовий пароль є сенс використовувати як додатковий захист при роботі з сервісами, що містять важливу інформацію.

1.5 Персональні дані

Згідно з Законом України «Про захист персональних даних», ПД - відомості чи сукупність відомостей про фізичну особу, яка ідентифікована або може бути конкретно ідентифікована; суб'єкт ПД - фізична особа, ПД якої обробляються [8].

Деякі дані та інформація, що зберігаються на комп’ютері, є особистими та мають бути конфіденційними. Люди хочуть зберігати свою зарплату, банківські реквізити та медичні записи в таємниці та подалі від будь-кого. Якщо хтось, хто не має права бачити ці відомості, може отримати доступ без дозволу, це є

несанкціонованим доступом. Закон про захист даних встановлює правила для запобігання цьому [9].

З цього випливає, що логін та пароль є ПД, адже вони дають змогу ідентифікувати особу, наприклад, на якомусь сайті або у якому-то сервісі.

Отже, важливість збереження ПД у безпеці не треба недооцінювати, інакше будь-хто тоді зможе ідентифікувати себе, як людину, за яку він хоче себе видавати. Та таким чином скористатися можливостями, що надає той чи інший сервіс.

Нажаль, рядові користувачі не приділяють достатньої уваги безпеці їх ПД, що призводить до крадіжок з їх банківських рахунків, витокам їх ПД до мережі або конфіденційних даних компанії, якій працює особа. І одною з найпоширеніших причин є прості логіни та паролі.

За онлайн опитуванням, яке було проведене серед 2000 англомовних дорослих людей у 2016 році, було виявлено, що у звичайного середньостатистичного користувача є 27 облікових записів, які використовуються кожен день. У той же час кількість унікальних паролів у середнього користувача набагато менша.

Цей факт дозволяє розраховувати на те, що один із паролів (або його модифікація) підійде для розшифровки файлів, лобова атака на захист яких буде надзвичайно ресурсомісткою. Насправді близько 60% користувачів використовує набір з тих самих паролів для захисту різних ресурсів. Якщо ж рахувати і тих користувачів, які варіюють свої паролі в мінімальних межах (наприклад, використовуючи варіації розряду password<>Password<>Password1<>password1967), їх число досягає 70%.

«Сама кількість облікових записів різко зросла за останні кілька років», — сказав Брюс Снелл, директор із кібербезпеки та конфіденційності Intel Security [10].

З цього виникає проблема, що запам’ятати 27 унікальних, стійких з погляду криптографії пароля – завдання не для середнього користувача. Склалася ситуація, у якій кількість облікових записів (та інших ресурсів, доступ яких захищається у вигляді паролів) в рази перевищує кількість паролів, які може запам'ятати користувач. У період з 2012 по 2016 рік було проведено низку досліджень, які

однозначно показали, що 59-61% користувачів використовує одні й ті самі паролі для захисту різних ресурсів [11].

Злом паролю є одним з найпоширеніших типів атак на сервіси, які використовують аутентифікацію за паролями або за допомогою пари логін-пароль. Кінцевим результатом атаки є отримання зловмисником паролю користувача.

Цікавість такої атаки для зловмисника полягає в тому, що при успішному отриманні пароля він гарантовано отримує всі права користувача, обліковий запис, який був скомпрометований, крім того, вхід під існуючою облікового запису зазвичай викликає менше підозр у системних адміністраторів.

Технічно атака може бути реалізована двома способами: багаторазовими спробами прямої аутентифікації в системі, або аналізом хешів паролів, отриманих іншим способом, наприклад перехопленням трафіку.

При цьому можуть бути використані наступні підходи:

1. Прямий перебір. Перебір всіх можливих поєднань символів допустимих в паролі.

2. Підбір за словником. Метод заснований на припущенні, що в паролі використовуються існуючі слова будь-якої мови або їх поєднання [13].

3. Виходячи з підходів до проведення атаки, можна сформулювати критерії стійкості пароля до неї.

4. Пароль не повинен бути надто коротким, оскільки це спрощує його злом повним перебором. Найбільш поширена мінімальна довжина – вісім символів. З тієї ж причини не повинен складатися з одних цифр.

5. Пароль не повинен бути словниковим словом або простим поєднанням, це спрощує його підбір за словником.

6. Пароль не повинен складатися лише із загальнодоступної інформації про користувача.

7. Як рекомендацію до складання паролів можна назвати використання поєднання слів з цифрами та спеціальними символами (#, $, * і т.д.), використання малопоширених або неіснуючих слів, дотримання мінімальної довжини.

Як вже було зазначено вище, то середній користувач має як мінімум 27

облікових записів. Серед цих облікових записів з високою вірогідністю виявляться і такі, що колись були зламані. LinkedIn, eBay, Twitter, Dropbox - всі ці сервіси об'єднує те, що в різні роки хакери витягували з них облікові дані користувачів, включаючи паролі. Mark Burnett проаналізував, зібравши всі відомі витоки воєдино і обробивши дані. Марк відсортував усі паролі, що були знайдені у мережі у відкритому доступі, за популярністю і склав список з 10,000 найбільш поширених паролів [12].

Аналіз паролів із витоків дозволив виявити цікаві закономірності у поведінці англомовних користувачів:

- 0.5% користувачів використовують слово password як пароль;
- 0.4% користувачів як пароль використовують послідовності password або 123456;
- 0.9% використовують password, 123456 або 12345678;
- 1.6% використовують пароль із десятки найпоширеніших (top-10);
- 4.4% використовують пароль із першої сотні (top-100);
- 9.7% використовують пароль із top-500;
- 13.2% використовують із top-1,000;
- 30% використовують із top -10,000.

Як приклад, можна також навести дослідження скомпрометованих пар логінів-паролів, які були використовувалися або використовуються на сервісі електронної пошти, що використовується на території пострадянських країн. Тому саме з цієї причини у більшості пар використовується кирилиця.

Файл з скомпрометованими парами з’явився на просторі Інтернет в 2015 році. Він містив 25929527 пар логінів-паролей. Значна кількість з яких виявилися навіть дійсними.

Отже, при аналізі цих пар було виявлено, що створюючи список з топ 50 паролів, ймовірність підібрати вірний пароль до логіну дорівнює 6,98% (а це лише 0,00054% від загальної суми), а при використанні топ 100, ймовірність дорівнює 8,72%. З цього можна зробити висновок, що при збільшенні часу в 2 рази на метод грубої сили ефективність зростає лише на 1,74%.

Також було виявлено, що більшість користувачів для того, щоб «збільшити» рівень захищеності, у кінці самого паролю додають такі закінчення, як: «123» або «1». Це призводить до того, що при невдалій спробі підібрати пароль навіть з топ 10000, достатньо лише використати паролі з топ 100 та на кінці додати «1» або «123». Також при складанні словника паролів із закінченнями слід враховувати, що закінчення найчастіше додають до словникових паролів, наприклад "password123", "qwerty1", але рідко "19411945123". У разі не завжди навіть буде зрозуміло, що це «закінчення», а не «корінь» пароля [14].

Подібні витоки паролів надають зловмисникам можливість створювати списки з найпопулярніших паролів. А надалі ці списки використовувати для атаки на пароль за словником, що значно пришвидшує підбор паролю.

Проте, також слід зазначити, що зловмисник може й використовувати звичайний метод грубої сили. В таблиці 1.1 представлено залежність часу від складності паролю, використовуючи метод грубої сили.

Таблиця 1.1

Залежність часу підбору паролю від його складності

Кількість символів	Тільки числа	Літери у нижньому регістрі	Літери у верхньому регістрі	Літери та цифри у нижньому та верхньому регістрах	Літери та цифри у нижньому та верхньому регістрах та спецсимволи
4	Миттєво	Миттєво	Миттєво	Миттєво	Миттєво
5	Миттєво	Миттєво	Миттєво	Миттєво	Миттєво
6	Миттєво	Миттєво	Миттєво	Миттєво	Миттєво
7	Миттєво	Миттєво	2 с	7 с	31 с
8	Миттєво	Миттєво	2 хв	7 хв	39 хв
9	Миттєво	10 с	1 год	7 год	2 дні
10	Миттєво	4 хв	3 дні	3 тижні	5 міс
11	Миттєво	2 год	5 міс	3 роки	34 роки
12	2 с	2 дні	24 роки	200 років	3 тис років
13	19 с	2 міс	1 тис. років	12 тис. років	202 тис років
14	3 хв	4 роки	64 тис. років	750 тис. років	16 млн років
15	32 хв	100 років	3 млн. років	46 млн. років	1 млрд. років
16	5 год	3 тис. років	173 млн. років	3 млрд. років	92 млрд. років
17	2 дні	69 тис. років	9 млрд. років	179 млрд. років	7 трлн. років
18	3 тижні	2 млн. років	467 млрд. років	11 трлн. років	438 трлн. років

Виходячи з цієї таблиці, можна зробити висновок, що для складання оптимального паролю потрібно 11-12 символів, що складаються з цифр, літер (верхнього та нижнього регістрів) та можливо спец символів, але зважаючи, що навіть без спец символів на підбор паролю буде витрачено дуже багато часу, то більшість користувачів з високою імовірністю знехтують цим. Проте не все так просто, як здається.

Перш за все, потрібно розібрати як це все працює та як такі обчислення робляться. Дані обчислення виконувалися на графічному процесорі RTX 3090,який є одним з найпотужніших на 2022 рік серед відеокарт, призначених для споживчих цілей.

У контексті паролів «хеш» — це зашифрована версія тексту, яку можна відтворити, якщо ви знаєте, яке програмне забезпечення хешування було використано. Іншими словами, якщо хешується слово «password» за допомогою програмного забезпечення для хешування MD5, вихідний хеш буде 5f4dcc3b5aa765d61d8327deb882cf99. Тепер, якщо хешується слово «password» за допомогою програмного забезпечення для хешування MD5, ви також отримаєте 5f4dcc3b5aa765d61d8327deb882cf99. Ми знаємо, що секретним словом є “password”, але будь-хто інший бачить лише 5f4dcc3b5aa765d61d8327deb882cf99. З цієї причини паролі, які використовуються на веб-сайтах, зберігаються на серверах у вигляді хешів, а не у вигляді простого тексту, наприклад «password», тому, якщо хтось перегляне їх, теоретично вони не дізнаються фактичного пароля [15].

