МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«____»_____ 2023 р.

На правах рукопису
УДК 004.056.5:510.22(043.3)

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: Криптографічний модуль гешування паролів в реальному часі

Виконавець: Ярослав МАРЧЕНКО

Науковий керівник: к.т.н., доц. Сергій ІЛЬЄНКО

Консультант розділу «Охорона навколишнього середовища»: к.т.н., доцент Тетяна ДМИТРУХА

Нормоконтролер: к.т.н., доц. Сергій ІЛЬЄНКО

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Магістр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«__» _____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

здобувача вищої освіти Марченка Ярослава Володимировича

1. Тема: Криптографічний модуль гешування паролів в реальному часі затверджена наказом ректора від «15» вересня 2023 р. № 1814/ст.
2. Термін виконання: з 16.10.2023 р. по 31.12.2023 р.
3. Вихідні дані: проаналізувати існуючі системи та методики аналізу і оцінки модулів гешування паролів з використаннням кодів автентифікації повідомлення; на основі аналізу виділити вхідні і вихідні параметри, сильні та слабкі сторони , завдяки ним провести порівняння існуючих систем; дослідити коди автентифікації повідомлень та їх особливості, розробити методику, алгоритм та на їх основі програмний модуль гешування паролів в реальному часі; тестування функціональних можливостей розробленого застосунку.
4. Зміст пояснювальної записки: аналіз існуючих систем та методик забезпечення безпеки паролів користувачів і оцінки ризиків інформаційної

безпеки; розробка методики та криптографічного модулю гешування паролів в реальному часі; створення програмного забезпечення запропонованої системи, верифікація отриманих результатів та порівняння з подібними існуючими системами.

5. КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

№ з/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	16.10.2023	*Виконано*
2.	Аналіз літературних джерел	19.10.2023	*Виконано*
3.	Обґрунтування вибору рішення	22.10.2023	*Виконано*
4.	Збір інформації	28.10.2023	*Виконано*
5.	Дослідження сучасних систем і методик аналізу та оцінки менеджерів паролів	09.11.2023	*Виконано*
6.	Дослідження кодів автентифікації повідомлень, та їх особливості	13.11.2023	*Виконано*
7.	Розробка методики та алгоритму реалізації криптографічного модуля гешування паролів в реальному часі	16.11.2023	*Виконано*
8.	Створення програмного забезпечення	17.11.2023	*Виконано*
9.	Оформлення і друк пояснювальної записки	14.12.2023	*Виконано*
10.	Оформлення презентації	18.12.2023	*Виконано*
11.	Отримання рецензій від рецензента	22.12.2023	*Виконано*

6. Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Хімічне забруднення довкілля	Дмитруха Т.І.		

7. Дата видачі завдання: «16» жовтня 2023 р.

Здобувач вищої освіти Ярослав МАРЧЕНКО
(підпис, дата)

Керівник кваліфікаційної роботи Сергій ІЛЬЄНКО
(підпис, дата)

РЕФЕРАТ

Кваліфікаційна робота студента складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел, додаток А, додаток Б загальним обсяг роботи складає 97 сторінок, має 75 рисунків, 8 таблиць. Список використаних джерел містить 25 найменувань і займає 3 сторінки.

Метою кваліфікаційної роботи є розробка криптографічного модуля гешування паролів в реальному часі

В кваліфікаційній роботі розглянуті питання дослідження сучасних методів хешування паролів, процедура хешування паролів в реальному часі на базі кодів автентифікації

Проведені дослідження базуються на сучасних методах побудови захищених інформаційних мереж.

Запропоновано авторський криптографічний модуль гешування паролів в реальному часі, за рахунок реалізації HMAC в поєднанні з динамічною сіллю, що дозволяє застосувати його в системах автентифікації суб'єкта інформаційної діяльності на основі застосування додатково згенереваного унікального ключа для кожного користувача.

Ключові слова: функції хешування, надійність паролю, коди автентифікації повідомлення, статична сіль, динамічна сіль, автентифікація.

ЗМІСТ

ВСТУП

Актуальність. У наш час проблема створення ненадійних паролю, з кожним роком стає все більшою**.** Кожен з нас проходить автентифікацію кілька разів на день, сам того не помічаючи. Для доступу до інформації, облікового запису від користувача очікується введення логіну та паролю. Тобто тільки ця інформація забезпечує безпеку конфіденційних даних. В залежності від довжини, та використання додаткових символів, користувач самостійно може збільшити час, потрібний для атакою грубою силою. Компанії в свою чергу мають вимоги при створенні паролю. До цих вимог частіше за все входить: фіксована кількість символів, використання малих та великих літер, цифр, спеціальних символів. Це все робиться, для уникнення використання ненадійних паролів, та популярних паролів.

Після створення, та проходження всіх вимог, відповідальність за безпеку вашої інформації несе компанія, в продукті, якої ви зареєструвалися.
Інформація зберігається у вигляді хешу. Кожен раз при проходженні автентифікації, буде звірятися надана інформація з тою, яка міститься в базі даних. І тільки при відповідності даних, надається доступ до облікового запису. До хешу, зазвичай додається сіль. Вона допомагає збільшити стійкість до атак.

Додатковим забезпеченням безпеки при автентифікації є використання Двофакторної автентифікації. Зазвичай при успішному введені паролю та логіну, користувач отримує додатковий код з смс повідомлення.

Метою дипломної роботи є розробка криптографічного модуля гешування паролів в реальному часі.

Виходячи з поставленої мети завданнями кваліфікаційної роботи є:

1) дослідження сучасних методів хешування паролів;

2) розробити процедуру хешування паролів в реальному часі на базі HMAC в поєднанні з динамічною сіллю;

3) провести оцінку доцільності впровадження програмної реалізації криптографічного модуля захищеної гешування паролів в реальному часі.

Галузь застосування. Розроблений мною, модуль гешування паролів, розрахований для застосування в системі автентифікації

Об'єкт дослідження –являється процедура формування криптографічних хеш-функцій.

Предмет дослідження. методи підвищення ефективності функціонування захищених систем та мереж на основі використання методів формування криптографічних хеш-функцій за умови забезпечення проведення автентифікації, динамічна сіль.

Методи дослідження. Проведені дослідження базуються на сучасних методах побудови захищених інформаційних мереж.

Практична цінність. полягає в тому, що розроблено програмний модуль реалізації модуля гешування паролів в реальному часі мовою програмування Python, що дає можливість його інтегрувати в модуль двофакторної автентифікації за допомогою електронної пошти.

Наукова новизна. Обумовлена рішенням задач щодо підвищення захисту інформаційних систем, зокрема інформації, що зберігається, і оброблюється в сучасних системах на основі забезпечення цілісності та автентичності.

Запропоновано авторський криптографічний модуль гешування паролів в реальному часі, за рахунок реалізації HMAC в поєднанні з динамічною сіллю, що дозволяє застосувати його в системах автентифікації субєкта інформаційної діяльності на основі застосування додатково згенереваного унікального ключа для кожного користувача.

Апробація. Ільєнко С., Марченко Я. В., Кравчук І. А. Сучасне забезпечення безпеки паролів // Modern problems of science, education and society: VIII Міжнародна науково-практична конференція, 9-11 жовтня 2023 р.: тези доповіді. – К., 2023. – С.313-316.

РОЗДІЛ 1.

ЗАГАЛЬНІ ПОНЯТТЯ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ

В цьому розділі, я розкрию основні питання, щодо забезпечення безпеки паролів, розгляну сучасні підходи захисту паролів від атак, та їх вразливі місця. Використання алгоритмів хешування у комбінації з паролями, їх недоліки та переваги.

1.1 Сучасний стан забезпечення безпеки та проблеми безпеки паролів.

Для початку, щоб зрозуміти всю актуальність проблеми створення не надійного паролю, розглянемо дослідження, яке провела компанія Hive Systems[1]. Атакувавши секретні ключі різної довжини, компанія надала інформацію, як швидко зловмисник може дізнатися ключ користувача, в залежності від його кількості символів, використовуючи атаку брутфорс, ці дані наведені в таблиці 1.1.

Таблиця 1.1.

Стійкість паролів до брутфорс атак

Кількість символів	8	12	16
Секретний ключ створений з цифр	Моментальне розсекречення	Моментальне розсекречення	Для отримання секретного ключа необхідно 15 секунд
Секретний ключ створений з маленьких літер	Моментальне розсекречення	Для отримання секретного ключа необхідно 2 хвилини	Для отримання секретного ключа необхідно 2 роки

Продовження таблиці 1.1.

Кількість символів	8	12	16
Секретний ключ створений з маленьких та великих літер	Моментальне розсекречення	Для отримання секретного ключа необхідно 7 днів	Для отримання секретного ключа необхідно 140 тисяч років
Секретний ключ створений з маленьких та великих літер, цифри	Моментальне розсекречення	Для отримання секретного ключа необхідно 2 місяці	Для отримання секретного ключа необхідно 2 мільйони років
Секретний ключ створений з маленьких та великих літер, цифри, спеціальні символи	Для отримання секретного ключа необхідна 1 секунда	Для отримання секретного ключа необхідно 8 місяців	Для отримання секретного ключа необхідно 16 мільйонів років

З отриманої інформації можна сформувати критерії для створення надійного паролю для забезпечення безпеки конфіденційних даних[2]:

- Рекомендована довжина секретного слова становить 8 і більше символів;
- Використовуйте різні унікальні слова та словосполучення;
- Використовуйте різні символи та їх комбінації;
- Систематично змінюйте паролі, хоча б раз в півроку;

Що не рекомендується використовувати при створенні паролю:

- Не використовуйте тільки один вид символів;
- Уникайте особисту інформації, яка може перебувати в публічному доступі;

До цього пункту можна віднести таку інформацію: місце роботи та навчання, ім’я, прізвище, дата народження, дата весілля, ім’я домашньої тварини тощо.

- Не використовуйте однакові паролі, для різних електронних ресурсів;
- Забудьте про часто вживані паролі;

Приклад часто вживаних паролів[3]: 12345, qwerty, password. Якщо витратити хвилину свого життя, кожен користувач може знайти, ці паролі у вигляді таблиць, списків тощо, і навіть отримати статистику, про кількість створень таких паролів. Тобто такі паролі гарантовано будуть в словниках, які зловмисники можуть використовувати, для того, щоб отримати інформацію користувачів.

Виконавши ці не складні вимоги, ви зможете забезпечити захист конфіденційної інформації, та паролю від брутфорс атак. Про цей вид атак поговоримо більш детально в наступному пункті дипломної роботи.

В наш прогресивний час, враховуючи те, що у кожного користувача повинен бути унікальний пароль, для кожної соціальної мережі або програмного забезпечення. Рано чи пізно постає питання, де зберігати таку велику кількість паролів, та у першу чергу забезпечити їх безпеку.

Методи зберігання паролів:

- Зберігати інформацію в голові;

Такий метод є досить надійним підходом, бо виток інформації може статися тільки за участі самого користувача. Але користувач у свою чергу просто може забути свій пароль, в деяких випадках, це може гарантувати втрату доступу до конфіденційних даних.

- Записувати секретні ключі в блокнот;

На мою думку, цей метод гарантує вам досить високий рівень безпеки, бо для отримання інформації зловмисник повинен викрасти блокнот, або мати безпосередній доступ до нього. Головним нюансом цього методу, є легкість отримання потрібної хакеру інформації. Це може трапитися при ненадійному

зберіганні самого блокноту. Користувач може залишити блокнот на столі. Без надійного спостереження, інформація може потрапити до сторонніх осіб. Вирішити цю проблему можна, додатково додаючи комбінації символів, до паролів, які вже написані в блокноті. Ці комбінації повинен знати, і пам’ятпти лише власник паролю.

- Зберігати дані в текстовому блокноті;

Цей метод є досить популярним, але він не має нічого спільного з безпекою даних. Текстові файли потрібно використовувати у поєднанні з продуктами для захисту інформації.

- Використовувати WinRAR;

Архіватор файлів WinRAR, має вбудовану можливість створити архів з паролем[4]. Користувач при створенні архіву повинен обрати опцію встановити пароль, на виході ми отримуємо архів захищений паролем. Якщо переглянути створений архів, то ми можемо помітити, що користувач без знання паролю може ознайомитися з його вмістом, але не може модифікувати чи переглядати файли.

На рисунку 1.1 показано створений архів з паролем

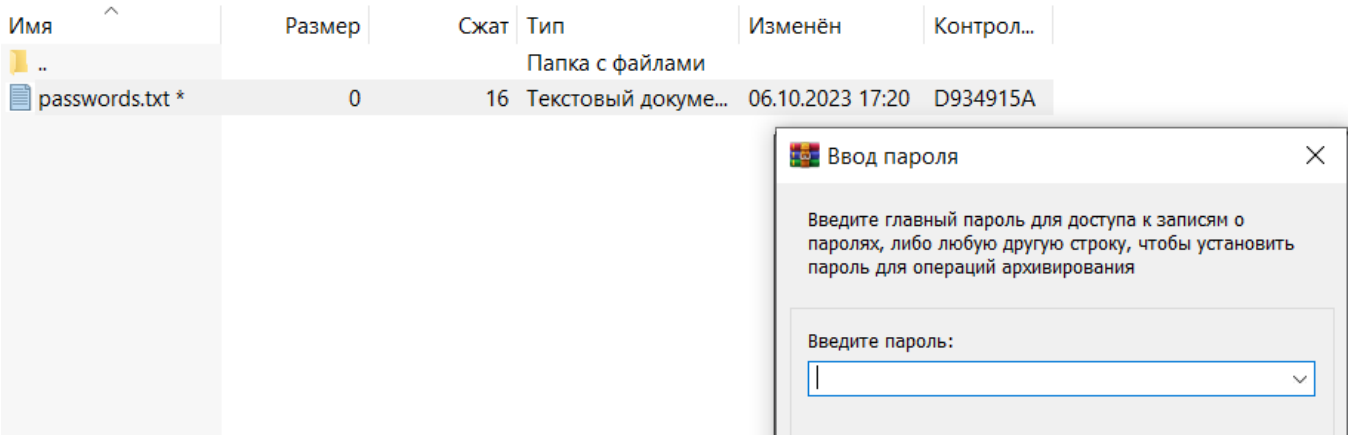


Рис.1.1 Створений архів з паролем

Уникнути це можна, додатково зашифрувавши назви файлів та папок, за допомогою можливостей WinRAR. Таким чином без знання паролю, користувач не може отримати інформацію про архів, та доступ до нього.