Так як хеш функція – одностороння функція, то з хешу неможливо вичислити, які символи були захешовані.

Вони вирішують цю проблему, зламом самого паролю. У цьому контексті мається на увазі, що хакер створює список всіх можливих комбінацій символів, які можна ввести з клавіатури, а потім їх хешує. Таким чином, порівнюючи список хешів, які отримуються перебором усіх можливих комбінацій та подальшим хешуванням, вони отримують пароль уже у вигляді відкритого тексту.

Такі обчислення виконуються за рахунок відеокарти, то ж швидкість виконання обчислення хеш функції залежить лише від продуктивності тієї чи іншої

відеокарти. В таблиці 1.2 наведено результати 2020 року, обчислювання виконувалися на графічному процесорі RTX 2080, який був на той момент одним з найпродуктивніших.

В таблиці 1.3 наведено порівняння продуктивності актуальних графічних процесорів.

Таблиця 1.2

Залежність часу підбору паролю від його складності

	Тільки числа	Літери у нижньому регістрі	Літери у верхньому регістрі та нижньому	Літери та цифри у нижньому та верхньому регістрах	Літери та цифри у нижньому та верхньому регістрах та спецсимволи
4	Миттєво	Миттєво	Миттєво	Миттєво	Миттєво
5	Миттєво	Миттєво	Миттєво	Миттєво	Миттєво
6	Миттєво	Миттєво	Миттєво	1 сек	5 сек
7	Миттєво	Миттєво	25 сек	1 хв	6хв
8	Миттєво	5 сек	22 хв	1 год	8 год
9	Миттєво	2 хв	19 год	3 дні	3 тижні
10	Миттєво	58 хв	1 міс	7 міс	5 років
11	2 сек	1 день	5 років	41 рік	400 років
12	25 сек	3 тижні	300 років	2 тис років	34 тис років
13	4 хв	1 рік	16 тис років	100 тис років	2 млн років
14	41 хв	51 рік	800 тис років	9 млн років	200 млн років
15	6 год	1 тис років	43 млн років	600 млн років	15 млрд років
16	2 дні	34 тис років	2 млрд років	37 млрд років	1 трлн років
17	4 тижні	800 тис років	100 млрд років	2 трлн років	93 трлн років
18	9 міс	23 млн років	61 трлн років	100 трлн років	7 квдрлн

Таблиця 1.3

Порівняння продуктивності

MD5	Обчислення за секунду	Хешів за секнуду
RTX 2080	10,070,000,000,000	37,085,000,000
RTX 3090	35,580,000,000,000	69,379,700,000

Будь то відеокарта чи хмарний сервіс, вони у будь-якому випадку мають мають якусь продуктивність, яка вимірюється у обчисленнях за секунду. Під обчисленнями мається на увазі операції з плаваючою комою (FLOPS). Дані про швидкість обчислень хешу за секунду були узяті з сервісу hashcat.

Графічний процесор RTX 2080 обчислював близько 37 085 мільйонів хешів в секунду (MH/s). Порівнюючи результати двох відеокарт, різниця між якими лише 2 роки, отримуємо наступні результати:

- від 10 трильйонів FLOPS до 35 трильйонів FLOPS лише за 2 роки, ~250% збільшення необробленої обчислювальної потужності;
- від 37 GH/s до 69 GH/s за 2 роки - збільшення хешів за секунду на ~86%.

Виходячи з цих результатів та даних таблиць, можна помітити, що на злам паролю, складність якого визначена стандартом NIST. А саме 8 символів та більше, то лише 2 роки тому на злом потрібно було 8 годин, а зараз лише 5. 5 годин це доволі багато, але чи можливо якимось чином прискорити це?

Звичайно, для цього можна використати потужність високопродуктивних обчислювальних кластерів Amazon. На даний момент Amazon пропонує оренду 8 графічних процесорів NVIDIA A100 Tensor Core через свою пропозицію EC2 P4d під назвою «p4d.24xlarge» , яка рекламується як «найвища продуктивність для програм навчання машинного навчання та HPC у хмарі», всього за 32,77 доларів США на годину. В таблиці 1.4 наведено продуктивність 8x NVIDIA A100 Tensor Core.

Таблиця 1.4

Залежність часу підбору паролю від його складності

Кількість символів	Тільки числа	Літери у нижньому регістрі	Літери у верхньому регістрі та нижньому	Літери та цифри у нижньому та верхньому регістрах	Літери та цифри у нижньому та верхньому регістрах та спецсимволи
4	Миттєво	Миттєво	Миттєво	Миттєво	Миттєво
5	Миттєво	Миттєво	Миттєво	Миттєво	Миттєво
6	Миттєво	Миттєво	Миттєво	Миттєво	Миттєво
7	Миттєво	Миттєво	2 с	7 с	31 с

Продовження таблиці 1.4

8	Миттєво	Миттєво	2 хв	7 хв	39 хв
9	Миттєво	10 с	1 год	7 год	2 дні
10	Миттєво	4 хв	3 дні	3 тижні	5 міс
11	Миттєво	2 год	5 міс	3 роки	34 роки
12	2 с	2 дні	24 год	200 років	3 тис років
13	19 с	2 міс	1 тис років	12 тис років	202 тис років
14	3 хв	4 роки	64 тис років	750 тис років	16 млн років
15	32 хв	100 років	3 млн років	46 млн років	1 млрд років
16	5 год	3 тис років	173 млн років	3 млрд років	92 млрд років
17	2 дні	69 тис років	9 млрд років	179 млрд років	7 трлн років
18	3 тижні	2 млн років	467 млрд років	11 трлн років	438 трлн років

За умови, що хакер не хоче витрачати так багато грошей та бажає уникнути використання великої корпоративної мережі, він може звернутися до інших сервісів. Наприклад, сервіс vast.ai, що дозволяє звичайним користувачам здавати в оренду своє комп’ютерне обладнання через підключення до Інтернету.

На даний момент одною з найефективніших була збірка з восьми RTX 3090s та, у порівнянні з Amazon, за дешевшою ціною. Приблизно 3.20$ - 5.60$ за годину. В таблиці 1.5 наведено порівняння двох збірок.

Таблиця 1.5

Порівняння продуктивності

MD5	Обчислення за секунду	Хешів за секунду
RTX 2080	10,070,000,000,000	37,085,000,000
8 x RTX 3090s	355,200,000,000,000	555,000,000,000

Проте слід зазначити, що з часом окрім підвищення потужності графічних процесорів, також і вдосконалюються засоби захисту. Наприклад, використовуються так звані солі при хешуванні. Та ускладняються самі функції хешування. На даний момент хеш-функція MD5 вважається застарілою[15].

Висновки до розділу 1

Отже, ПД є важливим ресурсом для сучасної людини, тому кожен з нас потребує достатнього рівня ІБ. Одним з методів надання безпеки є використання сучасних методів аутентифікації.

У ході виконання роботи було проаналізовано існуючі методи аутентифікації та було визначено, що оптимальним засобом аутентифікації є саме логічна. Вона полягає у використанні паролів та парольних фраз для аутентифікації.

Для підвищення рівня обізнаності були розглянуті та проаналізовані скомпрометовані паролі, що опинилися у мережі Інтернет за рахунок витоку. На основі аналізу скомпрометованих паролів та залежності швидкості підбору до складності були визначені загальні рекомендації для створення надійного паролю.

РОЗДІЛ 2
МЕНЕДЖЕР ПАРОЛІВ, ЯК ЗАСІБ НАДАННЯ ОПТИМАЛЬНОГО РІВНЯ БЕЗПЕКИ ДЛЯ ЗБЕРІГАННЯ ІНФОРМАЦІЇ

2.1 Завдання розділу 2

У минулому розділі було визначено, що оптимальним засобом аутентифікації з метою надання достатнього рівня безпеки для рядового користувача є логічна аутентифікація. При логічній аутентифікації використовуються паролі. Саме тому були проаналізовані паролі, що витекли. Базуючись на цих знаннях були визначені загальні рекомендації щодо створення оптимального паролю.

Проте, як вже було визначено, запам’ятовування унікального та достатньо складного рівня паролю – не тривіальна задача для рядового користувача. Особливо зважаючи на той, факт, що середня кількість використовуваних щодня сервісів сягає 27.

Тому цей розділ присвячений засобам зберігання даних, таких як менеджер паролів.

Для виконання цього розділу були поставлені наступні завдання:

1. Проаналізувати існуючі засоби зберігання інформації.
2. Проаналізувати принцип роботи менеджера паролів.
3. Проаналізувати актуальні алгоритми шифрування.
4. Ознайомитись з призначенням хеш-функцій.
5. Визначити недоліки, що виникають при використанні менеджерів паролей.

2.2 Засоби зберігання персональних даних.

Ще не так давно, чи не єдиним засобом зберігання інформації були книги, паперові документи тощо. Проте, зараз ринок пропонує безліч варіантів збереження

інформації. Тому вони мають бути проаналізовані та відокремлені кращі.

Незважаючи на багато зовнішніх рішень для цифрових файлів, люди все ще зберігають свої фотографії, відео та файли вмісту на своєму робочому столі або ноутбуці. Єдина проблема цього методу полягає в тому, що комп'ютер може швидко заповнюватися тисячами файлів, що, як наслідок, уповільнює його.