На рисунку 1.2 показано, створений архів з зашифрованими назвами файлів та папок

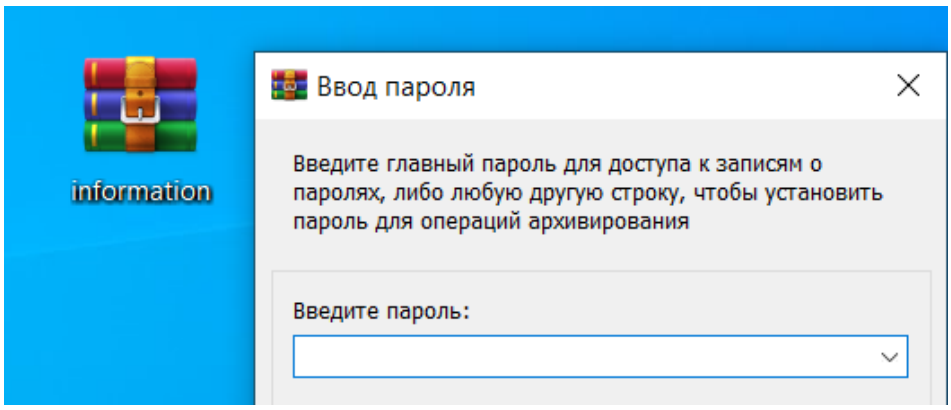


Рис.1.2. Архів з зашифрованими назвами файлів та папок

До створення паролів, які захищають архіви, дешифрують жорсткий диск, або надають доступ до менеджера паролів, ставтеся відповідально. В такому випадку надавайте перевагу секретним ключам, довжиною 15 - 20 символів. Такі паролі потрібно тримати в голові.

- Використовувати утиліти Microsoft Office;

Продукти word та excel, теж мають змогу надати безпеку даним користувача[5]. Це можливо, за допомогою вбудованої функції шифрування документу з використанням паролю.

- Використовувати програмне забезпечення, що реалізує функцію шифрування;

Представником такого програмного продукту є VeraCrypt[6].

Функціонал програмного забезпечення:

- Шифрування операційної системи;
- Шифрування дисків;
- Створення криптоконтейнеру;
- Створення криптоконтейнеру з подвійним дном;
- Можливість шифрування периферійних пристроїв;
- Забезпечує захист від перебору паролів;

До переваг можна віднести:

- Відкритий вихідний код;
- Ліцензія Apache License 2.0;
- VeraCrypt підтримує різні операційні системи;

- Наявне автоматичне вимкнення томів при неактивності;
- Використання параметру PIM у комбінації з паролем;

PIM - використовується для збільшення числа ітерацій, тобто повторень алгоритмів хешування, які використовуються безпосередньо для розшифровки даних.

- Використання периферійних пристроїв;

Цей метод є хорошим варіантом зберігання вашої інформації, але від користувача потребується виконувати одну не складну дію, перевіряти стан безпеки персонального комп'ютера, електронного носія. Бо віруси можуть спотворити інформацію, без можливості її відновлення.

Цей метод краще використовувати у комбінації з програмним забезпеченням, що реалізує функцію шифрування, або купувати носії з вбудованою функцією шифрування.

- Використовувати менеджер паролів;

Пропоную розглянути кілька менеджерів паролів, навести їх переваги та недоліки[7],[8].

Dashlane

Функціонал менеджеру Dashlane:

- Об'єм сховища 1 – 5 гб
- Автентифікація методами 2FA, Touch ID, Face ID, та для платної версії U2FA.
- Має розширення для Safari, Chrome, Edge, Firefox
- Наявний вбудований VPN

Переваги:

- Доступна безкоштовна версія
- Простий у використанні та встановлені
- Хороша робота автозаповнення
- За досить не великому вартість, додатково має VPN

Недоліки:

- Не повні можливості в безкоштовній версії

- Не має можливість імпортувати паролі з телефону

NordPass

Функціонал менеджеру NordPass:

- Об’єм сховища 3 гб
- Двофакторна автентифікація, та можливість використання Touch ID, Face ID
- Має розширення для Safari, Chrome, Edge, Firefox, Opera, Brave, Vivaldi
- Шифрування XChaCha20

Переваги:

- Доступна безкоштовна версія
- Нове покоління алгоритму шифрування
- Продукт йде в комплекті з NordVPN та NordLocker

Недоліки:

- Використання додаткового програмного забезпечення для двофакторної автентифікації

1Password

Функціонал менеджеру 1Password:

- Об'єм сховища 1– 5 гб
- Автентифікація методами 2FA, Touch ID, Face ID
- Підтримка Chrome OS
- Має розширення для Chrome, Edge, Firefox, Brave.
- Може працювати без підключення до інтернету

Переваги

- Підтримка Chrome OS
- Легкість у використані
- Можливість роботи без інтернету

Недоліки

- Не має безкоштовної версії

Keeper

Функціонал менеджеру Keeper:

- Об'єм сховища 5 гб
- Автентифікація методами SMS, TOTP, RSA SecurID, DUO Security, U2F, KeeperDNA, Touch ID, Face ID
- Має розширення для Chrome, Edge, Firefox, Safari, Opera
- Наявний застосунок для Windows, macOS, Linux, Android, IOS

Переваги

- Використання алгоритму AES
- Додаткова функція зберігання файлів та фотографій
- Велика кількість варіантів автентифікації

Недоліки

- Не має безкоштовної версії, для ознайомлення доступна версія trial

В таблиці 1.2 будуть вказані головні критерії, які повинен знати користувач при виборі менеджера паролів.

Таблиця 1.2

Критерії менеджерів паролів

Критерії оцінювання	1Password	NordPass	Keeper	Dashlane
Наявність безкоштовної версії	Ні	Так	Ні	Так
Об’єм сховища	1– 5 гб	3 гб	5 гб	1 – 5 гб
Методи автентифікації	2FA, Touch ID, Face ID	Додаткове програмне забезпечення для двофакторної автентифікації	SMS, TOTP, RSA SecurID, DUO Security, U2F, KeeperDNA,	2FA, Touch ID, Face ID, для платної версії U2FA

Продовження таблиці 1.2

Критерії оцінювання	1Password	NordPass	Keeper	Dashlane
Вартість за повну версію на місяць	111 грн	74 грн	111 грн	185 грн
Розширення браузерів	Chrome, Edge, Firefox, Brave	Safari, Chrome, Edge, Firefox, Opera, Brave, Vivaldi	Chrome, Edge, Firefox, Safari, Opera	Safari, Chrome, Edge, Firefox
Підтримка операційних систем	Windows, macOS, Linux, Android, IOS	Windows, macOS, Linux, Android, IOS	Windows, macOS, Linux, Android, IOS	Windows, macOS, Android, IOS
можливість збереження інформації	Так	Ні	Так	Так

1.2. Аналіз загроз та вразливостей.

В цьому пункті, більш детально буде розкриті можливі атаки на пароль, їх принцип роботи, та можливі варіанти їх усунення.

Вважаю потрібним почати наше дослідження з атаки під назвою брутфорс, або атака грубою силою[9],[10].

Головною метою цієї атаки, є отримання секретного ключа користувача. В процесі атаки злочинець підбирає комбінації символів, або підставляє часто вживані паролі.

Є кілька різновидів атак грубої сили:

- Проста атака

Стандартний вид атаки, який розрахований на слабкі паролі, які не відповідають нормам безпеки. В ході атаки перебираються всі можливі примітивні комбінації символів та слів.

Таблиця 1.3.

Статистика використання слабких паролів

Використовувані символи	Кількість створених паролів
Тільки цифри	924,4 млн
Тільки букви	1,4 млрд
Цифри, букви, спеціальні символи	201,7 млн

Часто вживані паролі тільки з цифр:

- 123456

- 33112211
- 1234567890
- 111111

Часто вживані паролі тільки з букв:

- qwerty
- abcde
- password
- iloveyou

Часто вживані паролі з використанням цифр, букв, спеціальних символів:

- P@ssw0rd
- abc123!
- Password1!
- Pass@123

На цьому етапі, стає зрозуміло, що дотримавшись вимог з пункту 1.1, користувач може вберегти свій пароль від простої атаки грубої сили.

- Атака перебором вмісту словника[11]

При такому підході зловмисник, буде використовувати вже готове рішення у вигляді словника. Словник в свою чергу створюється на дослідженнях компаній, або це може бути незаконне викрадення інформації. Тобто вище наведена інформація знаходиться в публічному доступі, з вірогідністю 99,9 % вона вже може знаходитись в словнику.

- Комбінована атака

При такому виді атаки, хакер полегшує собі роботи, додаючи до словника комбінації цифр, букв, спеціальних символів. Таким чином діапазон пошуку збільшується, і цим звісно зростає ймовірність знаходження потрібного паролю. Звісно це не єдиний варіант комбінування атак задля знаходження потрібного паролю. Можна атакувати словник створений на облікових даних користувачів у поєднанні з символами.

- Зворотня атака

Тут використовується кардинально інший підхід. Зловмисник зацікавлений дізнатися більше інформації про жертву. Для цього хакер може моніторити соціальні мережі користувача. Йому необхідно знайти особисту інформацію, яка може перебувати в мережі інтернет.

До такої інформації належить:

- Ім’я, прізвище, дата народження
- Номер телефону, машини, квартири
- Місце роботи та навчання

Зараз надмірна публікація особистої інформації дуже велика проблема. Ми навіть не будемо знати, що за нами хтось стежить та збирає інформації. Зараз кожен користувач соціальних мереж може отримати інформацію, про будь яку людину. Наприклад одною публікацією з відпочинку, ви можете допомогти хатнім злочинцям зрозуміти, що ви зараз не в місті. Про безпеку особистої інформації поговоримо більш детально у фішинг атаках.

Кілька популярних слабких паролів, створених на основі особистої інформації:

- Наташа
- Максим
- Андрій

В деяких випадках користувачі використовують ім’я + дату народження. Потрібно розуміти, що такі паролі, є тільки ілюзією захисту.

- Підстановка облікових даних користувачів

Зловмисник може мати викрадену базу даних з логінами, або буде одночасно підбирати логін та пароль, за допомогою методів, які перераховані вище. Атака продовжується до першої вдалої спроби увійти в обліковий запис користувача. Після цього комбінацію використовують на різних сайтах, розраховуючи, що користувач використовує один пароль для автентифікації.

Зараз можна підсумувати, отриману інформацію, щодо брутфорс атак.

Основні моменти, які повинен знати користувач про атаку грубої сили[11]:

- Час на знаходження потрібного паролю, напряму залежить від його довжини, чим більше різноманітних символів має пароль, тим краще для користувача
- Кількість комбінацій для пошуку паролю. Наприклад користувач створює ненадійний пароль на 8 символів, використовуючи малі літери та цифри. Враховуючи, що в англійському алфавіті 26 літер, 10 символів ми маємо, використовуючи цифри. Підхід такий додаємо кількість символів алфавіту та цифр, результат підносимо до степеню довжини паролю. Виконавши цю роботу, зловмисник буде мати 2821109907456 можливих комбінацій.
- Користувачу може здатися, що кількість комбінацій дуже велика, але потрібно пам’ятати, що обчисленням і перебором займається комп’ютер. Якщо зловмисник має потужний пристрій, кількість виконуваних операцій в секунду може становити кілька млн або млрд. Тобто зловмиснику, для знаходження комбінації, потрібен тільки час.

Як зрозуміти, що ви стали жертвою атаки грубої сили[12]:

- З’явилось багато не зрозумілих спроб увійти у ваш обліковий запис
- Помітне збільшення мережевого трафіку
- Помітне зменшення потужності персонального комп’ютера
- Помітна активність в log файлах

Варіанти забезпечення безпеки від брутфорс атак:

- Головним є дотримання вимог до створення надійного паролю.

Проаналізувавши вище наведену інформацію, стає зрозуміло, що атаки грубої сили добре працюють проти слабких та часто вживаних паролів[13],[14].

- Використання мережевих екранів для фільтрування трафіку
- Систематично змінюйте паролі, та не використовуйте однакові

ключі для різних ресурсів

- Використовуйте якісне антивірусне програмне забезпечення, яке має велику сигнатурну базу даних

Фішинг

Найпопулярніша атака, для отримання інформації користувачів за їх згодою. Досить дивне формулювання речення, але на практиці фішинг на це і розрахований.

Найчастіше такий вид атак можна зустріти, використовуючи електрону пошту, месенджери[15],[16]. Головною метою фішингу є отримання інформацію незаконним способом. Але частіше за все, сам користувач надає свої дані зловмиснику.

Види фішингових атак

- Адресна відправка повідомлень

На відміну від стандартних фішингових атак, які розраховані на масове розсилання. Адресний фішинг, використовується для нападу на окремо вибраних осіб та компаній. При такому методі, зловмисник ретельно готується до атаки, збираючи особисту інформацію жертви. Щоб атака пройшла успішно, потрібно знайти дані, які будуть слабким місцем користувача. Це може бути інтереси, місце роботи та посада, імена колег, тощо. З отриманих даних підбирається контент, який може зацікавити жертву. Ця зацікавленість при необережності може призвести до витоку інформації.

- Атака клонуванням листів

Досить поширений вид атаки. Хакер використовує вже готові шаблони повідомлень, які надходять від офіційних компаній. Частіше за все шахраї прикидаються представниками банку. Хоча звернення може в правду залишитися з офіційного листа, але посилання, чи файл, будуть виконувати вже інше призначення.

Для того, щоб зрозуміти, як може виглядати типове фішингове повідомлення. Я наведу його можливий приклад.

Варто зазначити, що приклад використовується тільки в наукових цілях, та не розрахований, для того, щоб нанести шкоди користувачам. Цей лист не мав підозрілих посилань та файлів. Повідомлення було надіслане з особистого облікового запису на інший мій акаунт.

На рисунку 1.3. наведено приклад можливого фішингового листа

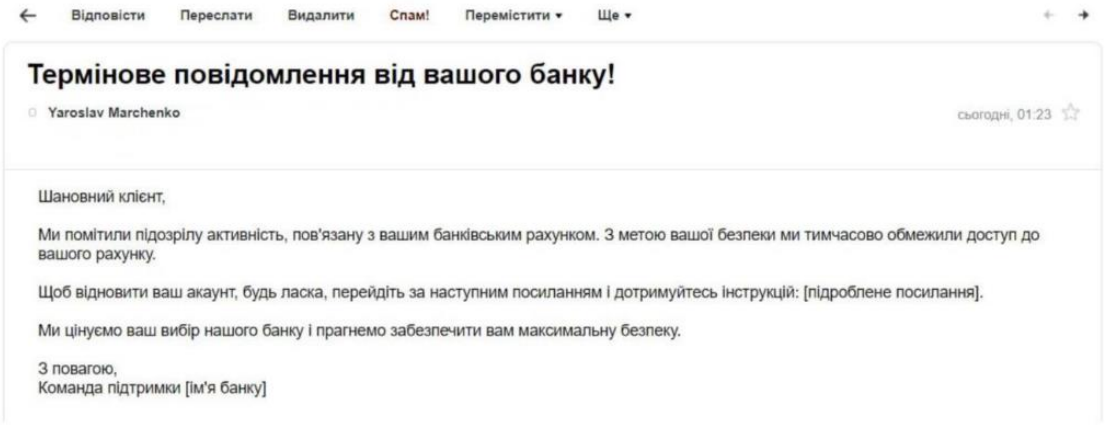


Рис.1.3. Приклад фішинг листа

Щоб точно зрозуміти всю гостроту проблеми. Наведу вже приклад справжнього листа від банку.