Коли ви хочете знайти цифровий файл, ви, ймовірно, очікуєте, що цей файл миттєво з'явиться на вашому екрані. Тим не менш, будь-хто, хто зберігає багато фотографій на комп'ютері, знає, що пошук однієї може зайняти хвилини, іноді години, навіть якщо ви зберігаєте її на робочому столі. Просто не так зручно зберігати речі таким чином. Найважливіше те, що просто зберігання цих цифрових файлів на робочому столі робить їх уразливими для вірусів, пошкоджень або крадіжки. Люди, які покладаються на це, також зазвичай не мають резервного плану[16].

Варіант цифрового зберігання у хмарному сховищі стає все більш популярним через його численні переваги, незважаючи на деякі недоліки. Такі компанії, як Dropbox, з'явилися, щоб запропонувати варіант для цифрових файлів, які вирішували існуючі проблеми зі зберіганням даних. Незадовго з'явилися Google Drive, Box та численні інші хмарні платформи.

Хмарне сховище усунуло частину потреби зберігати файли на комп'ютерах, оскільки воно пропонує організований спосіб пошуку, обміну та контролю вашої інформації. Такі сервіси пропонують миттєву синхронізацію та доступ з будь-якого пристрою для більшої зручності. Деякі навіть мають історію редакцій, щоб ви могли повернутися до старішої версії цифрового файлу.

Однак навіть у цього рішення для зберігання є свої недоліки. Перш за все, безкоштовно вони надають лише обмежений об'єм пам'яті, який доволі швидко закінчується. Для збільшення об'єму потрібно заплатити за підписку, ціна якої доволі висока. Єдиним варіантом оплати є підписка, що не дає змогу купити на невизначений срок потрібний об'єм, а лише орендувати його на місяць. Після закінчення сроку потрібно оновити оплату, тоді що станеться з даними, якщо підписка буде скасована?

Багато експертів із безпеки висловлювали занепокоєння щодо реальної безпеки за допомогою хмарного сховища. Наприклад, InformationWeek зазначив, що існують деякі ключові загрози, зокрема порушення даних, втрата даних, захоплення трафіку, незахищені API тощо. CSO пояснила, що з кожним роком виникає все більше проблем із безпекою, оскільки хакери усвідомлюють, як вони можуть отримати ці дані в першу чергу під час передачі даних до хмарного постачальника.

Відсутність резервного копіювання спонукала багатьох до вивчення холодного зберігання. Сюди входять окремі накопичувачі, такі як флеш-накопичувачі та флеш-накопичувачі, а також SD-карти та DVD. Однією з переваг є економічно ефективний спосіб відносно безпечного зберігання великих файлів. Файли можна впорядкувати, щоб інформацію можна було легко знайти та поширити – після того, як вона підключена до комп'ютера. Однак є і зворотний бік. Дані фрагментовані та важкодоступні – чи буде доступ до машини з дисководом, коли прийде час запустити DVD?

Крім того, його можна легко втратити або пошкодити. Зовнішні диски часто потребують обслуговування через кілька років, як і будь-яке інше обладнання, що вимагає додаткових витрат на відновлення файлів, якщо вони пошкоджені, і виникає необхідність перенести всю цю інформацію на новий пристрій.

Наразі соціальні мережі займають велику частину життя у великої більшості людей, особливо у молодшого покоління. Люди майже "живуть онлайн", дивлячись на світ через призму їх профілів у соціальних медіа. В соціальних мережах люди спілкуються, знайомляться, дивляться новини, читають статті на теми, що цікаві користувачу тощо. Під час спілкування користувачі також обмінюються даними, такими як: фото, відео, документи, посилання, тощо. Що важливо, то усі ці дані залишаються на серверах соціальних мереж, то ж при поверненню до діалогу їх можна знайти, скачати та використовувати. Таким чином ці цифрові файли не займають місце на пристрої користувача та можуть таким чином зберігатися, як хмарне сховище. Крім того, такі сайти зазвичай дають можливість створювати альбоми для легкого доступу до даних та обміну ними.

Таке рішення дуже схоже на хмарне сховище, проте воно має деякі недоліки. Перш за все потрібно зазначити, що зазвичай фото та відео зберігається у низькій роздільній здатності. Це зроблено для того, щоб швидше підгружалися файли та менше місця займали на сховищах соціальних мереж, проте у такому разі це завдає користувачу деяких незручностей. З іншої сторони, у більшості випадків, можна зробити, що файли примусово зберігалися у високій роздільній здатності.

При випадковому видаленні файлів, нажаль, не має змоги їх повернути. У багатьох випадках компанії соціальних медіа не дають жодних гарантій щодо того, що зберігаються на їхній платформі чи ні.

Іншим важливими недоліком подібного зберігання є незахищеність аккаунтів користувачів, а значить і усіх даних від витоку шляхом взлому. Також слід зазначити, що сайти соціальних медіа оброблюють, зберігають та навіть можуть використовувати ПД користувачів. Наприклад, Facebook заявив, що їм належать усі фотографії користувачів. Навіть за умови видалення їх або всього облікового запису, Facebook має право на їх використання. Наразі є способи змінити це, проте у будь якому випадку мова не йде про безпечне зберігання даних.

Враховуючи вищезгадані пункти, можна зробити висновок, що локальні та хмарні рішення зберігання мають свої переваги та недоліки.

Локальні рішення вважаються більш захищеними, так як вони вимагають від зловмисників складних рішень для отримання НСД, таких як: використання троянських прогам, кейлогерів тощо. В даному випадку пароль зберігається та оброблюється на комп’ютері користувача, а значить, користувач його повністю контролює.

Хмарне зберігання, з точки зору звичайного користувача, надає доступність та зручність. Так як зашифровані паролі зберігаються на хмарних серверах, тим самим це дає змогу для отримання доступу з будь-якого пристрою та відносно легко синхронізувати паролі між пристроями. Вони зберігають зашифровані копії даних на власних серверах. Також слід зазначити, що такі служби виступають гарантами безпеки зберігаємих даних, шифруючи передачу даних між пристроями та серверами та синхронізуючи всі пристрої користувача.

Хмарні рішення здаються менш захищеними, однак згідно з прогнозом Gartner, робочі процеси, що виконуються в загальнодоступній в загальнодоступній хмарі, зазнаватимуть на 60% менше проблем із безпекою, ніж ті, що виконуються в традиційних центрах обробки даних [17].

На відміну від локального сховища, у вас буде команда фахівців із кібербезпеки з різних країн, які безперервно допомагатимуть захистити ресурси й дані вашої компанії.

Однак порушення безпеки можуть виникати так само, як і в локальних сховищах. Компанії, які використовують хмару, можуть мінімізувати такий ризик, застосовуючи процеси приєднання та від'єднання, щоб керувати доступом працівників і вказувати, як та коли вони можуть користуватися зовнішніми програмами.

Здатність компаній поєднати загальнодоступну хмару та локальні дані може збільшити гнучкість IT-інфраструктури й максимально підвищити ефективність. У гібридному рішенні дані й програми можна переміщати між локальними серверами та загальнодоступними хмарами, щоб забезпечити більшу гнучкість і кількість доступних параметрів розгортання. Іншими словами, ви можете зберігати файли, які використовуються бізнес-програмами, на локальних серверах, а решту файлів і документів – у хмарі. Серед інших переваг гібридного рішення:

Контроль. Ваша організація може обслуговувати приватну інфраструктуру для конфіденційних об'єктів.

Гнучкість. Ви можете користуватися додатковими ресурсами в загальнодоступній хмарі, коли вони вам потрібні.

Рентабельність. Завдяки можливості пристосуватися до загальнодоступної хмари ви платите за додаткову обчислювальну потужність, лише коли вона потрібна.

Зручність. Перехід у хмару не повинен бути обтяжливий. Ви можете робити це крок за кроком – поступово впроваджуючи робочі процеси з часом.

Хоча здається, що гібридне рішення поєднує в собі найкращі якості обох варіантів для деяких компаній, воно є тимчасовим.

Сьогодні компанії малого та середнього бізнесу вже виконують 43% своїх робочих процесів у загальнодоступній хмарі, і за прогнозами це число зростатиме. З огляду на це важко ігнорувати переваги повного переходу в хмару для компаній малого та середнього бізнесу з недоукомплектованими й перевантаженими відділами IT [18].

Менеджер паролів - програмне забезпечення, яке допомагає користувачу працювати з логінами та паролями. Вони генерують унікальні надійні паролі та зберігають їх в зашифрованому вигляді. Зазвичай у такого програмного забезпечення є вмонтована база даних або файли, що зберігають зашифровані дані паролів. Більшість з них також виконують функцію заповнювача форм, мається на увазі що вони заповнюють поле логіну та під логін заповнюють відповідний пароль. Зазвичай це реалізовано у вигляді браузера.

2.3 Менеджер паролів, як засіб надання оптимального рівня безпеки

Менеджер паролів - програмне забезпечення, яке допомагає користувачу працювати з логінами та паролями. Вони генерують унікальні надійні паролі та зберігають їх в зашифрованому вигляді. Зазвичай у такого програмного забезпечення є вмонтована база даних або файли, що зберігають зашифровані дані паролів. Більшість з них також виконують функцію заповнювача форм, мається на увазі що вони заповнюють поле логіну та під логін заповнюють відповідний пароль. Зазвичай це реалізовано у вигляді браузера.

Вони поділяються на три основні категорії:

Десктоп - зберігають паролі до програмного забезпечення, що встановлено на жорсткому диску комп’ютеру.

Портативні - зберігають паролі до ПЗ на мобільних пристроях, таких як смартфон, USB-накопичувачі.

Мережеві - менеджери паролів онлайн, де паролі збережені на веб-сайтах провайдерів [19].

Менеджери паролів також можуть використовуватися як захист від фішингу. На відміну від людей, програма менеджер паролів може поводитися з автоматизованим скриптом логіна. Несприйнятлива до візуальних імітацій, які схожі на веб-сайти, тобто, перейшовши за сумнівним посиланням на фішинговий сайт менеджер паролів не підставить логін-пароль у форми введення, а користувач що сайт є підробкою. З цією вбудованою перевагою використання менеджера паролів вигідне, навіть якщо у користувача є кілька паролів, які він пам'ятає. Однак, не всі менеджери паролів можуть автоматично поводитися з більш складними процедурами ідентифікації, накладеними багатьма банківськими веб-сайтами.