На рисунку наведено приклад справжнього листа з банку

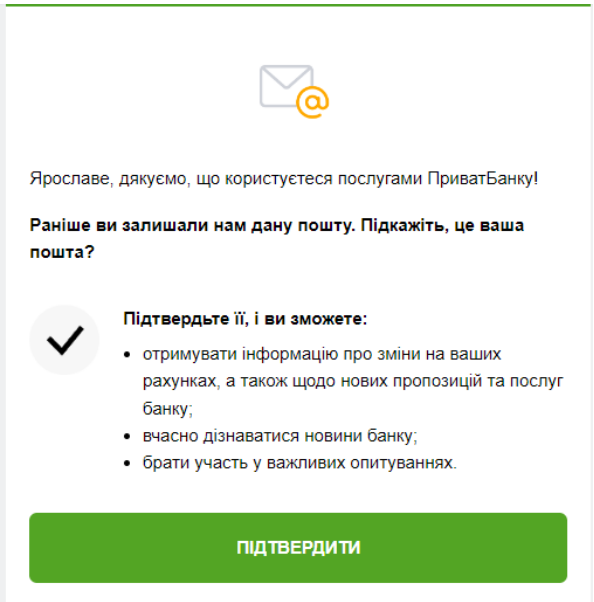


Рис.1.4 Справжній лист від банку

Після цього прикладу, остаточно можна заплутатися, тому що чесно кажучи ці повідомлення виглядають однаково підозріло, і от так одразу складно

сказати, що на рисунку 1.4. представлений офіційний лист від банку. До цього питання повернемося пізніше.

- Фішинг атака 419

Головна мета злочинця викликати співчуття. Зазвичай люди отримують повідомлення з добре продуманою, зазвичай вигаданою історією. Фінал цієї історії, повинен дати зрозуміти користувачу, що ситуація безвихідна. На цьому етапі злочинець починає просити суму грошей, з гарантією повернення та навіть з обіцянками нагороди за допомогу. Частіше за все користувачів просять надати свої реквізити, для повернення коштів. Саме цей запит, використовується, для того, щоб отримати мінімальну довіру від людини. Звісно, ця вся атака проходить в ілюзію поспіху, і це не дає користувачам нормально проаналізувати отриманий лист.

- Фішинг в телефонній розмові та SMS повідомленнях

В останні пару років, саме цей метод став найпопулярнішим. Частіше за все зловмисник прикидається співробітником банку. Сама розмова з користувачем ведеться за стандартним планом, підхід чимось схожий з 419 атакою. Може навіть використовуватися подібний макет, який був наведений на рисунку 1.3. Тобто наявна якась проблема, підозріла активність, компроментація паролю. Але замість посилання чи зараженого файлу, для вирішення вигаданої проблеми, хакер намагається отримати дані карток, чи інформацію, яка потрібна для автентифікації у застосунку банку. В основному для цієї атаки використовують викрадені бази даних, тому зловмисник може знати ваше ім’я та прізвище, та ще деяку особисту інформації. Скажу з особистого досвіду, що банки дуже рідко телефонують своїм клієнтам.

Схожий принцип використовується з sms повідомленнями. Злочинець може запропонувати перейти за посиланням, заповнити форму, чи пройти автентифікацію на підозрілому сайті.

Методи визначення фішингових атак:

- В повідомлені присутній негайний заклик до дії

Повідомлення може мати такий вигляд: Ваш акаунт було зламано, чи ми помітили підозрілу активність. Часто до такого листа додають посилання, чи навіть заражені файли. Перед ти, як реагувати на таке повідомлення, отримайте всю можливу інформацію, про відправника. Якщо це компанія, перейдіть на офіційний сайт, та пошукайте контактну інформацію. Часто на сайті надається номер телефону та адреса електронної пошти. Зв'яжіться з офіційним представником для уточнення інформації.

- В повідомленні присутній заготовлений макет

Якщо ви отримали повідомлення, дуже нагадує те, яке було наведене на рисунку 1.3. Це скоріш за все фішинг. Якщо це банк, то пошукайте додаткову інформацію. Але зі свого досвіду можу сказати, що банки не відправляють повідомлення, а тим більш не просять перейти за посиланням, чи надати додаткову особисту інформацію.

- В повідомлені наявні посилання чи файли

Не відкривайте файли, чи посилання, якщо ви не впевнені у відправнику. Скористуйтеся Virustotal, для отримання додаткової інформації про вміст повідомлення.

Ще одним методом для незаконного отримання інформації є Кейлогери.

Зазвичай, це програмне забезпечення, задача якого, зчитувати натискання клавіш та записувати їх у log – файл. З яким потім може ознайомитись зловмисник.

Місце зберігання файлу з інформацією:

- Жорсткий диск пристрою
- В пам'яті

Методи передачі файлу:

- З використанням електронного листа
- Протоколи ftp, http, https
- Bluetooth, Wi-fi

Принцип роботи кейлогерів зображено на рисунку 1.5.

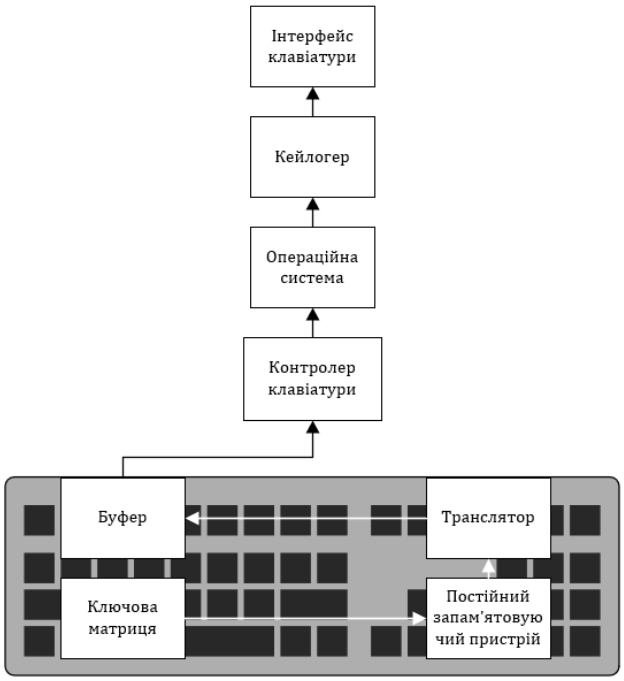


Рис. 1.5. Принцип роботи кейлогерів

Кейлогер у вигляді апаратного пристрої, можна зустріти не часто. Задача такого методу не відрізняється від програмного.

За методом використання:

- Несанкціоноване

Продукт встановлюється в секреті від користувача комп’ютеру.

- Санкціоноване

В цьому випадку, перед установкою, було сповіщено користувача комп’ютеру для отримання дозволі.

Методи захисту від кейлогерів:

- Використання якісних та перевірених антивірусних програм, для моніторингу операційної системи
- Систематично виконувати апаратну перевірку комп’ютеру

1.3. Дослідження методів захисту паролів користувачів. Криптографічні підходи і методи гешування.

Один з найпоширеніших способів захисту паролів, полягає у їх гешуванні[21],[22]. При цьому пароль перетворюється у вигляд, який неможливо просто відновити до вхідного значення. В цьому пункті, я розгляну поширені алгоритми хешування, такі як MD5 та SHA-1.

Всі отримані дані від користувача зберігаються у базі даних у вигляді Хешу. Для підвищення безпеки додатково використовують сіль.

Основні вимоги до Хеш-функцій:

- Функція повинна перетворювати інформацію будь-якого розміру, в рядок фіксованої довжини
- Функція повинна мати лавинний ефект

Це означає, що при зміні значення аргументу, функція повинна значно змінюватись.

- Опір колізіям

Це така ситуація, коли різні вхідні дані при хешуванні дають однакове хеш значення.

- Детермінованість

Інформація при повторному хешуванні, повинна давати один і той же хеш.

- Однонаправленість

Неможливо знаючи хеш отримати вхідні дані. Тобто розшифрувати хеш, та отримати початку або вхідну інформації практично неможливо.

Алгоритми хешування:

MD5

Розмір виходу становить: 128

Розмір внутрішнього стану становить: 128

Розмір блоку становить: 512

Довжина становить: 64

Довжина слова становить: 32

Кількість раундів: 64

Атаки на алгоритм MD5:

- День народження $2^{20.96}$

Цей вид атаки базується на тому, що існує ймовірність виникнення колізій, оскільки на виході в md5, ми отримуємо дайджест повідомлення розміром 128 біт, то ймовірність знаходження колізії $2^{20.96}$

- Знаходження прообразу $2^{123.4}$

Маючи значення хешу h зловмисник прагне знайти таку вхідну інформацію m, яка при хешуванні, на виході буде мати значення хешу, який вже є у зловмисника $hash(m) = h$.

- Знаходження другого прообразу $2^{123.4}$

Цей спосіб полягає в знаходженні колізії, тобто маючи одне вхідне значення $m1$, зловмисник перевіряє, чи немає ще одного вхідного значення $m2$, хеш якого буде дорівнювати першому вхідному значенню $hash(m1) = hash(m2)$.

Логіка виконання MD5:

На рисунку 1.6. зображена схема роботи алгоритму MD5

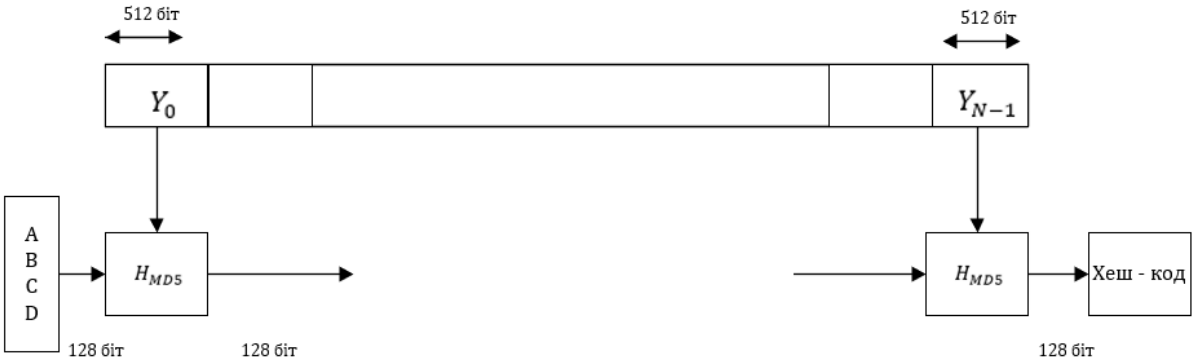


Рис. 1.6. Логіка роботи MD5

Роботу алгоритму, можна розділити на кілька кроків:

1) Додавання невистачаючих бітів для отримання потрібної довжини

В першому кроці, відбувається додавання бітів до повідомлення, таким чином, щоб довжина стала кратною 448 mod 512. Додавання проходить, навіть при умові, якщо ми вже маємо повідомлення потрібної довжини.

За результатами першого кроку, ми отримуємо розширене повідомлення, у якого довжина кратна 448 mod 512. Далі це повідомлення ділиться на блоки від Y_0 до Y_{N-1} розмір цих блоків 512 біт, з цього виходить, що довжина повідомлення становить $N \cdot 512$.

2) Використання MD-буфера для зберігання проміжних результатів

В алгоритмі MD5 використовується буфер розміром 128 біт, для зберігання результатів при обробці блоків. Буфери в md5 представляють собою чотири регістри розміром в 32 біта.

В основному випадку початкові значення регістрів будуть константами, та визначаються стандартом. Значенням кожного регістру є шістнадцяткові числа.

Початкове значення буферів md5 має такий вигляд:

A = 01 23 45 67

B = 89 AB CD EF

C = FE DC BA 98

D = 76 54 32 10

Принцип обробки блоків змінюється в залежності від вибраного алгоритму.

Тобто при обробці першого блоку, на виході ми отримаємо значення, яке поміщується в буфер, і при обробці наступного блоку вихідні значення першого блоку, будуть вхідними значеннями для наступного блоку. На початку наступного раунду значення буфера будуть оновлюватись Цей принцип забезпечує потокову обробку даних, в нашому випадку обробка відбувається блоками. При обробці останнього блоку, на виході ми отримуємо дайджест вхідного повідомлення, тобто хеш значення.

3) Обробка отриманих блоків

Для обробки блоку алгоритм використовує, модуль в якому містяться чотири циклічні обробки. Кожен з цих циклів має свою елементарну функцію:

$$f_F = (B \,\&\, C)(not\ B \,\&\, D)$$

$$f_G = (B \,\&\, D) \vee (C \,\&\, not\ D)$$

$$f_H = B \oplus C \oplus D$$

$$f_I = C \oplus (B \,\&\, not\ D)$$

При обчислені блоку кожна функція гарантовано отримає по три слова розміром 32 біта, як результат, ми на виході отримуємо слово розміром 32 біта.

Елементарні функції використовують такі бітові операції (AND, OR, XOR). З цього стає зрозуміло, що n-ний біт отриманого виходу являється функцією від n-них бітів трьох поданих раніше слів.

4) Отримання дайджесту повідомлення

Результатом обробки останнього блоку, є дайджест повідомлення розміром 128 біт, тобто в кінці, ми отримуємо потрібне нам хеш значення.

На рисунку 1.7 наведений хеш, отриманий за допомогою алгоритму MD5.

```
Qwerty12345
Пароль: Qwerty12345
Хеш паролю (MD5): d137043cbd9a0f3eddf22667c934959f
```

Рис 1.7. Хешування алгоритмом MD5

Потрібно зауважити, що алгоритми (MD5, SHA-1) мають в основі однакову структуру, принцип отримання дайджесту повідомлення залишається один і той же, але буде різниця в значеннях алгоритмів, та в підході обробки блоку.

SHA-1

Розмір виходу становить: 160

Розмір внутрішнього стану становить: 160

Розмір блоку становить: 512

Довжина становить: 64

Довжина слова становить: 32

Кількість раундів: 80

Атаки на алгоритм SHA-1:

- День народження 2^{51}

Цей вид атаки базується на тому, що існує ймовірність виникнення колізій, оскільки на виході в sha-1, ми отримуємо дайджест повідомлення розміром 160 біт, то ймовірність знаходження колізії 2^{51}

У порівнянні з MD5 алгоритм SHA-1 не є вразливим до атак знаходження прообразу та знаходження другого прообразу.

Логіка виконання SHA-1:

На рисунку 1.8. зображена схема роботи алгоритму SHA-1

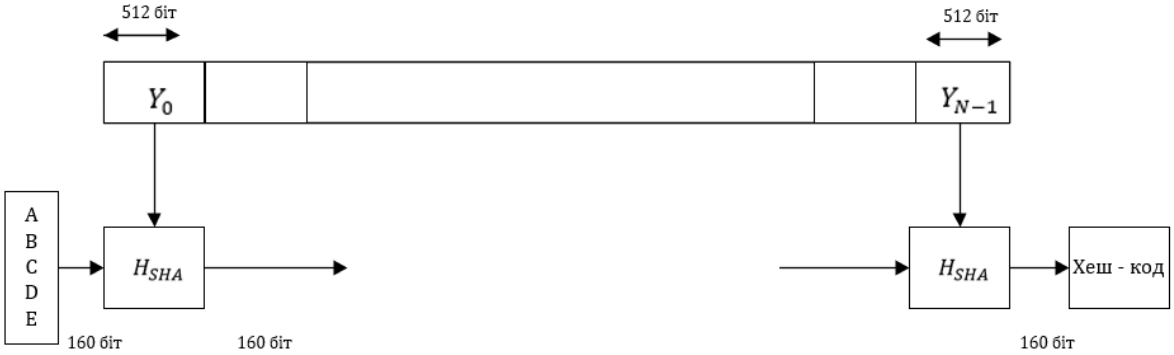


Рис. 1.8. Логіка роботи SHA-1

Роботу алгоритму, можна розділити на кілька кроків:

1) Додавання невистачаючих бітів для отримання потрібної довжини

В першому кроці, відбувається додавання бітів до повідомлення, таким чином, щоб довжина стала кратною 448 mod 512. Додавання проходить, навіть при умові, якщо ми вже маємо повідомлення потрібної довжини.

За результатами першого кроку, ми отримуємо розширене повідомлення, у якого довжина кратна 448 mod 512. Далі це повідомлення ділиться на блоки від Y_0 до Y_{N-1} розмір цих блоків 512 біт, з цього виходить, що довжина повідомлення становить $N \cdot 512$.

2) Використання SHA-1 буфера для зберігання проміжних результатів

В алгоритмі SHA-1 використовується буфер розміром 160 біт, для зберігання результатів при обробці блоків. Буфери в sha-1 представляють собою п’ять регістри розміром в 32 біта.

В основному випадку початкові значення регістрів будуть константами, та визначаються стандартом. Значенням кожного регістру є шістнадцяткові числа.

A = 67 45 23 01

B = EF CD AB 89

C = 98 BA DC FE

D = 10 32 54 72

E = C3 D2 E1 F0

На вхід першого раунду, в алгоритмі SHA- 1 подається вектор ініціалізації. Після ініціалізації вектора, його результат об'єднується з іншими даними:

$$SHA_0 = A \| B \| C \| D \| E$$

3) Обробка отриманих блоків

При обробці кожен з циклів отримує блок Y_q на вхід, та значення буфера для SHA-1 ABCDE, розмір якого становить 160 біт, після цього змінюється значення буфера.

В основі алгоритму знаходиться модуль, який складається з 80 циклічних обробок. Всі обробки мають однакову структуру. Для отримання SHA_{q+1}, вихід останнього циклу складається із значень SHA_q. Додавання по модулю 232, буде виконуватися для слова в буфері з відповідним словом в SHA_q.

Однією з особливостей алгоритму SHA-1, є використання константи K_t, в свою чергу константа, може приймати чотири значення в залежності від номеру цикла.

$$\begin{cases} 2^{30} \times \sqrt{2}, & 0 \le t \le 19 \\ 2^{30} \times \sqrt{3}, & 20 \le t \le 39 \\ 2^{30} \times \sqrt{5}, & 40 \le t \le 59 \\ 2^{30} \times \sqrt{10}, & 60 \le t \le 79 \end{cases}$$

Логічні функції, які використовуються в кожному раунді

Функції використовуються в залежності від номера цикла:

$$f_t(B,C,D) = \begin{cases} (B \wedge C) \vee (\overline{B} \wedge D), & 0 \le t \le 19 \\ B \oplus C \oplus D, & 20 \le t \le 39, 60 \le t \le 79 \\ (B \wedge C) \vee (BD) \vee (C \wedge D), & 40 \le t \le 59 \end{cases}$$

1) Отримання дайджесту повідомлення

Результатом обробки останнього блоку, є дайджест повідомлення розміром 160 біт, тобто в кінці, ми отримуємо потрібне нам хеш значення.

На рисунку 1.9. наведений хеш, отриманий за допомогою алгоритму SHA-1.

```
Qwerty12345
Пароль: Qwerty12345
Хеш паролю (SHA-1): b651576965c77a1bd2f2a373cf9a4e09f8ad5fe1
```

Рис 1.9. Хешування алгоритмом SHA-1

В таблиці 1.4. буде наведена порівняльна інформація, основного функціоналу наведених алгоритмів хешування.

Таблиця 1.4

Порівняння алгоритмів MD5 та SHA1

Основні критерії	MD5	SHA- 1
Розмір виходу	128	160
Розмір внутрішнього стану	128	160
Розмір блоку	512	512
Розмір слова	32	32
Довжина	64	64

Продовження таблиці 1.4

Основні критерії	MD5	SHA- 1
Кількість раундів	64	80

Зараз ми поговоримо про правильність вибору функцій, з урахуванням часу життя функцій хешування.

В таблиці 1.5 наведено життєвий цикл відомих хеш функцій.

Таблиця 1.5

Життєвий цикл хеш функцій

Функції хешування	Доступний для оцінки	Достатньо сильна	незначна слабкість	Функція Ослаблена	зламана	Знайдена колізія
MD2 (128 біт)	-	-	1990 рік	1995 рік	2004 рік	-
MD4	-	1990 рік	-	1991 рік	-	1995 рік
MD5	1991 рік	1992 рік	1993 рік	1996 рік	-	2004 рік
SHA-0	-	1993 рік	-	1998 рік	-	2004 рік
SHA-1	-	1995 рік	-	2004 рік	-	2017 рік
SHA-2 family	-	2000 рік	2008 рік	-	-	-
RepiMD 128	-	1992 рік	-	1997 рік	-	2004 рік
RepiMD160	-	1996 рік	2013 рік	-	-	-

В таблиці - означає відсутність етапу. За допомогою цієї таблиці, ми можемо побачити всю картину. Звісно потрібно зазначити, що для розробки модуля, потрібно використовувати надійні функції хешування. Для цього можна роздивитися SHA-2 family. Представниками є SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/256 и SHA-512/224.

Для того, щоб зрозуміти, як використовується хешування у процесі автентифікації наведу приклад рисунку 1.11.

Рис. 1.11. Процес автентифікації користувача

Про формування солі, її безпосереднє застосування на прикладах, звісно поговоримо про її види, їх доцільність використання вже в другому розділі дипломної роботи

1.4. Висновки до першого розділу.

В ході написання першого розділу, мною були розкриті основні питання, які стосуються паролю:

- Створення надійного паролю, методи його зберігання
- Основні атаки на пароль, та методи їх запобігання
- Криптографічні підходи захисту пароля

Цей розділ, є фундаментом цієї роботи. Без розуміння вище наведених пунктів, не можливе забезпечення безпеки паролів. Основні моменти, які повинні запам’ятатися після прочитання наведеної інформації. Перш за все,

користувач несе відповідальність за безпеку своїх даних, тому що створення надійного паролю, не вимагає багато часу, а при дотримані вимог, ви вже самостійно зможете захиститися від атак грубої сили. Не діліться особистою інформацією, з людьми або сайтами, які викликають у вас підозрілість. Не бійтеся, шукати додаткову інформацію про сайт, або користувача. Уникайте підозрілі посилання та документи. Перед відкриттям перевірте файли за допомогою Virustotal. І звісно, знання цієї інформації допоможе у реалізації криптографічного модулю гешування паролів в реальному часі.

РОЗДІЛ 2.
ТЕОРЕТИЧНИЙ ОПИС МЕТОДІВ ХЕШУВАННЯ

В цьому розділі мною буде надана теоретична інформація, яка стосується безпосередньо реалізації модуля гешування паролів. В розділі будуть розкриті такі поняття, як : використання солі з розрахунком уникнення радужних таблиць та підвищення безпеки хешу, радужні таблиці в цілому, хеш таблиці, хеш-функції на основі блокових шифрів, тобто MAC коди.

2.1 Використання солі для підвищення стійкості до атак грубої сили та райдужних таблиць.

Сіль, це невід'ємна частина процесу автентифікації. Почнемо з очевидного, при створені облікового запису, користувач обов'язково створює пароль для захисту даних, та повторного доступу до аккаунта. Створений пароль зберігається в базі даних, на цьому етапі начебто все, просто бери і при автентифікації перевіряй введений пароль зі збереженим в базі, але ні. Пароль не може зберігатися в первинному вигляді, тому що при вкрадені бази даних, захисту облікових даних просто немає. Але рішення є, перед збереженням секретний ключ хешується, за допомогою наприклад MD5, SHA-1. Це ідеальний підхід для збереження паролів, тому що хеш функції для цього мають дві головні властивості:

- Детермінованість

Інформація при повторному хешуванні, повинна давати один і той же хеш.

- Однонаправленість

Неможливо знаючи хеш отримати вхідні дані. Тобто розшифрувати хеш, та отримати початку або вхідну інформації практично неможливо.

Але зловмисники знайшли способи отримання вхідної інформації маючи хеш.

Для уникнення таких сценаріїв, та підвищення безпеки використовується сіль.

В основному сіль використовують, для підняття стійкості хеш значення до атак райдужними таблицями.

Таблиці побудовані на принципі створення ланцюгів, тобто головною задачею таблиці є знаходження прообраз. Початком ланцюгів є пароль який хешується, за допомогою вибраного алгоритму. Функцією редукції отримане значення зіставляється з множиною паролів.

На рисунку 2.1 наведено приклад ланцюжку.

$$\mathbf{aaaaaa} \underset{H}{\longrightarrow} 281\mathrm{DAF40} \underset{R}{\longrightarrow} \mathrm{sgfnyd} \underset{H}{\longrightarrow} 920\mathrm{ECF10} \underset{R}{\longrightarrow} \mathbf{kiebgt}$$

Рис. 2.1. Створення ланцюжку в радужних таблицях

Опишу робота райдужних таблиць на наведеному прикладі. Припустимо, що ми маємо хеш 920ECF10, який при зіставлені з множиною паролів дає kiebgt.

$$920\mathrm{ECF10} \underset{R}{\longrightarrow} \mathbf{kiebgt}$$

Рис. 2.2 Зіставлення значення хешу з множиною паролів

Наступним етапом є перевірка ланцюжків на наявність отриманого значення.

В моєму випадку це значення є кінцем ланцюжку. Потрібний нам ланцюжок знайдено, наступним кроком, ми беремо початкове значення цього ланцюжку, та відновлюємо безпосередньо ланцюжок, допоки не отримаємо потрібне нам хеш значення.

$$\mathbf{aaaaaa} \underset{H}{\longrightarrow} 281\mathrm{DAF40} \underset{R}{\longrightarrow} \mathrm{sgfnyd} \underset{H}{\longrightarrow} 920\mathrm{ECF10}$$

Рис. 2.3. Відновлення ланцюжку для отримання паролю.

В цьому прикладі наше хеш значення це результат хешування пароля sgfnyd.

Побудова райдужних таблиць можлива, за допомогою властивості детермінованості.

При побудові райдужних таблиць, зловмисник повинен дотримуватися деяких параметрів. В таблиці 2.1 наведені параметри, які необхідні при створенні таблиць:

Таблиця 2.1

Параметри для створення райдужних таблиць

Параметри	Варіант 1	Варіант 2	Варіант 3	Варіант 4
Комбінації символів пароля	Створюється на основі букв (26)	Створюється на основі букв та цифр (36)	Створюється на основі букв та цифр, символи (50)	Створюється на основі всіх комбінацій
Довжина ланцюжку (кількість хешів)	2100	2400	12000	20000
Лічильник ланцюжків	8000000	40000000	40000000	100000000
Кількість таблиць	5	7	13	20
Ймовірність знаходження прообразу	99,9%	99,9%	99,9%	99,3%
Необхідне місце на диску	640 мб	4480 мб / 4,48 гб	8320 мб / 8,32 гб	32000 мб / 32 гб

Продовження таблиці 2.1

Параметри	Варіант 1	Варіант 2	Варіант 3	Варіант 4
Необхідний час на генерацію	17 годин	5 днів 14 годин	52 дні	332 дні
Необхідний час на знаходження прообразу	7 секунд	14 секунд	11 хвилин	48 хвилин

Після цієї таблиці, стає зрозумілим, що процес створення райдужних таблиць є досить кропіткою роботою, яка вимагає багато часу. Наприклад при використані таблиць з повним набором символів, для знаходження вхідної інформації потребується 48 хвилин, що в свою чергу досить швидко. Атакою грубої сили, знаходження паролю буде становити приблизно три тижні.

І для того, щоб уникнути атаку стандартними райдужними таблицями разом з паролем хешують сіль.

Сіль поділяють на статичну та динамічну:

- Статична сіль

Особливість цієї солі, полягає в тому, що вона генерується один раз і в подальшому використовується для всіх паролів. Перевагою такої солі є простота реалізації, тобто у розробника відпадає потреба кожного разу генерувати унікальну соль та зберігати її. Статична соль вирішує проблему з радужними таблицями. При цьому слід розуміти, що при викрадені бази даних, зловмисник все ж таки може отримати значення солі, це не так просто, але можливо. В такому випадку безпека облікових записів стрімко падає до нуля, тому що маючи значення солі хакер може створити райдужну таблицю на основі отриманого

значення, або скористатися атакою брутфорс, додаючи до можливих варіантів пароля значення солі.

- Динамічна сіль

На відміну від статичної солі, динамічна є унікальною для кожного паролю, тобто це і є ідеальне рішення, для забезпечення безпеки хешу. При цьому, якщо зловмисник і отримає значення солі, то в такому випадку райдужна таблиця буде створюватись для знаходження тільки одного паролю. З цього можна зрозуміти, що динамічна сіль робить атаку райдужною таблицею просто не ефективною.

Варто зазначити, що досить велику роль грає правильність зберігання солі. Значення солі повинно залишатися секретно, тобто в жодному разі не можна сіль в окремій комірці бд. Головним варіантом є збереження солі за принципом hash(salt + pass), ще можна зберігати солі в окремій бд або зберігати значення солі за межами бази даних, останній варіант варто роздивитися, якщо хочете використовувати статичну сіль.

А зараз ми на практиці подивимось принцип роботи райдужної таблиці. Варто вказати, що цей дослід не несе на меті завдати шкоди, а виконується тільки в наукових цілях. Для хешування паролів буду використовувати функції MD5, SHA-1, SHA-256. Для початку візьму три популярні паролі за статистикою компанії NordPass (password, 123456, qwerty).

На рисунку 2.4. наведено хешування password, хеш-функцією MD5

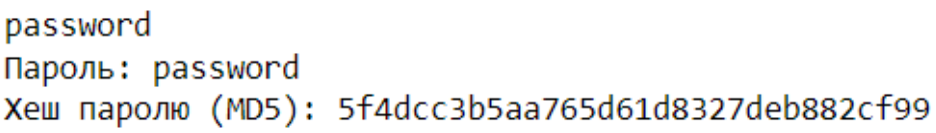


Рис. 2.4. Хешування password функцією MD5

Наступним кроком, перевіряємо наявність отриманого хешу в райдужній таблиці.

На рисунку 2.5. наведено приклад пошуку значення паролю.

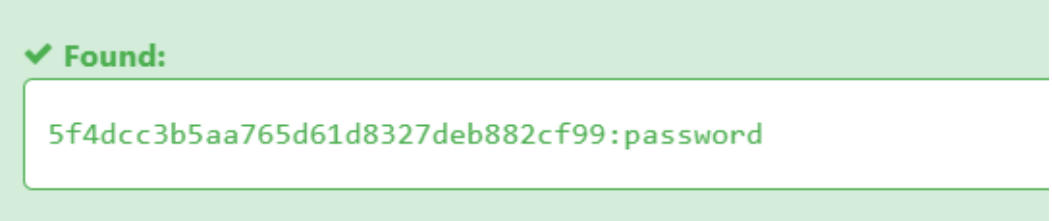


Рис. 2.5. Пошук прообразу

Звісно отриманий результат, не дивує. В першу чергу, потрібно розуміти, що в даному випадку, паролі не відповідають мінімальним вимогам безпеки. Додавши до пароля великі літери, спеціальні символи та цифри, ми можемо значно змінити результат, але головне на чому хочеться наголосити, використовуйте унікальний пароль, це безумовно зменшить ймовірність його потрапляння до райдужних таблиць.