Менеджери паролів зазвичай використовують вибраний користувачем основний пароль, або секретну фразу (passphrase), щоб сформувати ключ, який використовується для зашифрування паролів, що зберігаються. Цей основний пароль має бути досить складним, щоб устояти під час атак зловмисників (наприклад, повний перебір).

Якщо основний пароль буде зламаний, то будуть розкриті всі паролі, що зберігаються в базі даних програми. Це демонструє зворотний зв'язок між зручністю використання та безпекою: єдиний пароль може бути зручніший, але якщо він буде зламаний, то поставить під загрозу всі паролі, що зберігаються.

Основний пароль може бути атакований і виявлений при використанні кейлоггера або акустичного криптоаналізу (acoustic cryptanalysis). Така загроза може бути знижена шляхом використання віртуальної клавіатури, як, наприклад, KeePass.

Деякі менеджери паролів включають генератор паролів. Згенеровані паролі можуть бути відгадуваними, якщо менеджер пароля не використовує криптографічно безпечний генератор випадкових чисел [20].

У загальному випадку, робота менеджеру паролів зводиться до шифрування даних, які користувач зберігає (паролів та логінів). Тому коли дані зашифровані, вони перетворюються на шифр, так що тільки ті, хто має правильний «ключ», можуть розшифрувати і прочитати його. Це означає, що якщо хтось колись спробує вкрасти ваші паролі з вашого менеджера паролів, він вкраде інформацію, яку неможливо прочитати або використовувати.

Шифрування є однією з основних функцій безпеки менеджерів паролів і тому їх так безпечно використовувати.

Зберігати свої паролі в записнику було небезпечно, тому що будь-хто міг прочитати інформацію, але шифрування менеджерів паролів гарантує, що тільки ви можете прочитати свої паролі та логіни.

Для забезпечення надійного збереження даних у загальному випадку менеджери паролів опираються на наступні принципи:

- зберігання інформації на сервері у зашифрованому вигляді.
- здійснення процесу шифрування всієї інформації на стороні клієнта
- передача інформації від клієнта серверу та назад лише у зашифрованому вигляді
- застосування як ключ шифрування інформації, введеної користувачем при вході в систему, причому серверу вона ніколи не передається

Завдяки шифруванню на стороні клієнта менеджер паролів гарантує збереження Ваших конфіденційних даних, навіть у разі злому сервера, крадіжки даних з БД, або перехоплення інформації при передачі. На рисунку номер 00 зображена діаграма, що ілюструє роботу менеджеру паролів.

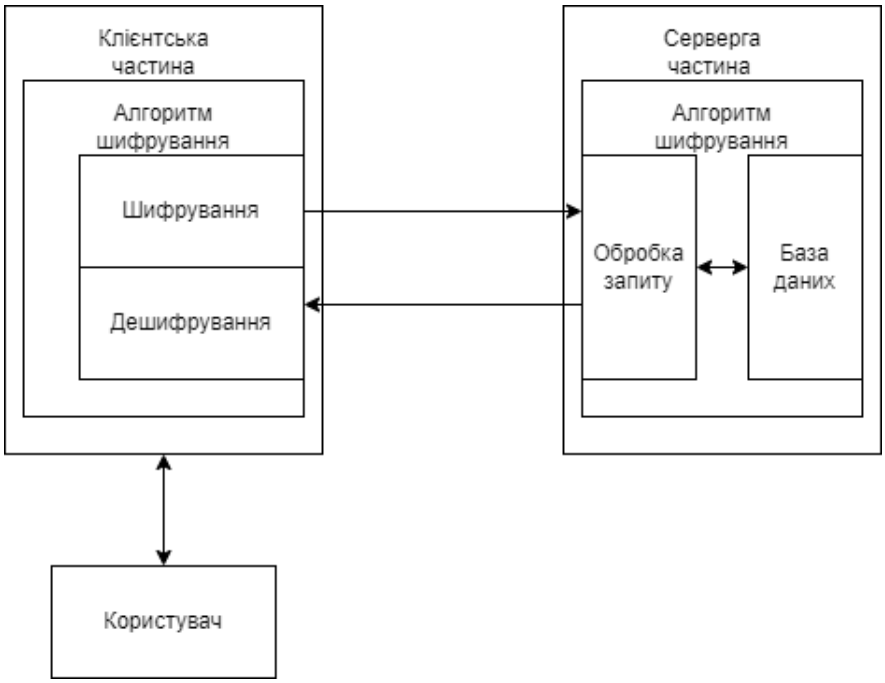

Рисунок 2.1 - Діаграма роботи менеджера паролів

Як це було виявлено у минулому розділі, враховуючи сучасні обчислювальні можливості, пароль має бути доволі складним, щоб гарантувати забезпечення безпеки.

Рядовий користувач з невисокою вірогідністю зможе вигадати унікальний та випадковий пароль, а для забезпечення безпеки потрібен саме такий. Тому у більшості менеджерів паролів є вмонтований генератор випадкових паролів. Він допомагає генерувати надійні паролі, а у деяких випадках навіть запам’ятовуванні.

Справа в тому, що більшість витоків відбувається за рахунок використання слабких або повторно використовуваних паролів, тому випадкові унікальні паролі - найкращий захист від різних загроз онлайн.

Пароль має бути унікальним з тієї причини, що якщо використовується той самий пароль для електронної пошти та для входу в банківський обліковий запис, зловмисник, вкравши тільки один пароль, отримає доступ до обох облікових записів, що означає подвійну вразливість. Якщо використовується один і той же пароль для 14 різних облікових записів, то робота зловмисника значно спрощується.

Пароль має бути випадковим, бо їх складно вгадати та важче зламати комп'ютерними програмами. Якщо є певна закономірність, шанси зловмисника атакувати та отримати доступ до вашого облікового запису зростають у рази. Випадкові паролі або можуть містити набір не пов'язаних символів, або є комбінацією не пов'язаних слів [21].

Незважаючи на те, що менеджер паролей є дуже зручним інструментом для підвищення рівня безпеки, нажаль, він не позбавлений мінусів. Першим недоліком, який одразу попадається на очі можна назвати – не завжди простий інтерфейс. Так, звичайно, це не стосується усіх, що є на ринку, але у більшості випадків це саме й так відбувається. Іноді у користувача бувають такі випадки, коли потрібно увійти в програму, скопіювати пару логін-пароль, а потім перейти назад до попереднього сайту або програми, щоб ввести свої дані. Нажаль, не завжди є можливість повернутися на сайт через плагін менеджера без переходу до налаштувань.

Також було б краще, якби для створення паролю у програмі не потрібно було б робити стільки кроків.

Зважаючи на те, що менеджер паролів використовується для надання оптимального рівня безпеки,

Проте, найбільшою проблемою, на мою думку, є відсутність змоги безпечно експортувати паролі не тільки у рамках одного вендора. Так як кожен виробник програмного забезпечення намагається зробити свій додаток більш безпечнішим, то з цієї причини у загальному випадку використовуються лише 2 способи експорту даних. Першим з яких є використання формату, який приймається лише цим додатком. Такий спосіб є безпечним, проте він не підходить для експорту в додатки від іншого виробника. Другий спосіб полягає у експорті в файл з розширенням .CSV. Цей файл приймається будь-яким менеджером паролів, проте він не безпечним, так як він являє собою перераховані через кому пари логін-пароль.

2.4 Шифрування інформації

Як було зазначено у пункті вище, менеджер паролів зберігає пари логін-пароль у зашифрованому вигляді. Шифрування – це технологія кодування та розкодування даних. Зашифровані дані - це результат застосування алгоритму кодування даних з метою зробити їх недоступними для читання. Дані можуть бути розкодовані у вихідну форму тільки шляхом застосування спеціального ключа. Шифрування є важливою частиною забезпечення безпеки даних, оскільки воно захищає КІ від загроз, серед яких використання шкідливого програмного забезпечення та несанкціонований доступ третіх сторін. Шифрування даних - це універсальне захисне рішення: воно може застосовуватися до частини даних, наприклад, паролю, до інформації у файлі або навіть до всіх даних, що містяться на носії [22].

Якщо необхідно надіслати приватне повідомлення або просто зашифрувати дані від сторонніх очей, то шифрування відбувається за рахунок якоїсь з програм, яка в свою чергу використовує той чи інший алгоритм.

Шифрованим текстом є, наприклад:

«79izXKgSYVN6DrRzX4vaOlNZ9uePCpkW2ZI5R8DWOIl42ZqssrYVyvN74+o»

На перший погляд здається, що це невпорядкований хаос, адже оригінальний текст виглядає наступним чином.

"name,url,username,password, mnogo files»

У якості алгоритму шифрування у даному випадку виступає алгоритм Advanced Encryption Standard (далі - AES).

Для роботи будь-якого алгоритму шифрування потрібен ключ. Від нього залежить кінцевий результат шифрування. За визначенням, ключ шифрування – параметр криптографічної системи. Він використовується для шифрування та дешифрування та накладення та перевірки коду автентифікації повідомлень або електронного цифрового підпису [23].

Загалом алгоритми шифрування поділяються на симетричні та асиметричні. Симетричний алгоритм шифрування вважається найпростішим. Він передбачає використання одного секретного ключа, який може будь-яким за довжиною та вмістом, це залежить лише від конкретного алгоритму шифрування. Ключі шифрування та дешифрування у такому випадку або збігаються, або є похідними один від іншого.

У випадку наявності двох ключів, які не є похідними один від іншого, то це йдеться мова про асиметричний алгоритм шифрування. Прикладами ключів асиметричного алгоритму шифрування є:

Відкритий ключ – ключ, що дає змогу передавати по відкритому каналу зв’язку.