На рисунку 2.6. наведено хешування 123456, хеш-функцією SHA-1

```
123456
Пароль: 123456
Хеш паролю (SHA-1): 7c4a8d09ca3762af61e59520943dc26494f8941b
```

Рис. 2.6. Хешування password функцією SHA-1

На рисунку 2.7. наведено приклад пошуку значення паролю.

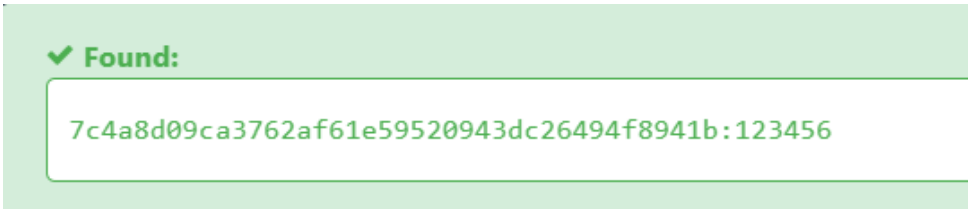


Рис. 2.7. Пошук прообразу

І останнім протестуємо хеш-значення SHA-256.

На рисунку 2.8. наведено хешування qwerty, хеш-функцією SHA-256.

```
qwerty
Пароль: qwerty
Хеш паролю (SHA-256): 65e84be33532fb784c48129675f9eff3a682b27168c0ea744b2cf58ee02337c5
```

Рис. 2.8. Хешування password функцією SHA-256

На рисунку 2.9. наведено приклад пошуку значення паролю.

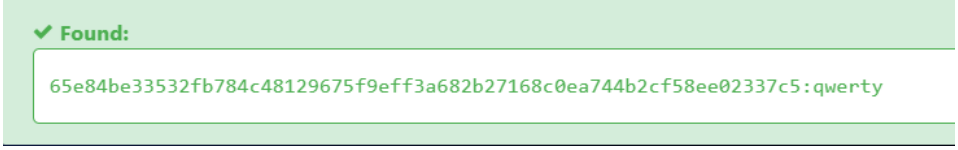


Рис. 2.9. Пошук прообразу

Чесно кажучи результати дуже очевидні. Тому наступним кроком згенеруємо паролі, які відповідають всім вимогам надійного паролю.

На рисунку 2.10. наведено хешування ['$+TmxC_4DZnM'], хеш-функцією MD5.

```
['$+TmxC_4DZnM']
Пароль: ['$+TmxC_4DZnM']
Хеш паролю (MD5): 361d094fe0c664389c324a9301a837b4
```

Рис. 2.10. Хешування ['$+TmxC_4DZnM'] функцією MD5

На рисунку 2.11. наведено приклад пошуку значення паролю.

Рис. 2.11. Пошук прообразу

Потрібно зазначити, що сайт не може надати повноцінний функціонал ра йдужної таблиці, він розрахований скоріше на слабкі паролі. Хеш-значення зген ерованого паролю ['$+TmxC_4DZnM'] не було знайдено в райдужній таблиці, ал е це не означає, що цього досить. У відкритому достіпі є безліч варіантів райдуж них таблиць, які можна завантажити та провести атаку. Тобто використання сол і є обов'язковим пунктом. На рисунку 2.12 наведені варіанти райдужної таблиці у відкритому доступі.

весь простір#1-7 [2]	
алфавіт #1-1, нижній алфавіт # 5-5, нижній буквено-цифровий # 2-2, цифровий # 1-3	362 ГБ: 0 1 2 3
альфа-простір №1-9	35 ГБ: 0 1 2 3
lm-frt-cp437-850#1-7	
нижчий альфа#1-10	
нижня буква#7-7, числова #1-3	26 ГБ: 0 1 2 3
нижні алфавітно-цифрові #1-10	587 ГБ: 0 8 16 24
нижній алфавітно-цифровий пробіл#1-8	15 ГБ: 0 1 2 3
нижній алфавітно-цифровий пробіл#1-9	
loweralpha-numeric-symbol32-space#1-7	33 ГБ: 0 1 2 3
loweralpha-numeric-symbol32-space#1-8	428 ГБ: 0 1 2 3
нижній альфа-простір №1-9	35 ГБ: 0 1 2 3
мікс буквено-цифровий №1-8	274 ГБ: 0 1 2 3
мікс буквено-цифровий №1-9	1 ТБ: 0 16 32 48
mixalpha-numeric-space#1-7	17 ГБ: 0 1 2 3
mixalpha-numeric-space#1-8	
mixalpha-numeric-symbol32-space#1-7 [3]	86 ГБ: 0 1 2 3
mixalpha-numeric-symbol32-space#1-8 [3]	1 ТБ: 0 8 16 24 32
числові №1-12	
числові №1-14	

Рис. 2.12. Райдужні таблиці у відкритому доступі

З таким підходом, повернувшись до таблиці 2.1, ми можемо побачити, що ймовірність знаходження прообразу (['$+TmxC_4DZnM']) маючи хеш- значення (361d094fe0c664389c324a9301a837b4) становить 99,3%.

На рисунку 2.13. наведено хешування U50I9ZY#Tsns, хеш-функцією SHA-1 .

```
U50I9ZY#Tsns
Пароль: U50I9ZY#Tsns
Хеш паролю (SHA-1): 5f87301d033e42b15f8c2436452a666b66370100
```

Рис. 2.13. Хешування U50I9ZY#Tsns функцією SHA-1

На рисунку 2.14. наведено приклад пошуку значення паролю.

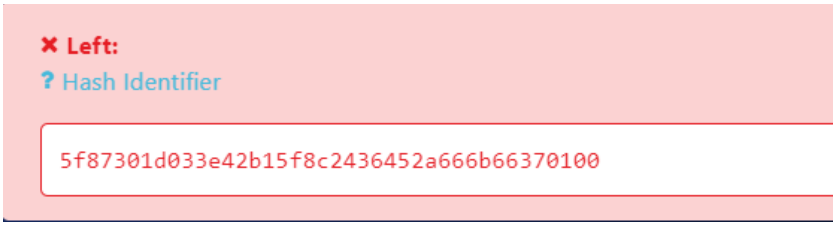


Рис. 2.14. Пошук прообразу

На рисунку 2.15. наведено хешування C4L=pKOYfug, хеш-функцією SHA-256.

```
C4L=pKOYfug
Пароль: C4L=pKOYfug
Хеш паролю (SHA-256): ec72f892d331f9b885b1b8dbd7bce6281fcbff368bc32730fd5f25b1434c7826
```

Рис. 2.15. Хешування C4L=pKOYfug функцією SHA-256

На рисунку 2.16. наведено приклад пошуку значення паролю.

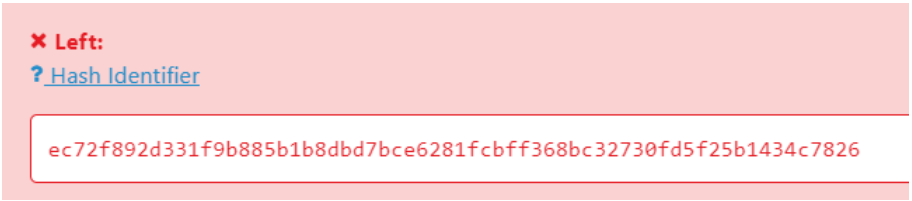


Рис. 2.16. Пошук прообразу

По отриманим результатам, можна точно сказати, що таблиці, які використовуються в досліді, створені з використанням не всіх символі, тобто Тобто повернувшись до таблиці 2.1, райдужна таблиця створена на основі варіанту 1 – 3. Єдине, що потрібно розуміти, це те, що при створені райдужної таблиці, зловмисник буде використовувати всі символи, і тоді знаходження

прообразуес72f892d331f9b885b1b8dbd7bce6281fcbff368bc32730fd5f25b1434c782
6 тільки питання часу.

Так як ми маємо хеш-значення, які були знайдені в таблиці, ми спробуємо захистити хеш додаванням до нього солі.

За допомогою функції MD5, ми хешували пароль password, та отримали значення хешу: 5f4dcc3b5aa765d61d8327deb882cf99. На рисунку 2.17. продемонстровано додавання солі до паролю.

```
Пароль: password
Сіль: d474c04cd8be5ca3477b73d8cb8c2f7c
Хеш паролю з соллю(MD5): b7bb2053689dfc7c0faca27282eb9447
```

Рис. 2.17. Додавання солі до паролю

Наступним кроком шукаємо в райдужній таблиці пароль, який дає на виході отримане раніше хеш-значення. На рисунку 2.18. наведено результат пошуку.

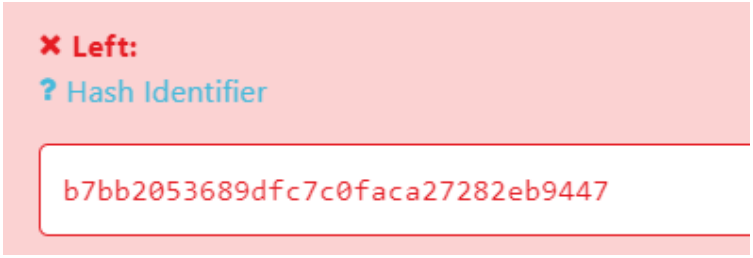


Рис. 2.18. Пошук прообразу

Функцією SHA-1, ми хешували пароль 123456, та отримали значення хешу: 7c4a8d09ca3762af61e59520943dc26494f8941b. На рисунку 2.19. продемонстровано додавання солі до паролю.

```
Пароль: 123456
Сіль: fa2cde872775e5d21a7f661a1682f015
Хеш паролю з соллю(SHA-1): 1af660438cb34b1e04ef3e294255f8b2e8cdf1df
```

Рис. 2.19. Додавання солі до паролю

Шукаємо в райдужній таблиці пароль, який дає на виході отримане раніше хеш-значення. На рисунку 2.20. наведено результат пошуку.

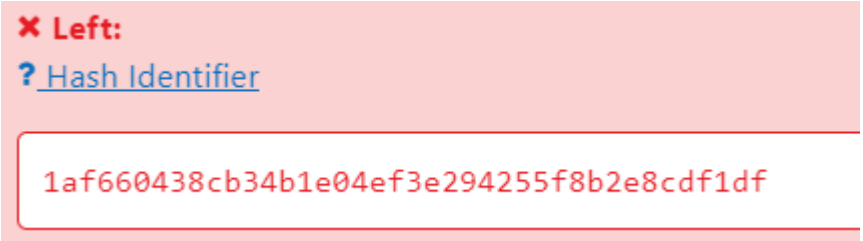


Рис. 2.20. Пошук прообразу

І остання функція SHA-256, ми хешували пароль qwerty, та отримали хеш:65e84be33532fb784c48129675f9eff3a682b27168c0ea744b2cf58ee02337c5. На рисунку 2.21. продемонстровано додавання солі до паролю.

```
Пароль: qwerty
Сіль: 9992012b42f2f1a5cc39eded886d9676
Хеш паролю з соллю(SHA-256): ae3975027795e16535af1d47a29ef86b81747c21592b98618bc9b7be011ba08e
```

Рис. 2.21. Додавання солі до паролю

Шукаємо в райдужній таблиці пароль, який дає на виході отримане раніше хеш-значення. На рисунку 2.22. наведено результат пошуку.

Рис. 2.22. Пошук прообразу

З отриманих результатів, можна побачити, як добре сіль ламає весь принцип роботи райдужних таблиць. В цих прикладах використовувалась динамічна сіль. Використання динамічної сілі, уникає очевидності використання одного паролю кількома юзерами. На рисунку 2.23 наведений головний недолік використання статичної солі.

```
qwerty12345
Хеш паролю зі статичною сіллю: 6e76e81a756e432d39525ff8d3ab36fd
```

Рис 2.23. Хешування паролю за допомогою статичної солі

За правилами автентифікації, у користувачів не може бути однаковий логін, що не можна сказати про пароль. І так уявімо, два користувачі, не дуже ознайомлені з проблемою безпеки паролів, і при реєстрації створили пароль qwerty12345, який навіть близько не відповідає вимогам безпеки. І звісно модуль

гешування паролів використовує статичну сіль. На рисунку 2.24 наведена повторна реєстрація паролю qwerty12345.

```
qwerty12345
Хеш паролю зі статичною сіллю: 6e76e81a756e432d39525ff8d3ab36fd
```

Рис 2.24. Хешування однакових паролів за допомогою статичної солі.

Звісно, це не означає, що статична сіль не виконує своє завдання. Це можна побачити, за допомогою райдужної таблиці. На рисунку 2.25. наведено результат пошуку.

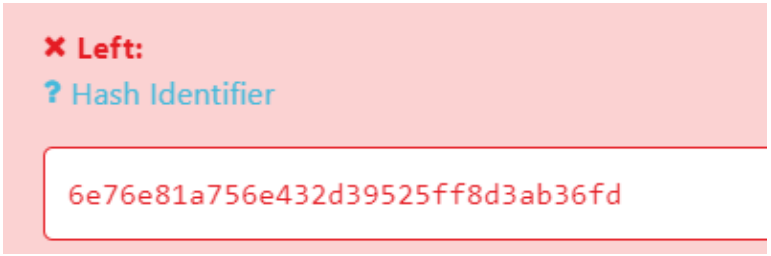


Рис. 2.25. Пошук прообразу

Значення хешу не було знайдено в таблиці, і ось здається все робота виконана. Але при отримані бд, зловмисник обов'язково будуть досліджувати отриману інформацію. І при знаходженні однакового хешу, зловмисникам може допустити, що при хешуванні використовується статична сіль. Що в свою чергу значно полегшує роботу. Після цього головна задача зловмисника зводиться до отримання значення солі.

Для отримання значення солі, та даних в цілому хакери можуть використовувати різноманітні методи :

- Перехоплення трафіку

В цьому випадку зловмисники користуються відсутністю безпечного підключення на сайті, тобто зв'язок між сервером та клієнтом не зашифрований та перехоплюють трафік, це може статися, якщо використовується протокол HTTP замість HTTPS.

- Дослідження вихідного коду

Безпосередній пошук місця зберігання солі в програмному коді.

- Атака людина посередині

Вид атаки, коли зловмисник, поміщає себе між сервером та клієнтом, та має змогу повністю фільтрувати потік. До можливостей цієї атаки, можна віднести:

- Можливість перехоплювати пакети
- Імітація зв’язку між клієнтом та сервером
- Змога вносити зміни в передачу даних
- Змога видаляти дані

Щоб уникнути цю атаку, варто використовувати надійні протоколи для передачі даних, як приклад HTTPS.

2.2 Коди автентифікації повідомлень або MAC коди

Головною задачею кодів автентифікації повідомлення, MAC є забезпечення цілісності відправленого повідомлення. Якщо при відправці, повідомлення було змінене, то одержувач, це неодмінно помітить. Автентифікація, в свою чергу забезпечує перевірку кількох пунктів:

- Інформація, була надіслана законним відправником;
- Одержувачем, є тільки той користувач, якому було надіслане повідомлення;

Одним із надійних способів перевірки цілісності отриманої інформації, є обчисленням MAC коду. При такому варіанті, MAC виступає автентифікатором, що був обчислений певним способом, і представляє собою певний блок даних, за допомогою якого можна перевірити отримане повідомлення. Іноді альтернативою, перевірки отриманої інформації на наявність змін, може бути симетричне шифрування всього повідомлення, але при такому виборі, повідомлення повинно мати достатню надмірність, що і дає можливість перевірити цілісність повідомлення. Тому досить часто на практиці для даного завдання, можна побачити реалізацію кодів автентифікації повідомлення. В такому випадку використовують блок даних, розмір якого фіксований, він і

виступає основною перевіркою, отриманої інформації. В MAC кодах, цей блок створюється за допомогою секретного ключа. Цей ключ мають одержувач і відправник. MAC обчислюється, тоді коли, відомо, що повідомлення є коректним і не було змінене. В кінці отриманий MAC приєднується до повідомлення, та відправляється одержувачу. При отримані, одержувач обчислює MAC, за допомогою ключа, та перевіряє отримане значення з надісланим, якщо значення рівні, то можна сказати, що при пересиланні, цілісність повідомлення не була порушена.

Основні властивості MAC

- Якщо розмір ключа, який використовується для створення MAC, становить k, то існує 2^k можливих комбінацій для підбору ключа. Якщо довжина MAC становить n, то існую 2^n можливих комбінацій значення MAC.
- Щоб обчислити ключ, зловмисник повинене мати кілька пар повідомлень і звісно MAC. Таким чином складність перебору ключів підвищюється.

Функція для обчислення MAC, має такі властивості:

- Знаючи M та K, зловмиснику має бути складно знайти M', так щоб K(M) = K(M')
- Ймовірність знаходження K(M) = K(M'), повинна становити 2^n, в цьому випадку n, це довжина MAC

Принцип роботи кодів автентифікації повідомлень, можна навести на прикладі.

Ми маємо два користувача – користувач A та користувач B, які передають повідомлення message. Для того, щоб гарантувати цілісність повідомлення, користувачам потрібно мати певний метод перевірки цього повідомлення при отриманні іншим користувачем, для цього вони створюють MAC код або код автентифікації повідомлення. Як ми уже знаємо MAC створюється за

допомогою, секретного ключа, і він доступний в нашому випадку тільки користувачу А та користувачу В.

Користувач А для створення MAC використовує хеш-функцію, потім користувач А об'єднує ключ та повідомлення, таким чином, щоб h (K | M). Користувач А надсилає повідомлення разом з MAC користувачу В використовуючи ненадійний канал. Користувач В, відділяє отримане повідомлення від MAC і потім на основі цього повідомлення у поєднанні з секретним ключем робить новий MAC. Отримавши новий MAC користувач В порівнює його з відправленим. Якщо отримані MAC значення рівні, то це свідчить про те, що повідомлення не було змінене.

При використанні MAC кодів у користувачів відпадає потреба у використанні надійного каналу, для обміну повідомленнями, тому що у зловмисника не має можливість, створити повідомлення, та отримати MAC значення повідомлення користувача А та В, не маючи секретного ключа.

На рисунку 2.26 наведено принцип роботи кодів автентифікації повідомлень.

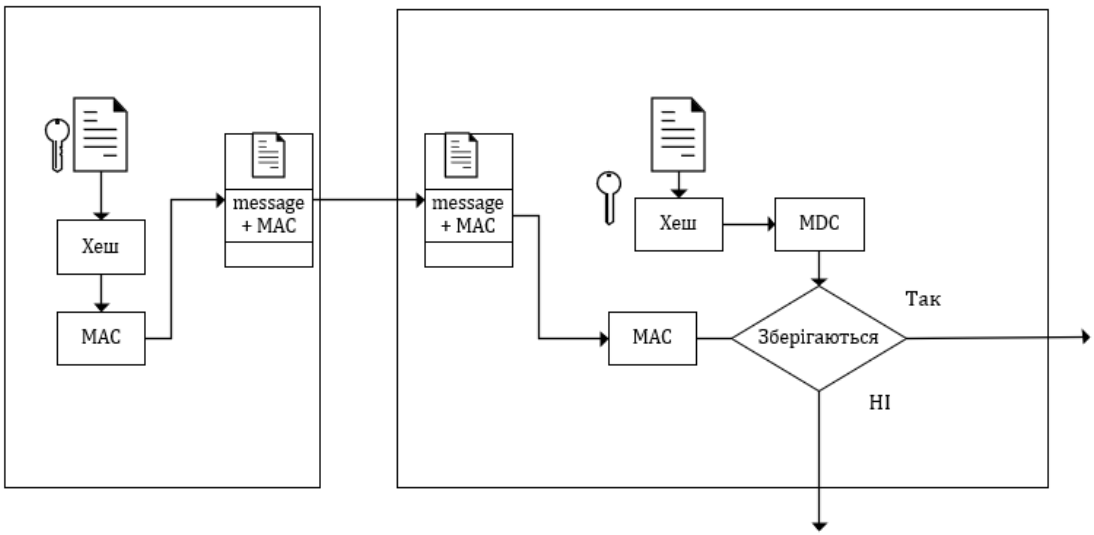


Рис. 2.26 принцип роботи кодів автентифікації повідомлень

А зараз пропоную більш детально розглянути коди автентифікації повідомлення HMAC та CMAC.

HMAC

HMAC код це варіант коду MAC, який будується за допомогою секретного ключа та хеш – функції. У випадку з HMAC, хешування в HMAC є результатом конкантенації створеного секретного ключа та та отриманого дайджесту повідомлення. Тому, як і в випадку з класичноним MAC кодом, заволодівши секретним ключем, зловмисник не має можливості обчислити значення HMAC.

В документі RFC 2104, при розробці HMAC дотримувалися, таких цілей:

- Можливість використовувати з кодом автентифікації існуючі функції хешування без змін, особливо тих, для яких уже існують перевірені часом, відкриті та широко розповсюджені програмні реалізації на різних мовах програмування.
- Наявна легка заміна функції хешування для отримання HMAC значення, більш швидкісними функціями або функціями хешування які мають більший показник захищеносі, тобто тими функціями, в яких враховують всі особливості життєвий циклів хеш-функцій.
- Дотримання швидкості роботи HMAC, в залежності від вибраної хеш функції, час обчислення HMAC повинен бути близьким до результатів вибраної функції хешування.
- Наявна можливість застосування ключа та звісно простота розподілу і звернення безпосередньо до ключів.
- Легкість отримання аналізу стійкості впровадженого коду аутентифікації. Стійкість коду аутентифікації, базується безпосередньо на показниках стійкості вибраної в основу хеш – функції. Це і є головною перевагою HMAC, у порівнянні з представниками, які використовують хеш функції. Тобто вибравши стійку функцію хешування, ми гарантовано отримуємо захищений HMAC.

На рисунку 2.27 наведено принцип формування HMAC

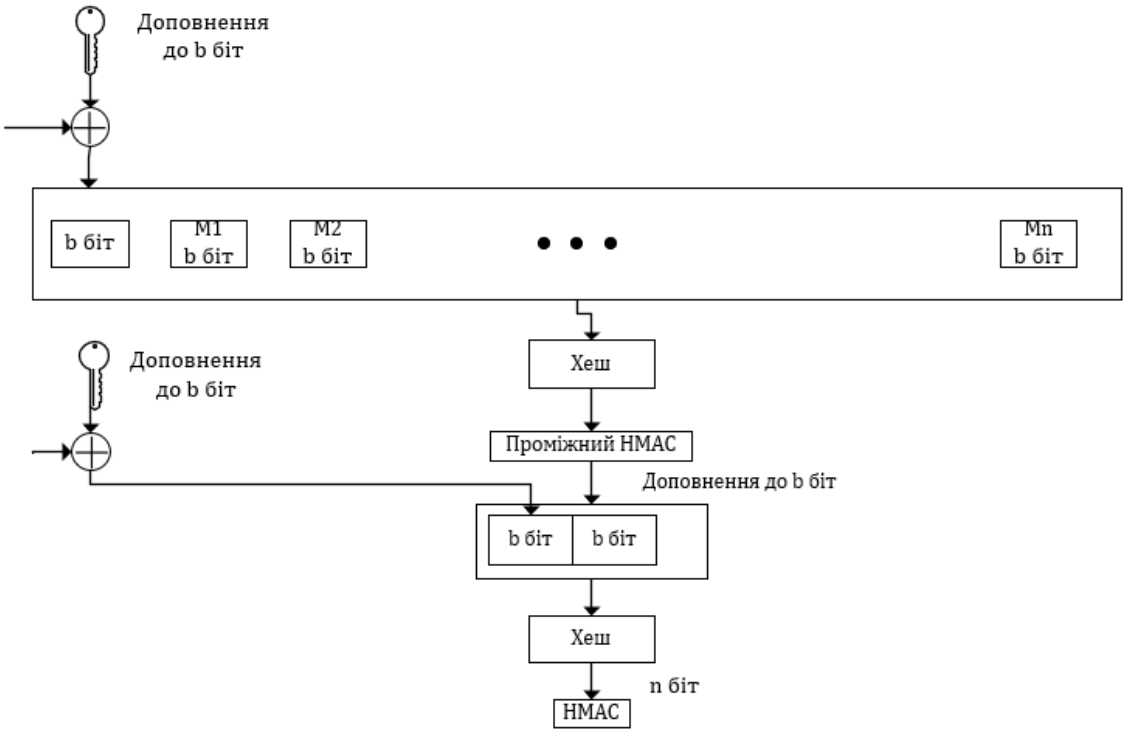


Рис 2.27 Принцип формування HMAC

Необхідні кроки для формування HMAC:

1) При отримані, вхідне повідомлення ділиться на N кількість блоків, розміром в b біт.
2) Ключ обов'язково доповнюється зліва нулями для отримання розміру b-біт ключа. В ідеалі, щоб ключ до доповнення, мав розмір більше розрядності HMAC, тобто більше розміру виходу хеш-функції.
3) Виконується операція XOR, до ключа, який є результатом другого кроку з додатковою константою ipad.
4) Отриманий в результаті блок додається з лівої сторони до повідомлення поділеного на N кількість блоків.
5) Отримане повідомлення в четвертому кроці хешується, в результаті отримується n-розрядний проміжний HMAC.
6) Проміжний HMAC доповнюється нулями з лівої сторони до поки не отримається b-розрядної блоку.
7) Відбувається повторення кроку один та два з додатковою

константою opad.

8) Результат отриманий в сьомому кроці додається з лівої сторони до результату шостого кроку.
9) Результат отриманий в восьмому кроці хешується, за допомогою хеш-функції, яка використовувалась в четвертому кроці, після цього ми отримуємо HMAC.

Стійкість HMAC безпосередньо залежить від розміру ключа, який використовується в процесі обчислення. Оптимальним розміром ключа є 32 байти. Поширеною атакою на HMAC є атака грубої сили, яка спрямована на розкриття секретного ключа. Але в свою чергу HMAC менш схильний до виникнення колізій.

Ще одним варіантом атаки є використання парадоксу про дні народження, для цього потребується $2^{n/2}$ атак взаємодії з системою.

CMAC

CMAC код, це один з різновидів коду MAC. Cipher-based Message Authentication Code, що в перекладі означає шифрований код автентифікації повідомлення. У випадку з CMAC, для отримання дайджесту повідомлення використовується секретний ключ, та блоковий шифр наприклад AES. Суть призначення, не відрізняється від HMAC та MAC, це в свою чергу перевірка цілісності повідомлення.

Принцип роботи CMAC

Вхідне повідомлення початково ділиться на N кількість блоків, блоки діляться таким чином, щоб у кожному було по m розрядів. У випадку, якщо в останньому блоці бракує бітів, тобто довжина блоку меньше за m, то він збільшується, за допомогою додавання одиничного біту, та потім додатково доповнюється нульовими бітами, допоки розмір не буде дорівнювати m. Припустимо, що n це розрядність нашого HMAC. Перший блок поділеного раніше повідомлення позначимо, як Mes1, шифрується за допомогою отриманого симетричного ключа Kcmac, за результатом цієї дії, ми отримуємо блок шифротексту потрібного розміру m

Цей процес виконується до останнього блоку повідомлення. В процесі отримані блоки шифротекста додаються між собою по модулю 2, тобто Mes1 додається до Mes2 і так до блоку Mes_N, після чого останнім етапом є отримання CMAC, з n лівих розрядів. Додатково для отримання CMAC, за допомогою, ще одного ключа. Він отримується, шляхом шифруванням Kcmac блоку який містить нульові біти, отриманий результат додатково множиться на x чи на x^2.

На рисунку 2.28 наведено принцип формування CMAC

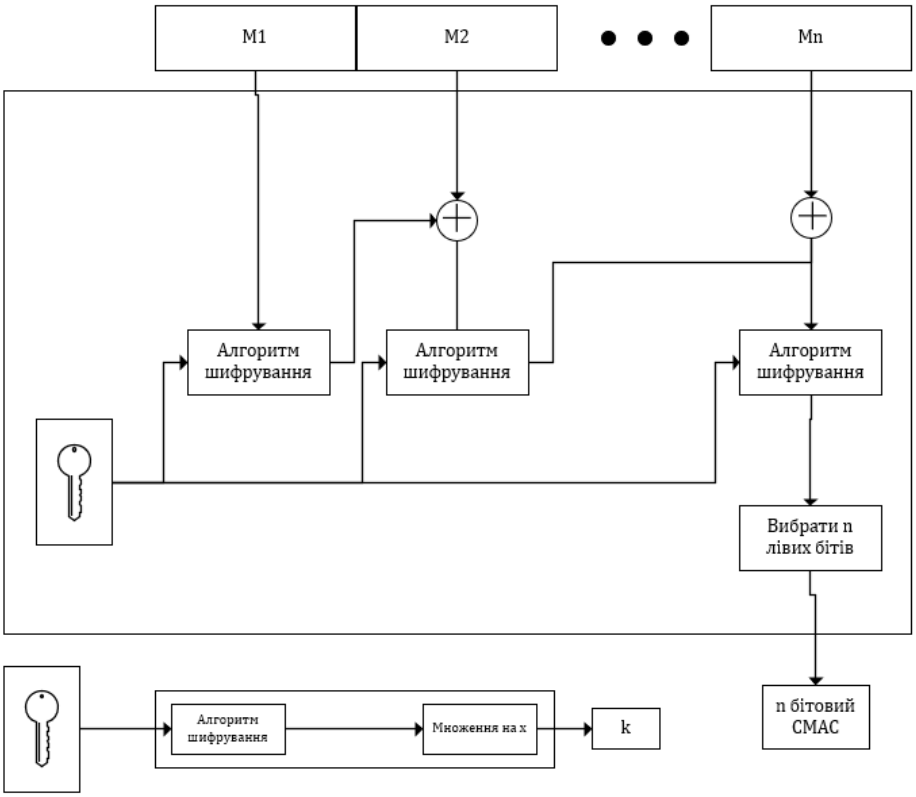


Рис 2.28 схема створення CMAC.