Таємний ключ – ключ, що використовується при асиметричному алгоритму шифрування. Проте на відміну від відкритого ключа, він не передається по відкритому каналу.

Сеансові ключі – ключі, що виробляється між двома користувачами, у загальному випадку при захисті каналу зв'язку. Зазвичай сеансовим ключем є загальний секрет - інформація, що виробляється на основі секретного ключа однієї сторони та відкритого ключа іншої сторони [24].

Як вже було вищезазначено, то у сучасному світі безпека ПД є дуже важливою для кожної людини. Однім з засобів надання безпеки ПД є шифрування. Новітні алгоритми шифрування ставлять собі за мету надання найбільш можливої безпеки, при цьому розмір та надійність ключів, як правило, є найбільшим між різними видами шифрування алгоритмів. Можемо розглянути найпопулярніші з них.

Розширений стандарт шифрування (AES) — це алгоритм, якому довіряють уряд США та численні організації як стандарт. Незважаючи на те, що він високоефективний у 128-бітній формі, AES також використовує ключі 192 і 256 біт для важкого шифрування.

AES в основному вважається непроникним для всіх атак, за винятком грубої сили, яка намагається розшифрувати повідомлення, використовуючи всі можливі комбінації в 128, 192 або 256-бітному шифрі. Проте більшість експертів приходять до думки, що AES буде врешті визначено, як фактичний стандарт шифрування даних у приватному секторі.

Triple DES був розроблений, як покращена версія оригінального Data Encryption Standard (DES), для якого зловмисники знайшли відносно легкий спосіб для його взлому. Раніше це був рекомендованим стандартом та був найбільш широко поширений у якості надійного алгоритму симетричного шифрування.

При роботі він використовує три окремих ключі по 56 біт кожний. Загальна довжина ключа сягає 168 біт.

Алгоритм шифрування Rivers, Shamir, Adleman (далі – RSA) – асиметричний алгоритм шифрування та також є стандартом для шифрування даних, що надсилаються через мережу Інтернет. У порівнянні з вищезгаданами алгоритмами, RSA є асиметричним, так як він використовує пару ключів (відкритий та закритий). Відкритий використовується для шифрування інформації, а закритий – навпаки, для її розшифрування.

Алгоритм Blowfish – це ще один алгоритм, при створенні якого, було поставлено за мету замінити застарілий DES. Він є симетричним шифром та розділює повідомлення на 64-бітні блоки та шифрує кожен з них окремо.

Виділяється на фоні інших, тим, що вважається одним з найшвидших, а також є доволі ефективним, оскільки вважається, що він ще жодного разу не був зламаний.

З комерційної точки зору, є безкоштовним та знаходиться у відкритому доступі. Сфера використання є доволі широкою, вона покриває починаючи від платформ електронної комерції для забезпечення платежів і закінчуючи інструментами управління паролями.

Алгоритм Twofish – є наступником вищезгаданого алгоритму Blowfish. Основна його відмінність – використання ключа, що тепер досягає 256 біт. Для шифрування та розшифровування потрібен тільки один ключ, а це значить, що Blowfish – симетричний алгоритм.

Як і його попередник, є одним з найшвидших рішень та ідеально підходить для використання як в апаратному, так і в програмному середовищі. Також слід зазначити, що він також є безкоштовним та є у вільному доступі.

2.5. Сучасні реалізації хеш-функцій

Як вже було вказано вище, хеш – функція, що здійснює перетворення масиву вхідних даних довільної довжини у вихідний бітовий рядок встановленої довжини, що виконується певним алгоритмом.

Криптографічні хеші використовуються всюди, від зберігання паролів до систем перевірки файлів. Основна ідея полягає в тому, щоб використовувати детермінований алгоритм (алгоритмічний процес, який видає унікальний та зумовлений результат для завдання вхідних даних), який приймає один вхід та створює рядок фіксованої довжини щоразу. Тобто, використання одного і того ж введення завжди призводить до одного й того самого результату. Детермінізм важливий не тільки для хешів, але і для одного біта, який змінюється у вхідних даних, створюючи зовсім інший хеш. Проблема з алгоритмами хешування – неминучість колізій. Тобто той факт, що хеші є рядком фіксованої довжини, означає, що для кожного введення, яке ми можемо собі уявити, є інші можливі вхідні дані, які призведуть до того ж хешу.

Чому колізія – погано? Якщо зловмисник може створювати колізії, він може передавати шкідливі файли або дані, які мають правильний і неправильний хеш і ховатися під правильним хешом. Мета хорошої хеш-функції полягає в тому, щоб зробити надзвичайно складним для зловмисників знайти способи генерації вхідних даних, які хешуються з однаковим значенням. Обчислення хеша має бути занадто простим, оскільки це полегшує зловмисникам штучне обчислення колізій. Алгоритми хешування мають бути стійкими до «атак знаходження прообразу».Тобто, отримуючи
хеш, було б надзвичайно складно обчислити обернені детерміновані кроки, вжиті для відтворення значення, яке створило хеш (тобто знаходження прообразу).

Алгоритми хешування мають мати наступні властивості:

- зміна одного біта у вхідних даних має створити ефект зміни всього хешу;
- обчислення хешу має бути занадто простим, висока складність знаходження прообразу;
- повинен мати дуже низьку можливість колізії.

Одним із перших стандартів алгоритму хешування був MD5 hash, який широко використовувався для перевірки цілісності файлів (контрольних сум) та зберігання хешованих паролів у базах даних веб-додатків. Його функціональність досить проста, тому що вона виводить фіксований 128-бітний рядок для кожного входу і використовує тривіальні односпрямовані операції у кількох раундах для обчислення детермінованого результату. Його коротка вихідна довжина та простота операцій зробили MD5 дуже легким для злому та сприйнятливим до атаки «дня народження».

На даний момент актуальними за всіма вимогами безпеки є хеш функції сімейства SHA (Secure Hash Algorithm).

Secure Hash Algorithm 1 - алгоритм криптографічного хешування . Описано в RFC 3174 . Для вхідного повідомлення довільної довжини (максимум 2^{64} - 1біт, що дорівнює 2 екзабайт) алгоритм генерує 160-бітне (20 байт) хеш-значення, зване також дайджестом повідомлення, яке зазвичай відображається як шістнадцяткове число довжиною в 40 цифр.

Використовується у багатьох криптографічних додатках та протоколах. Також рекомендований як основний для державних установ у США. Принципи, покладені в основу SHA-1, аналогічні тим, які використовували Рональд Рівест при проектуванні MD4 [26].

Нажаль, SHA1 просто покращив MD5, збільшивши довжину виведення, кількість односпрямованих операцій та складність цих односторонніх операцій, але не дає будь-яких фундаментальних покращень проти потужніших машин, які намагаються використовувати різні атаки [27].

SHA-256 встановлює додаткові константи, що визначають поведінку алгоритму SHA-2. Однією з таких констант є розмір виведення. "256" і "512" відносяться до відповідних розмірів вихідних даних у бітах [28].

Висновки до розділу 2

Отже, у даному розділі були проаналізовані засоби зберігання ПД. Одним з засобів є менеджер паролів, що зберігає ПД, а саме - пари логін-пароль. Був проаналізований принцип його роботи та засоби забезпечення безпеки, які використовуються менеджерами паролів. До засобів забезпечення безпеки відносяться алгоритми шифрування та хеш-функції. . При аналізі роботи менеджеру паролів буди визначені недоліки. Основним з яких є неможливість вільно експортувати дані з одного менеджеру в інший за допомогою захищених каналів. Це обумовлено тим, що кожен вендор використовує свої методи захисту.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Завдання розділу 3

У попередньому розділі був проаналізований менеджер паролів, як засіб надання оптимального рівня безпеки. При аналізі були виявлені проблема безпеки, що відбувається при спробі експорту даних до програми від іншого виробника.

Отже, завданням цього розділу є створення рішення для виправлення проблеми безпеки. Та експериментальне підтвердження його роботи.

3.2 Загальний опис програми

Для вирішення проблеми безпечного експорту даних між менеджерами паролів було обрано рішення, що полягає у створенні програмного забезпечення. У якості мови програмування була обрана мова Python.

Отже, загальна проблема, як це було визначено у розділі номер 2 - неможливість безпечно експортувати дані з одного менеджеру паролів, що належить до вендора А, до іншого, який належить вендору Б. У загальному випадку, експорт відбувається у відкритому вигляді, використовуючи файл з розширенням *.CSV. Файл формату *.CSV являє собою значення, що перераховані через кому.

Файл формату *.CSV - призначений для представлення табличних даних. Рядок таблиці відповідає рядку тексту, який містить одне або кілька полів, розділених комами.

Припустимо, що в нас є два пристрої. І ми маємо безпечним шляхом надіслати файл з експортом менеджеру паролів на інший пристрій, де у наступному цей файл буде імпортований у менеджер паролів, який там використовується.

Для цього спочатку шифрується файл формату *.CSV. При розробці програмного забезпечення був обраний алгоритм шифрування AES. У якості

параметрів для шифрування були обрані:

1. Ключ довжиною 10 байт. У даному випадку при кожному запуску програми створюється випадкове значення паролю.
2. Сіль довжиною 16 байт. У даному випадку при кожному запуску програми створюється випадкове значення солі.
3. Параметр n, що є фактором вартості. Він має бути степеню двійки. У даному випадку степінь дорівнює 2^14.
4. r – розмір блоку. У даному випадку він дорівнює 8.
5. Режим роботи GCM, що надає як конфіденційність, так і аутентифікацію переданих даних (гарантуючи їхню цілісність).

Наступним кроком ми обираємо, що саме ми робимо з зашифрованим файлом. Є два варіанти розвитку подій, де перший полягає у збереженні зашифрованого файлу на у папці, де знаходиться програма. А другий полягає в передачі зашифрованого файлу до іншого пристрою, де цей файл і буде у наступному розшифрований.