MDC

Message Digest Code, вже з самої назви, ми можемо припустити, що MDC отримується в наслідок роботи хеш функції, а саме MDC є хеш-значення повідомлення Mes, і звісно для отримання цього значення використовувалася хеш-функція, таким чином Mes’ = h(Mes)

Як і з MAC кодами, на вхід подається повідомлення з довільною довжиною, та на виході ми отримуємо дайджест цього повідомлення. Але є

головна відмінність функції хешування для роботи не потрібне значення секретного ключа.

MDC, можна реалізувати декількома способами:

1) Перший підхід

У цьому варіанті відправник(А), з використанням хеш-функцій отримує дайджест повідомлення, після чого отриманий дайджест додатково шифрується, за допомогою симетричного алгоритму, результат додається до повідомлення і відправляється отримувачу(В). При одержані отримувач(В) розшифровує, та отримує з цих даних дайджес, потім отриманий результат звіряє зі значення отриманого дайджесту, і після цього можна сказати чи була порушена цілісність повідомлення чи ні. На рисунку 2.29 наведений перший підхід для MDC

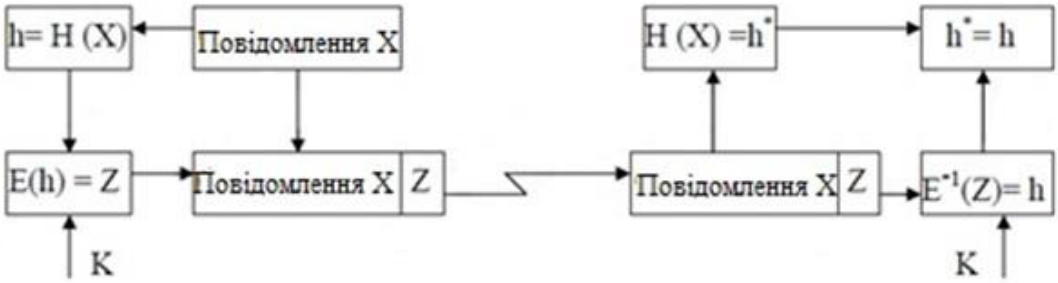


Рис. 2.29 Перший підхід для MDC коду

2) Другий підхід

В другому варіанті отримання MDC, користувачу, вже не потрібно знати значення секретного ключа, що в свою чергу, забирає необхідність його відправляти. На рисунку 2.30 наведений другий підхід MDC

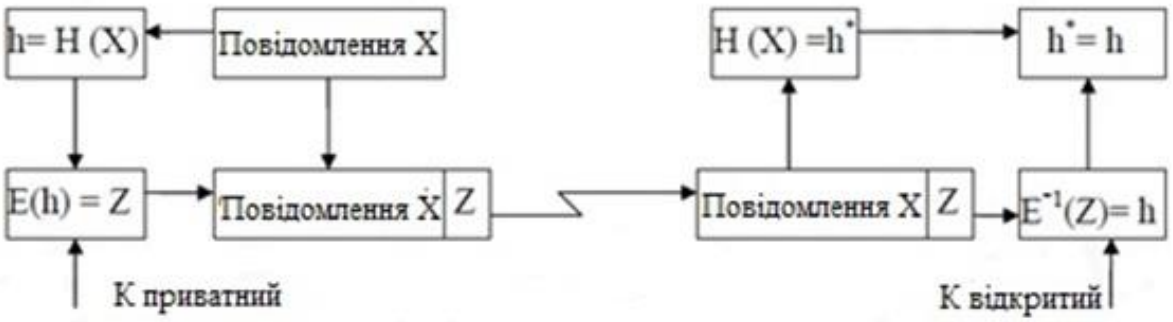


Рис 2.30 Другий варіант MDC коду

3) Третій підхід

У цьому варіанті відправник(A), з використанням хеш-функцій отримує дайджест повідомлення, після чого отриманий дайджест вже не шифрується

На рисунку 2.31 наведений третій підхід для MDC

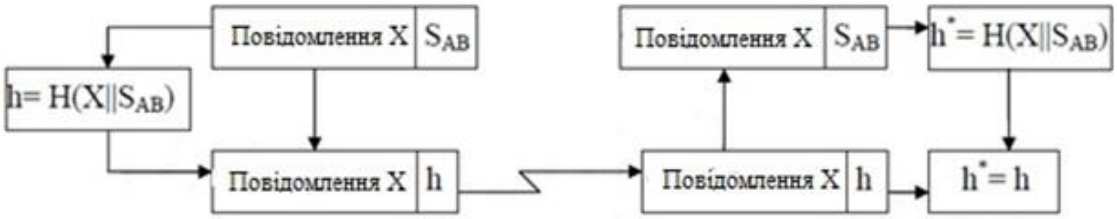


Рис 2.32. Третій підхід для MDC коду

Головною вимогою, при реалізації MDC до функції хешування є те, що не може існувати методу визначення масиву Mes, в якому знаходиться значення дайджесту h(Mes), крім перебору всіх можливих значень Mes.

Наведу приклад, який описує принцип роботи MDC

Маємо два користувачі, відправник(A) та отримувач(B). Відправник посилає повідомлення, і для того щоб впевнитися в цілісності відправленої інформації, відправник створює дайджест, після цього дайджест відправляється разом з повідомленням отримувачу(B). Отримувач створює новий дайджест повідомлення та перевіряє його з отриманим, якщо дайджести рівні, то цілісність отриманого повідомлення не була порушена. Але при такому підході, потрібно розуміти, що дайджест або MDC, потрібно передавати тільки по захищеному каналі, а саме повідомлення можна передати по не захищеному. В такому випадку ми маємо наприклад користувача(C), він звісно може перехопити саме повідомлення, та навіть його змінити, при такому варіанті отримувач(B) точно дізнається про зміни завдяки перевірки, за допомогою MDC.

У разі, якщо відправник(A) нехтує використанням захищеного каналу і відправляє дайджест та повідомлення по незахищеному каналі, то користувача(C) може перехопити повідомлення, без проблем його змінити, створити дайджест нового повідомлення, та відправити його отримувач(B). При перевірці, дайджести будуть свідчити про цілісність отриманої інформації, що фактично правда, але є один нюанс, отримувач(B) не зможе дізнатися, що

відправником цього повідомлення був користувача(С), замість очікуваного відправник(А).

На рисунку 2.33 наведений приклад відправлення повідомлення та дайджесту по різним каналам. Тобто принцип роботи MDC коду.

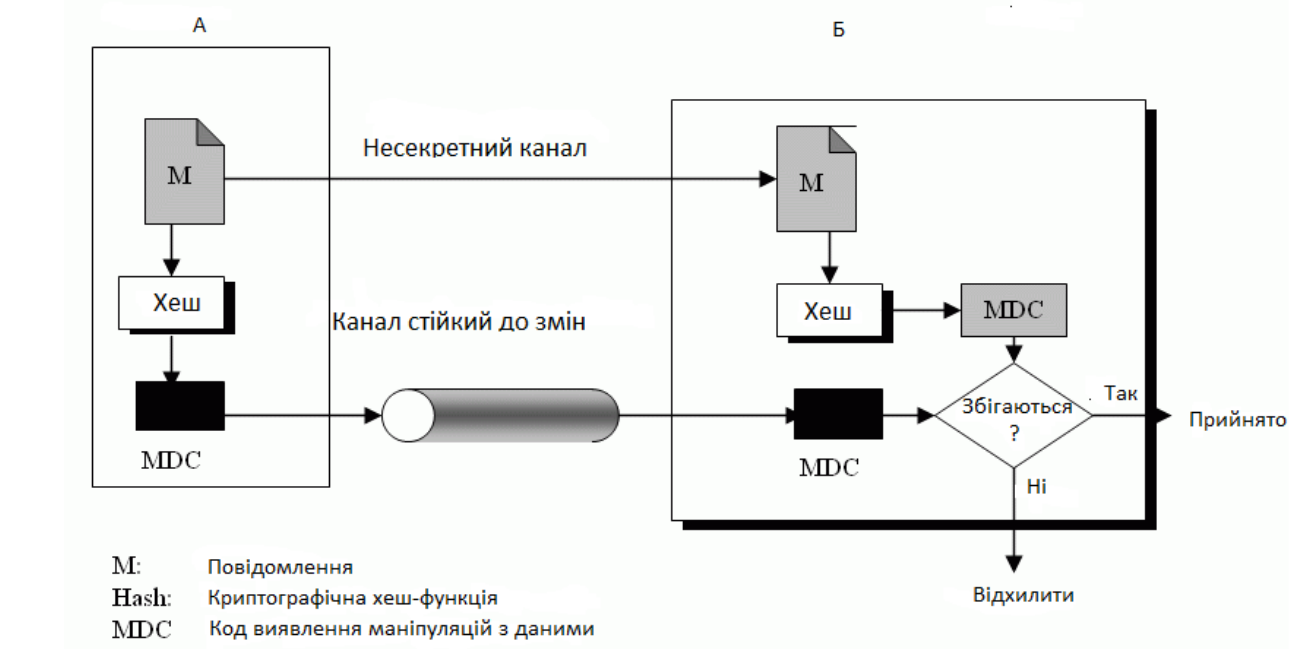


Рис 2.33. принцип роботи MDC коду

Таблиця 2.2

Порівняльні характеристики підходів до вирішення завдання контролю цілісності повідомлень

Параметр порівняння	MAC коди	MDC коди
В основі коду знаходиться	Реалізація відбувається за допомогою блоковий шифр	Реалізація відбувається за допомогою функцій хешування
Використання секретного ключа	Так	Ні
Обчислення коду стороннім користувачем	Не можливо обчислити без значення секретного ключа	Можливе обчислення коду стороннім користувачем

Продовження таблиці 2.2

Параметр порівняння	MAC	MDC
Принцип зберігання коду та його передача	Контрольне значення можна зберігати та передавати разом з вхідним повідомленням	Вхідне повідомлення та його дайджест потрібно передавати та зберігати окремо
Додаткові аспекти для реалізації	Повинен відбуватися розподіл секретних ключів	Необхідність використання захищеного каналу для відправки дайджесту
Області застосування	Захист інформації при передачі від спроб порушення цілісності	Для одноразової відправки інформації, контроль цілісності інформації

З вище наведеної таблиці, можна зрозуміти, що для реалізації модуля гешування паролів, доцільно використовувати саме MAC коди. Тому що у комбінації з використанням динамічної солі, ми значно підвищуємо безпеку паролю та автентифікації в цілому.

А зараз порівняємо HMAC та CMAC коди

Універсальність використання MAC кодів:

- HMAC дає можливість використовувати різні хеш функції для отримання дайджесту повідомлення, це можуть бути MD5, SHA-1 та SHA 2 family, при виборі функції потрібно враховувати такі аспекти, як довжина дайджесту, максимальний розмір вхідної інформації, та звичайно розмір блоку, і звісно бажано подивитися життєвий цикл вибраної хеш-функції, щоб впевнитися в її актуальності застосування.
- CMAC використовує блокові шифри для отримання хеш-коду,

головним представником є AES. AES є представником симетричного алгоритму шифрування, це означає, що для шифрування та дешифрування інформації використовується один ключ.

Незалежність розміру ключа та вибраної функції хешування:

- HMAC не має обмежень по розміру для ключа, та не обмежується розміром блоку, який буде мати вибрана функція хешування.
- CMAC Розмір ключа та блоку безпосередньо залежності від вибору алгоритму шифрування в основу, якщо вибраний алгоритм має певні обмеження по розміру ключа та блоку, то ці обмеження будуть актуальні і для CMAC коду.

Стійкість кодів автентифікації до атак на зіткнення:

- HMAC стійкість до колізій безпосередньо залежить від вибраної в основі функції хешування, якщо функція має хороший показник стійкості до колізій, то і сам HMAC буде стійким.
- CMAC як і у випадку з HMAC стійкість CMAC залежить від вибраного блочного шифру, наприклад шифр AES вважається досить стійким.

Простота реалізації та впровадження вибраного коду автентифікації:

- HMAC Досить просто реалізувати, тому що різні мови програмування мають можливість імпортувати бібліотеки, які надають можливість використовувати хеш функції.
- Реалізація CMAC вимагає вирішення проблем з управлінням ключами, так само проблема може виникнути безпосередньо з режимами блочного шифрування, та з векторами ініціалізації.

Вибір коду автентикації повідомлення, буде залежати від конкретно поставлений вимог. В моєму випадку для реалізацію модуля гешування паролів я використаю HMAC.

2.3 Висновки до другого розділу.

В ході написання другого розділу, мною були розкриті основні питання, які стосуються реалізації модулю гешування паролів:

- Різновиди солі
- Застосування статичної та динамічної солі, для підвищення стійкості до атак райдужними таблицями
- Коди автентифікації повідомлень або MAC коди

Цей розділ, є фундаментом цієї роботи. Без розуміння вище наведених пунктів, не можливо створювати модуль. Основні моменти, які повинні запам’ятатися після прочитання наведеної інформації. Перш за все, при хешуванні паролів, завжди використовувати, статичну та динамічну сіль. Використовувати MAC коди, для безпечного зберігання та отримання дайджесту повідомлення, при виборі алгоритму хешування звертати увагу на життєвий цикл хеш функцій.

РОЗДІЛ 3.
ПРАКТИЧНА РЕАЛІЗАЦІЯ МОДУЛЯ ГЕШУВАННЯ ПАРОЛІВ В РЕАЛЬНОМУ ЧАСІ

В цьому розділі мною буде реалізовано модуль гешування паролів в реальному часі, використовуючи інформацію, яка була наведена в розділі 1 та 2.

3.1 Опис середовища розробки рішення.

Цей модуль гешування паролів повністю заснований на мові програмування Python.

Мова програмування Python, є прекрасним вибором для реалізації модуля гешування паролів в реальному часі. В першу чергу для мене та навіть для всіх програмістів, головною перевагою Python є простий синтаксис, це в свою чергу дуже допомагає, як і в роботі над проектами так і при вивчені мови програмування.

Ще одною великою перевагою python є можливість інтегрування досить великої кількості бібліотек, що в основному дуже спрошує роботу в цілому, та може допомогти у вирішені багатьох проблем.

Основні переваги:

- Простий синтаксис

Вирізняється простотою написання, та безпосередньо читабельністю, що значно зменшує поріг складності вивчення мови, що є великою перевагою для початківців.

- Інтегрована мова

Для виконання коду в Python використовується інтерпретатор, що забезпечує при виконанні, використання однієї інструкції за один раз, цей підхід значно полегшує процес тестування створеного пз.

- Наявна велика кількість бібліотек

Це забезпечує правильність виконання деяких задач та значно полегшує роботу, велика кількість бібліотек робить мову програмування придатною для вирішення багатьох поставлених завдань.

- Можливість використання на різних платформах

Це дає можливість без усіляких проблем інтегрувати та виконувати код на різних операційних системах.

- Велика спільнота

Наявна велика кількість матеріалу для вивчення мови програмування, є багато ресурсів з програмістами, які можуть прийти на допомогу у вирішені проблеми.

Мова програмування Python широко використовується у:

- Веб-розробці
- Створені графічного інтерфейсу
- Реалізації штучного інтелекту
- Автоматизації та системах адміністрування
- Науковій галузі

Для написання та запуску кода я використовував Jupyter Notebook.