Для передачі повідомлення до іншого пристрою створюється словник, який у якості значень має зашифроване повідомлення та дані, що необхідні для розшифровки файлу. Ці параметри були наведені вище.

Передача файлів відбувається за допомогою Python socket. З програми, що виступає у ролі серверу передається тип даних словник, що має зашифрований файл та параметри для його розшифровки.

Для того, щоб не було можливості зловмиснику прослухати передачу, адже зашифрований файл передається з параметрами, за допомогою яких можна розшифрувати вміст файлу. То в такому випадку було обрано використання RSA алгоритму шифрування. Це означає, що при спробі перехопити пакети, зловмисник отримає лише зашифровані дані, які майже неможливо розшифрувати та використати.

Для того, щоб клієнт точно знав якого розміру буде файл, що передає до нього сервер, використовується заголовок, який прикріплюється до кожного з

відправлених файлів. Заголовок містить змінну, що відображає довжину файлу у байтах.

Далі за допомогою циклу отримувані байти поділяються на блоки за розміром буферу, з яких надалі складається повідомлення. Розмір буферу 16 байт. При заповненні буферу на значення, якому дорівнює розмір повідомлення, повідомлення, що складається з байтів конвертується у свій початковий формат, який був перед відправкою, а саме у словник.

З цього словника прибирається пара ключ-значення паролю. Далі модифікований словник приймається функцією, яка є дзеркальною до функції, що шифрувала вміст файлу. Таким чином отримується значення, що були у файлі до шифрування.

І кінцевим результатом є запис цього значення у файл з назвою decrypted_passwords.csv. Цей файл є копією того файлу, що був на стороні програми-сервера.

3.3 Діаграма роботи програми.

На даній діаграмі (див. рисунок 3.1) відображена загальна робота програмного забезпечення, що служить для безпечного експорту даних між менеджерами паролів. В нас є дві програми, які контактують між собою. Перша з них виконує шифрування над файлом, що обирається.

У даному випадку у якості такого файлу був обраний файл експорту паролів PASSWORDS.CSV. Програма, що названа RSA_Server.py виконує шифрування даного файлу. Далі користувачем обирається подальший шлях зашифрованого файлу.

Першим варіантом є збереження його у шифрованому вигляді у папці з програмою RSA_Server.py. Іншим варіантом є надсилання зашифрованого файлу до іншої програми, яка називається RSA_client.py. У такому випадку програма, що приймає дані, розшифровує дані за допомогою ключа, який передається одразу з зашифрованим файлом.

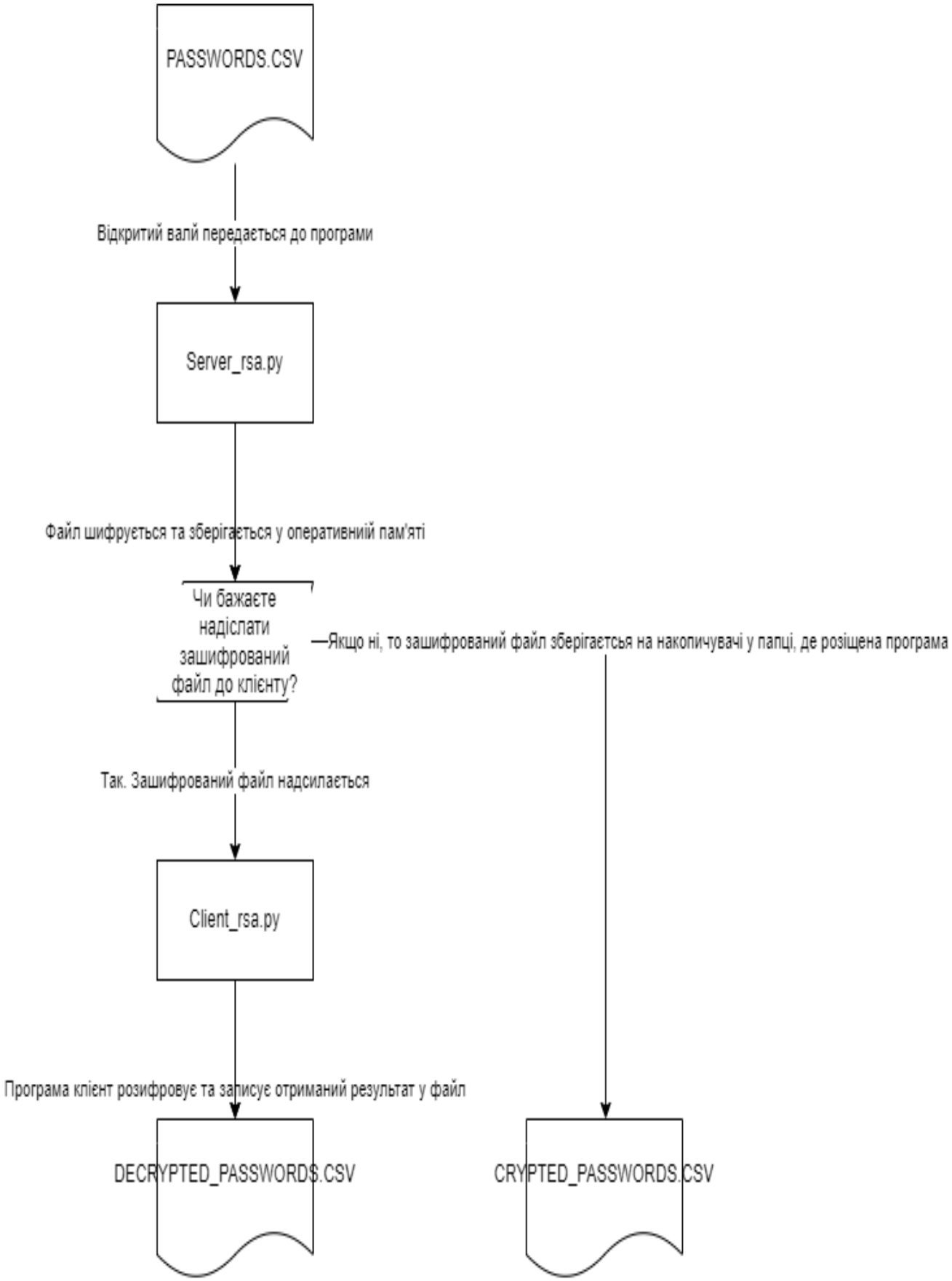

Рисунок 3.1 - Діаграма загальної роботи програми

На рисунку 3.1 зображено можливі результати роботи в залежності від обраних користувачем налаштувань. Де, першим варіантом є штфрування та збереження на носії, де знаходиться сама програма. А другим – надсилання зашифрованого файлу до іншого пристрою за допомогою захищеного каналу.

3.4 Опис методів та бібліотек, які були використані для написання програмного забезпечення.

На рисунку 3.2 наведено бібліотеки, які були підключені та використані для роботи програми.

```
import socket
import pickle
import ssl
from base64 import b64encode, b64decode
import hashlib
from Cryptodome.Cipher import AES
import os
from Cryptodome.Random import get_random_bytes
import random
import string
```

Рисунок 3.2 - Використані бібліотеки при розробці

Бібліотека socket використовується для встановлення контакту програми, що відіграє роль серверу та клієнту.

Бібліотека Secure Socket Layer (далі – ssl) використовується для «обернення» сокетів, що тим самим діє змогу використовувати безпечне ssl підключення, це гарантує, що пакети, котрі передаються не будуть підслухані.

Hashlib використовується для створення приватного ключа, який буде використаний далі для шифрування файлу. Бібліотека Cryptodome надає змогу отримати доступ до модулю AES. Надалі він буде використаний, як алгоритм для шифрування файлу.

На рисунку 3.3 зображено метод generate_random_string(), який приймає будь-яке значення числового типу та повертає довільну строку довжини, що була вказана при виклику цієї функції.

На рисунку 3.4 зображена функція encrypt(). Вона служить для шифрування даних. У якості вхідних даних вона приймає текст, що має бути зашифрованим, та пароль строкового типу. Вона повертає словник, що містить: зашифрований текст, сіль, випадковий одноразовий номер та тег, який потрібний для аутентифікації даних при використані AES у режимі GCM. Це гарантує, щ ніхто не зможе змінити дані, якщо ми не взнаємо про це під час розшифрування.

```
def generate_random_string(length):
    letters = string.ascii_lowercase
    rand_string = ''.join(random.choice(letters) for i in range(length))
    print("Random string of length", length, "is:", rand_string)
    return rand_string
```

Рисунок 3.3 - Метод generate_random_string

```
def encrypt(plain_text, password):
    # generate a random salt
    salt = get_random_bytes(AES.block_size)

    # use the Scrypt KDF to get a private key from the password
    private_key = hashlib.scrypt(
        password.encode(), salt=salt, n=2 ** 14, r=8, p=1, dklen=32)

    # create cipher config
    cipher_config = AES.new(private_key, AES.MODE_GCM)

    # return a dictionary with the encrypted text
    cipher_text, tag = cipher_config.encrypt_and_digest((plain_text))
    return {
        'cipher_text': b64encode(cipher_text).decode('utf-8'),
        'salt': b64encode(salt).decode('utf-8'),
        'nonce': b64encode(cipher_config.nonce).decode('utf-8'),
        'tag': b64encode(tag).decode('utf-8')
    }
```

Рисунок 3.4 - Метод encrypt()

На рисунку номер 3.5 зображена функція decrypt(). У загальному розумінні вона є протилежністю методу encrypt().

Так як ми маємо справу з симетричним алгоритмом шифрування, то для розшифрування потрібен той самий ключ та параметри, які були отримані при шифруванні.

```
def decrypt(enc_dict, password):
    # decode the dictionary entries from base64
    salt = b64decode(enc_dict['salt'])
    cipher_text = b64decode(enc_dict['cipher_text'])
    nonce = b64decode(enc_dict['nonce'])
    tag = b64decode(enc_dict['tag'])

    # generate the private key from the password and salt
    private_key = hashlib.scrypt(
        password.encode(), salt=salt, n=2 ** 14, r=8, p=1, dklen=32)

    # create the cipher config
    cipher = AES.new(private_key, AES.MODE_GCM, nonce=nonce)

    # decrypt the cipher text
    decrypted = cipher.decrypt_and_verify(cipher_text, tag)

    return decrypted
```

Рисунок 3.5 - Метод decrypt()

На рисунку номер 3.6 зображено методи, які використовуються у даній роботі для роботи з файлами. Перша з них читає вміст файлу, записує їх у змінну та повертає її. Друга використовується для запису байтів до файлу.

```python
def read_the_files_bytes(file_name):

    with open(file_name, 'rb') as fp:
        text_from_file = (fp.read())
    #print(text_from_file)
    return text_from_file

def write_bytes_into_file(file_name, bytes_for_file):
    with open(file_name, 'wb') as wp:
        wp.write(bytes_for_file)
```

Рисунок 3.6 - Методи для роботи з файлами

На рисунку 3.7 наведено функцію get_server_contion(). За допомогою її ми створюємо сокет та тим самим налаштовуємо підключення. Для зв'язку використовуємо ssl протокол, тому у файлі самої програми додаєм ssl сертифікат. Цей протокол дає змогу виконувати обмін пакетами по захищеному каналу зв'язку. Також слід зазначити, що до повідомлення, яке відправляється у початок додається заголовок, що відображає загальну довжину у байтах, що має бути отримана програмою отримувачем.

```python
def get_server_contion(file_data):
    HEADERSIZE = 10
    context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
    context.load_cert_chain('new.pem', 'private.key')

    soc = socket.socket()
    soc.bind((HOST, PORT))
    print(f"Server is running at {HOST}' with '{PORT}'.")
    print("Ready for sending the information")
    soc.listen(3)
    s_soc = context.wrap_socket(soc, server_side=True)
    while True:
        conn, add = s_soc.accept()
        msg = file_data
        msg = bytes(f"{len(msg):<{HEADERSIZE}}", 'utf-8') + msg
        print(msg)
        conn.send(msg)
```

Рисунок 3.7 - get_server_contion()

На рисунку 3.8 наведено метод main(). Цей метод є головною функцією, з нього починається робота усієї програми.

```
def main():

    filename = 'passwords.csv'
    file = open(filename, "rb")  # Теперь мы читаем этот файл
    password = generate_random_string(10)  # создаем пароль для шифрования
    plain_text = read_the_files_bytes(filename)
    encrypted = encrypt(plain_text, password)
    encrypted_plus_password = {}
    encrypted_plus_password.update(encrypted)
    encrypted_plus_password["password"] = password

    file_data = pickle.dumps(encrypted_plus_password)


    option = input("The file was crypted. To send it to another device enter 1. If you want to just save and on this device enter 2\n")
    if option == "1":
        get_server_contion(file_data)
    if option == "2":
        write_bytes_into_file('crypted.txt', encrypted_plus_password['cipher_text'].encode())

if __name__ == "__main__":
    main()
```

Рисунок 3.8 - Метод main()

У ньому відбувається відкриття файлу, який у наступному буде зашифровано, далі формується ключ для шифрування цього файлу та відбувається саме шифрування. Після вже зашифрований файл може бути або збережений у папці з програмою, або надісланий до програми-клієнта, яка зможе обробити його.

3.5 Результат розробки

Для демонстрації роботи програми будемо шифрувати файл passwords.csv. Припустимо, що він був отриманий у якості експорту даних з менеджеру паролів. Його вміст можемо бачити на рисунку 3.9.

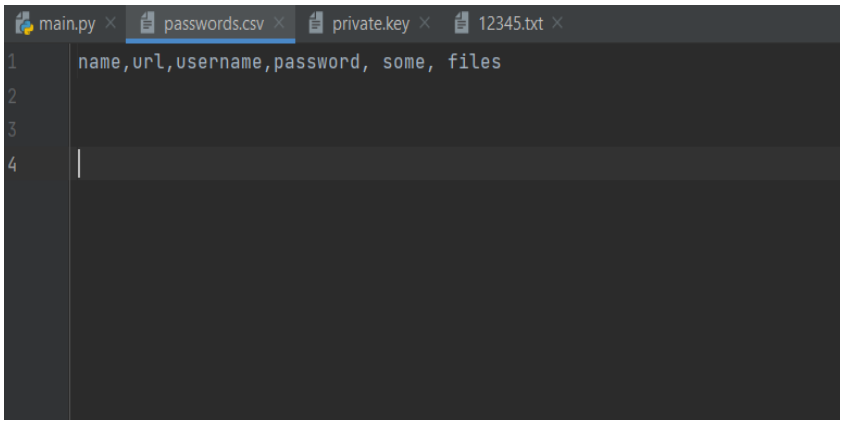

Рисунок 3.9 - Вміст файлу passwords.csv

Далі для презентації роботи програмного забезпечення запустимо програму RSA_server та отримаємо наступне у командній строчці. Результат запуску програми можемо бачити на рисунку 3.10.

Рисунок 3.10 - Результат запуску програми RSA_server

Можемо бачити, що при запуску програми сформувалася випадкова строка довжиною 10. Посилаючись на пункт вище, нам відомо, що ця строка випадкових символів служить ключем для шифрування алгоритмом шифрування AES.

Також можемо бачити, що файл вже зашифрований, і в нас є два варіанти, що ми можемо з ним зробити. Де перший з них – надсилання файлу до програми клієнта. А другий – збереження зашифрованого файлу на комп’ютері, на якому була запущена програма. Спочатку почнемо з збереження файлу на пристрої. Для цього користувач потрібен ввести «2» до командної строки. Шифротекст записується у файл crypted.txt. Результат наведено на рисунку 3.11.

Рисунок 3.11 - Результат шифрування файлу

Наступним кроком є перевірка працездатності першого режиму. Для його запуску потрібно ввести до командної строки цифру 1. Тепер для того, щоб програма одержувач отримала повідомлення, потрібно її запустити. У нашому випадку вона називається RSA_client. Результат запуску програми одержувача наведено на рисунку 3.12.

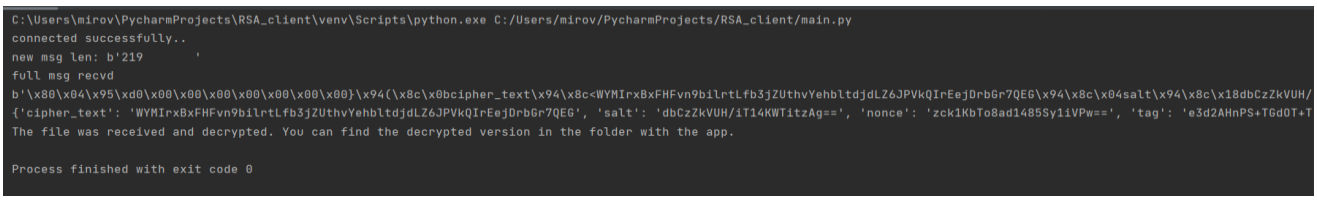

Рисунок 3.12 - Результат запуску програми-одержувача

На рисунку 3.12 можемо бачити, що запуск програми пройшов нормально. Виходячи з отриманих повідомлень у командній строчці від програми, повідомлення було отримано.

Також можемо бачити, що довжину повідомлення, саме повідомлення у вигляді байтів та повідомлення, яке вже оброблене та переформатовано у тип даних словник. Цей словник має сам шифр-текст, сіль, параметр nonce та тег. Усі ці параметри потрібні для розшифрування шифр-тексту.

Для підтвердження, що було отримано та надалі правильно розшифровано повідомлення, потрібно перевірити вміст файлу decrypted_passwords.csv. Після розшифрування у цей файл записується розшифроване повідомлення. На рисунку 3.13 наведено вміст файлу decrypted_passwords.csv.

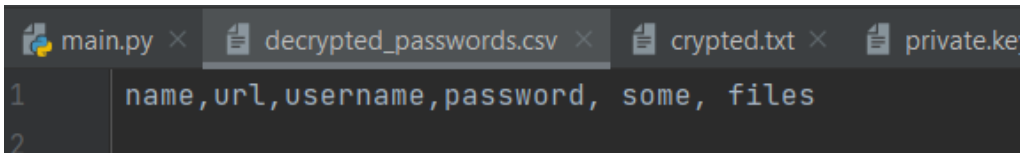

Рисунок 3.13 - вміст файлу

Для підтвердження справної роботи ssl протоколу використано програму для перехвату пакетів Wireshark. Для цього потрібно спочатку запустити програму для перехоплення пакетів, потім зробити підключення між сервером та клієнтом та передати файл.

На рисунку 3.14 зображено інтерфейс програми Wireshark. У даний час вона налаштована на перехоплення усіх можливих пакетів.

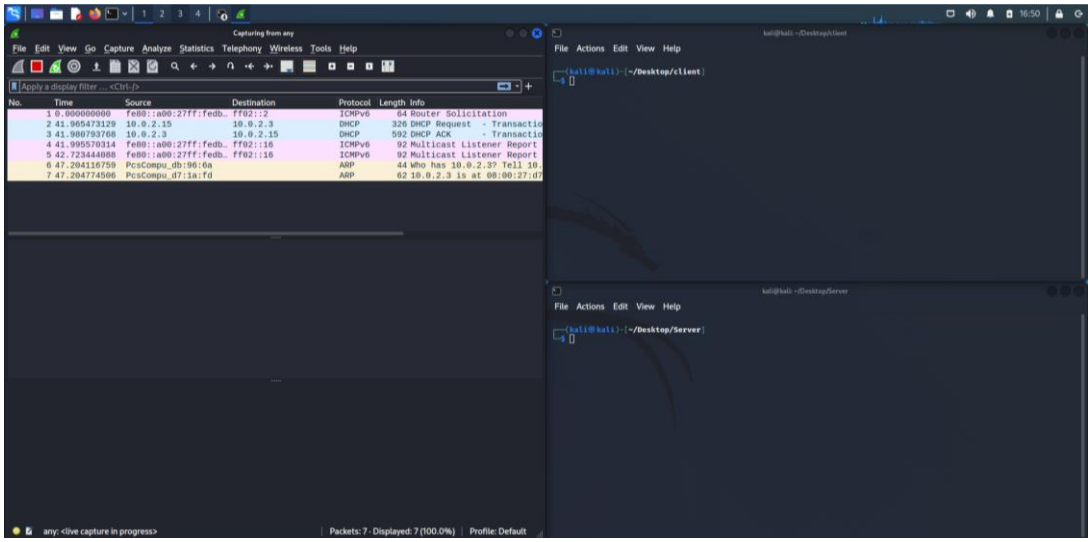

Рисунок 3.14 – Wireshark

На рисунку наведено 3.15 зображені перехопленні пакети, аналізуючи їх вміст, можна зробити висновок, що вони усі зашифровані та не піддаються читанню.

Як видно з рисунку 3.15, то для створення підключення використовується протокол TLSv1.3, який гарантує захищеність передачі даних.

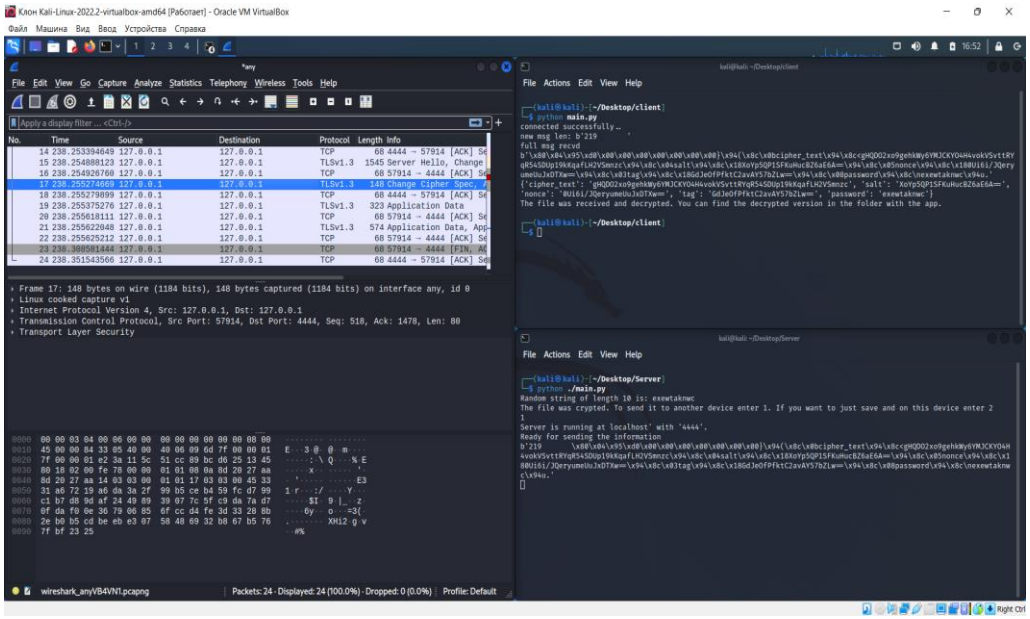

Рисунок 3.15 - Перехоплені пакети

Висновки до розділу 3

Отже, у даному розділі було виконано створення програмного застосунку, який вирішує проблему експорту паролів між менеджерами паролів. Його робота полягає у шифруванні файлу з паролями та подальшою його передачею до іншого пристрою. На стороні іншого пристрою має бути програма-клієнт, яка отримує файл, розшифровує його та записує у файл, який зберігається на пристрої.

Також слід зазначити, що експериментальним шляхом було доведено, що пакети, що передаються не можуть бути розшифровані, так як для передачі використовується протокол SSL.

Можливим наступним покращенням даного застосунку можна назвати відкриття CSV файлів з оперативної пам’яті, а накопичувача.

ВИСНОВКИ

Отже, при виконанні даної роботи була проаналізована проблема безпеки ПД.

Були визначені оптимальні засоби аутентифікації, що надають достатній рівень захищеності ПД для звичайного користувача.

Були проаналізовані скомпрометовані логіни та паролі. На базі цих знань були визначені загальні рекомендації для створення паролю, який надає достатній рівень захисту з огляду на сучасні обчислювальні можливості.

Було визначено, що з огляду на попередні результати аналізу, менеджер паролів є оптимальним інструментом для захисту ПД.

При аналізі менеджерів паролей, було визначено недолік, який був вирішений за рахунок створення програмного забезпечення. Працездатність програмного забезпечення було підтверджене експериментальним шляхом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Поняття про ідентифікацію [Електронний ресурс]. - Режим доступу: https://sites.google.com/site/identifikaciataautentifikacia/ponatta-pro-identifikaciu
2. Автентифікація (інформаційні технології) [Електронний ресурс]. - Режим доступу: https://vue.gov.ua
3. Що таке авторизація? [Електронний ресурс]. - Режим доступу: https://uk.theastrologypage.com/authorization
4. Методи аутентифікації [Електронний ресурс]. - Режим доступу: https://sites.google.com/site/identifikaciataautentifikacia/ponatta-pro-autentifikaciu/metodi-autentifikaciie
5. Ідентифікація і аутентифікація користувачів [Електронний ресурс]. - Режим доступу: https://e-tk.lntu.edu.ua/mod/resource/view.php?id=3787
6. The Current State Of Authentication: We Have A Password Problem [Електронний ресурс]. - Режим доступу: https://www.smashingmagazine.com/2016/06/the-current-state-of-authentication-we-have-a-password-problem/
7. Static Password [Електронний ресурс]. - Режим доступу: https://www.sciencedirect.com/topics/computer-science/static-password
8. Про захист персональних даних [Електронний ресурс]: закон України від 01.06.2010 № 2297-VI. - Режим доступу: https://zakon.rada.gov.ua/laws/show/2297-17#Text
9. Data Protection Act [Електронний ресурс]. - Режим доступу: https://www.bbc.co.uk/bitesize/guides/z6kj6sg/revision/4
10. Survey Says: People Have Way Too Many Passwords To Remember [Електронний ресурс]. - Режим доступу: https://www.buzzfeednews.com/article/josephbernstein/survey-says-people-have-way-too-many-passwords-to-remember#.mb8zOBb52
11. Как вскрыть до 70% паролей за несколько минутВсе гуд але заголовок [Електронний ресурс]. - Режим доступу: https://blog.elcomsoft.ru/2017/02/kak-vskryit-do-70-paroley-za-neskolko-minut/
12. Top 10 passwords [Електронний ресурс]. - Режим доступу: https://xa.to/top10k
13. Поняття про паролі: їх злом та методи захисту. [Електронний ресурс]. - Режим доступу: https://infopedia.su/30x2365.html
14. Анализ учетных записей одного (не)надежного email-сервиса [Електронний ресурс]. - Режим доступу: https://habr.com/ru/post/257881/
15. Are Your Passwords in the Green? [Електронний ресурс]. - Режим доступу: https://www.hivesystems.io/blog/are-your-passwords-in-the-green

16. Top Ways to Store Your Digital Files By Carlos Rodriguez [Електронний ресурс]. - Режим доступу: https://www.myamberlife.com/learn/top-ways-to-store-your-digital-files/

17. 5 Common Encryption Algorithms and the Unbreakables of the Future [Електронний ресурс]. - Режим доступу: https://www.arcserve.com/blog/5-common-encryption-algorithms-and-unbreakables-future

18. Хмарне сховище чи локальні сервери: 9 критеріїв, які слід врахувати під час вибору [Електронний ресурс]. - Режим доступу: https://www.microsoft.com/uk-ua/microsoft-365/business-insights-ideas/resources/cloud-storage-vs-on-premises-servers

19. Менеджеры паролей [Електронний ресурс]. - Режим доступу: https://hi-tech.ua/article/menedzheryi-paroley/

20. Менеджеры паролей для мобильников [Електронний ресурс]. - Режим доступу: https://web.archive.org/web/20090621042237/http://www.computerra.ru/reviews/419570/

21. Генератор надежных паролей 1Password [Електронний ресурс]. - Режим доступу: https://1password.com/ru/password-generator/

22. https://www.kaspersky.ru/resource-center/definitions/encryption?referer2=tcid_admitad_e779e05a169320fc9c047c29a6913059_1010045_x4&tagtag_uid=e779e05a169320fc9c047c29a6913059 [Електронний ресурс]. - Режим доступу: https://www.kaspersky.ru/resource-center/definitions/encryption?referer2=tcid_admitad_e779e05a169320fc9c047c29a6913059_1010045_x4&tagtag_uid=e779e05a169320fc9c047c29a6913059

23. Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. Handbook of applied cryptography. — CRC-Press, 1996. — С. 32. — ISBN 0849385237.

24. Сеансовый ключ [Електронний ресурс]. - Режим доступу: https://dic.academic.ru/dic.nsf/ruwiki/1863687

25. 5 Common Encryption Algorithms and the Unbreakables of the Future [Електронний ресурс]. - Режим доступу: https://blog.storagecraft.com/5-common-encryption-algorithms/

26. SHA-1 [Електронний ресурс]. - Режим доступу: https://www.webcitation.org/61A39qlpS?url=http://www.infosec.sdu.edu.cn/uploadfile/papers/Finding%20Collisions%20in%20the%20Full%20SHA-1.pdf

27. Алгоритмы хеширования - простое объяснение сложного [Електронний ресурс]. - Режим доступу: https://vc.ru/crypto/47132-algoritmy-heshirovaniya-prostoe-obyasnenie-slozhnogo

28. Пошагово объясняем, как работает алгоритм хеширования SHA-2 (SHA-256) [Електронний ресурс]. - Режим доступу: https://tproger.ru/translations/sha-2-step-by-step/