Jupyter Notebook – веб-інтерфейс, призначений для програмування та навіть для створення та обміну файлами.

Головними перевагами Jupyter Notebook є легкість встановлення та інтегрування в різні операційні системи.

Має можливість виконувати велику кількість кодів, в окремих комірках, та навіть об’єднувати коди в різних комірках для реалізації одного проекту. Зрозумілий і легкий інтерфейс у використанні.

3.2 Опис середовища розробки рішення.

Для реалізації модудя хешування паролів в реальному часі мною використовувалися бібліотеки python:

- Smtplib
- Ssl
- email.message.EmailMessage
- random
- Hashlib
- Hmac
- Os
- String
- Time

Це всі необхідні бібліотеки для реалізіції модуля гешування паролів в реальному часі

Почнемо з бібліотеки Smtplib

Це бібліотека python, використовується безпосередньо для реалізації двофакторної автентифікації за допомогою електронної пошти. За допомогою цієї бібліотеки відбувається відправка листа з використанням протоколу SMTP.

Бібліотека SSL

Ця бібліотека використовується, для створення додаткової безпеки при мережевих з'єднаннях, зазвичай ця бібліотека використовується в парі з smtplib, для створення та гарантування безпечного з'єднання підчас відправки електронного повідомлення.

email.message.EmailMessage

Це клас бібліотеки email, який використовується для створення шаблону або окремих об'єктів електронного пошти, функціонал цього класу дозволяє створювати, досить складні електронні листи, які можуть містити файли, текст, зображення тощо

Бібліотека random

Ця бібліотека використовується для генерації випадкових, або випадкових елементів з вказаної послідовності. Коді ця бібліотека

використовіється для генерації випадкового коду автентифікації розміром в чотири цифри.

Бібліотека Hashlib

Бібліотека hashlib використовується для підключення функцій хешування, MD5, SHA-1, SHA-256. Тобто використання цієї бібліотеки є обов'язковим для отримання потрібного нам HMAC значення.

Бібліотека Hmac

Ця бібліотека використовується для створення кодів автентифікації повідомлення, які використовують секретні ключі. Потрібно зазначити, що дану бібліотеку слід використовувати разом з hashlib, яка надає нам функцію хешування, для створення HMAC.

Бібліотека os

Бібліотека os, потрібна для коректної взаємодії з системою, бібліотека може отримувати інформації про середовище виконання, маніпулювати шляхами файлів, забезпечує роботу з файловою системою.

Бібліотека string

Використовується для роботи з рядками, бібліотека містить в собі багато функцій, та констант, які безпосередньо допоможуть при роботі з рядками, string може використовуватись, для генерації рядків за стандартом ASCII.

З початку користувачу потрібно зареєструватись, для цього йому необхідно ввести логін та пароль. На рисунку 3.1 наведен процес реєстрації

```
Введіть логін:
Yarmarch
Введіть свою електроннну адресу
marchenkoo.yaroslav@gmail.com
```

Рис 3.1 процес введення логіну та паролю

Коли дані введені, наступним етапом буде створення паролю. Даний модуль має 2 варіанти створення паролю:

1) З розкритою інформацією про створення надійного паролю.

```
Для створення надійного паролю:
1) Використовуйте великі літери
2) Використовуйте малі літери
3) Використовуйте спеціальні символи
4) Використовуйте цифри
5) Довжина паролю повинна становити 10, або більше символів
Не використовуйте часто вживані паролі (qwerty, 123456, pass12345...)
Пароль повинен бути унікальним
При створенні паролю, не використовуйте особисту інформацію
Створіть свій пароль:
```

Рис 3.2 Надання інформації для створення паролю

Користувачу залишається тільки ввести пароль. При введені паролю відбувається перевірка паролю на дотримання необхідних вимог

```
Створіть свій пароль:
AQDF213$&^adw
[True, True, True, True, True]
5 із 5
AQDF213$&^adw
```

Рис 3.3 Перевірка дотримання вимог надійності

У разі, якщо пароль не відповідає вимогам, користувачу буде надана можливість повторного вводу пароля

```
dadaw23133
[False, True, False, True, True]
3 із 5
Виконайте необхідні вимоги:
1) Великі літери
2) Малі літери
3) Спеціальні символи
4) Цифри
5) Довжина більше 10 символів
```

Рис 3.4 Повторний ввод паролю

У разі виконання всіх вимог пароль зберігається

```
SDD1233@&##^sadaf!
[True, True, True, True, True]
5 із 5
SDD1233@&##^sadaf!
```

Рис 3.4 Дотримання всіх вимог надійного паролю

```
Створення паролю: великі літери, малі літери, спеціальні символи, цифри, довжина 10, або більше символів
dawqeqw
1 із 5
Пароль не відповідає необхідним вимогам
QWErty12313@@412asdaw
5 із 5
QWErty12313@@412asdaw
```

Рис 3.5 Створення паролю за стандартним підходом

Для того, щоб уникнути казусів з ненадійними паролями, в модулі є можливість згенерувати пароль:

2) Генерація паролю

Для цього користувач користувач може налаштувати генератор, або використовувати його за замовчуванням.

На рисунку 3.6 наведена генерація паролю за замовчуванням

```
Генерація паролю :
V0Lqg&DKoZWmo>D
```

Рис 3.6 Згенерований пароль, без змінення значень генератора

У користувача є можливість внесення змін в генератор паролів

На рисунку 3.7. генерація паролю зі зміною значення генератора

```
Генерація паролю :
Введіть мінімальну кількість символів :(>= 12)
12
Введіть максимальну кількість символів :
16
Введіть мінімальну кількість великих літер :
2
Введіть мінімальну кількість малих літер :
1
Введіть мінімальну кількість цифр :
1
Введіть мінімальну кількість спеціальних символів :
1
=o_G2MKc4dh8N.
Введіть логін:
```

Рис 3.7 модифікація генератора паролів

Реалізація коду автентифікації HMAC.

Простими словами hmac, це механізм обміну даними з вппровадженням секретного ключа та функції хешування.

Для реалізації на вхід очікується:

- Ключ
- Повідомлення, в моєму випадку пароль
- Хеш функція

В моєму випадку додаю, динамічну сіль для підвищення безпеки користувачів.

Процес отримання хеш значення за допомогою Hmac

При отриманні паролю, створюється сіль та ключ розміром 32 байти. Перед тим, як хешувати пароль, його потрібно перевести в байти, та в моєму випадку додати сіль.

```
Q97Z$1vf0<Fc
b'Q97Z$1vf0<Fc'
b'\xf7\x1f\xb6\xde\xc3\xbc\xb3\xc5\xeeOA\x9d+m\xd0/\xeb\xc7\xd4\x81\x1b(\ri\xa7\xdb\x94`@\xea\xa1\x8b'
b'\xf7\x1f\xb6\xde\xc3\xbc\xb3\xc5\xeeOA\x9d+m\xd0/\xeb\xc7\xd4\x81\x1b(\ri\xa7\xdb\x94`@\xea\xa1\x8bQ97Z$1vf0<Fc'
b'p\xe6\xece\t\xfa\x08\xeeh\x8d$A\xd6m\x126\xc1x\xfd\xa0$\xca\xd0\xf0w\xa3\x9a _Y\xab\xa7'
4c7cd26ddebd16dd78da2c94aa85eb1b73d1d99d0b92820b2a7c5c3ff7d7758d
```

Рис 3.8 Хеш-сума Hmac

Наступним етапом є автентифікація, за для підвищення безпеки в модулі є можливість реалізації двухфакторної автентифікації, за домопогою електронного листа.

```
bbebf0aa8ed3819de3e44e5a54778b44cd1dd331149ea6efc7a0b704b3258fbf
Введіть логін:
Yarmar
Введіть пароль:
Tdxeof.3.XWUP
bbebf0aa8ed3819de3e44e5a54778b44cd1dd331149ea6efc7a0b704b3258fbf
```

Рис 3.9 Процес автентифікації

При введені логіну, в базі даних починається пошук, якщо введений логін дорівнює збереженому, hmac значень. Якщо значення hmac рівна, то на почтову скриньку надсилається лист с кодом автентифікації.

```
bbebf0aa8ed3819de3e44e5a54778b44cd1dd331149ea6efc7a0b704b3258fbf
Введіть логін:
Yarmar
Введіть пароль:
Tdxeof.3.XWUP
bbebf0aa8ed3819de3e44e5a54778b44cd1dd331149ea6efc7a0b704b3258fbf
Перевірте свою почту та введіть отриманий код
Введіть отриманий код
```

Рис 3.10 Отримання листа для автентифікації

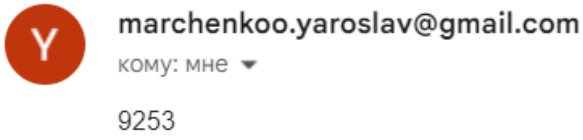


Рис 3.10 Отриманий код автентифікації

І при введені, отриманого коду, автентифікація завершується успіхом.

```
Ласкаво просимо аутентифікація успішна
```

Рис 3.11 Повідомлення про успішну автентифікацю

Після розуміння принципу роботи алгоритму, можна спробувати зібрати модуль гешування, тільки зараз навести можливу візуальну реалізацію.

Перше вікно, вхід в систему. На рисунку 3.12 зображено вікно входу в систему.

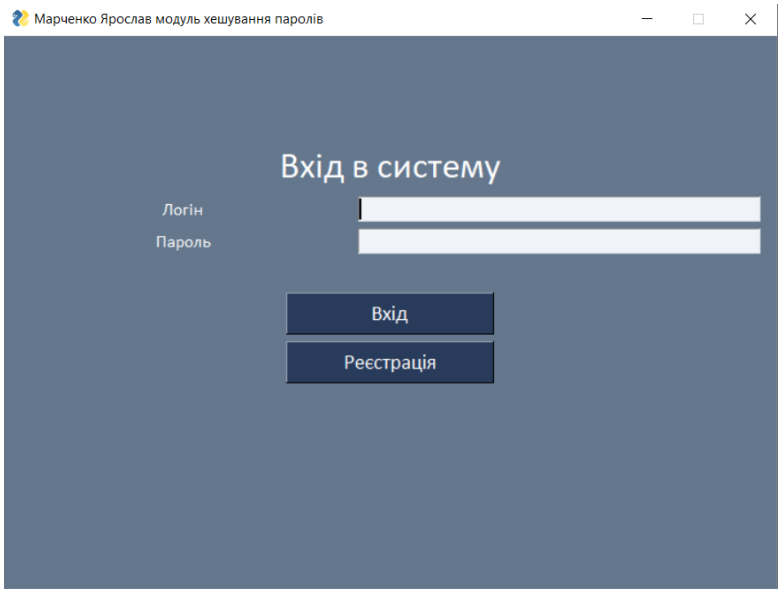


Рис 3.12 Вікно входу в систему

Але перш ніж, скористатися ніж користувач повинен пройти реєстрацію. Потрібно зазначити, що реалізація відбувається, за принципом, який був описаний з верху, але за деякими винятками, щоб продемонструвати гнучкість даного модулю, я прибрав можливість генерації паролю, та введення за стандартним шаблоном паролів.

На рисунку 3.13 наведено вікно реєстрації користувача

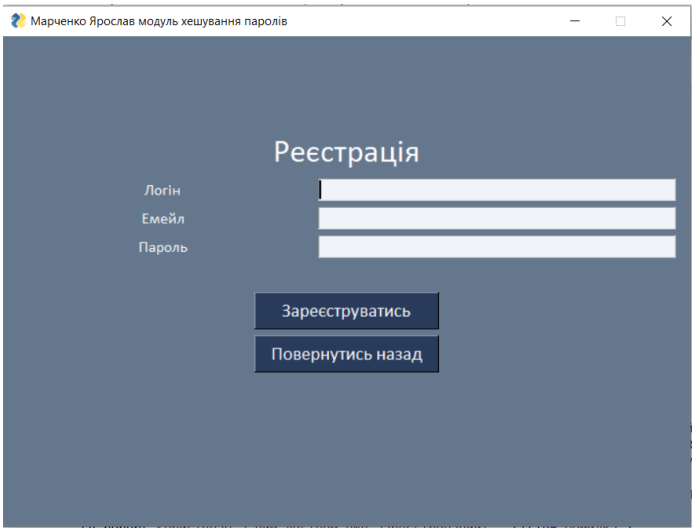


Рис 3.13 Вікно реєстрації

У разі введення не надійного паролю, ми отримаємо допоміжне вікно, з допомогою, для створення надійного паролю.

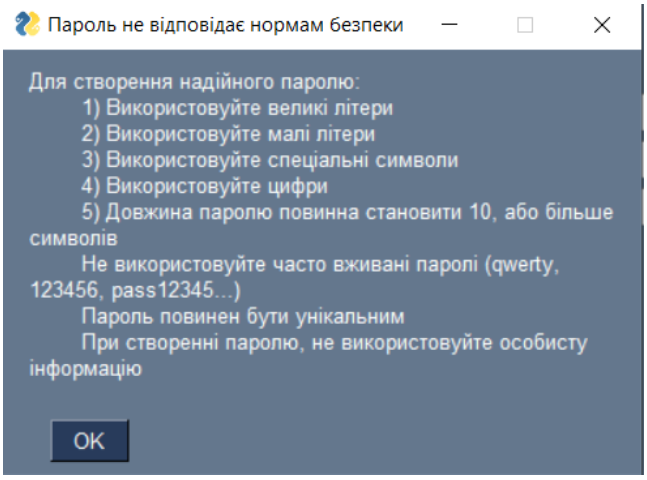


Рис 3.14 Допоміжне вікно для створення паролю

Закривши вікно, і дотримавшись вимог, користувач успішно реєструється Про, що свідчить повідомлення в наступному вспливаючому вікні.

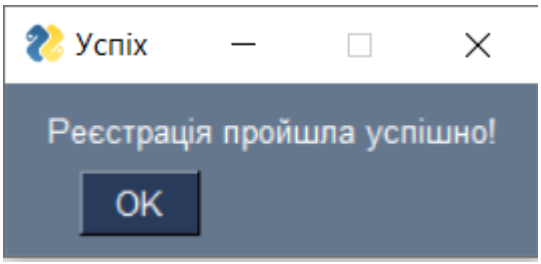


Рис 3.15 Успішна реєстрація користувача

Після успішної реєстрації, логін користувача присвоюється йому, і при повторній спробі реєстрації з таким самим логіном, ми отримаємо помилку.

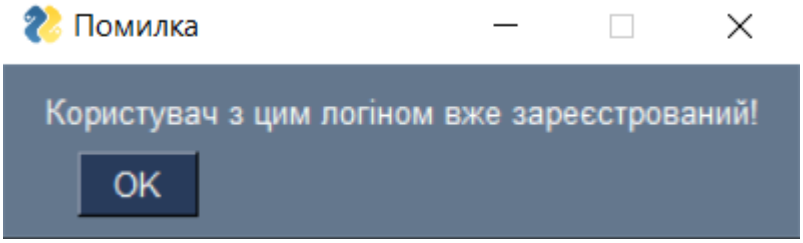


Рис. 3.16 спроба повторної реєстрації з одним логіном

На цьому етапі, користувач вже може спробувати пройти автентифікацію.

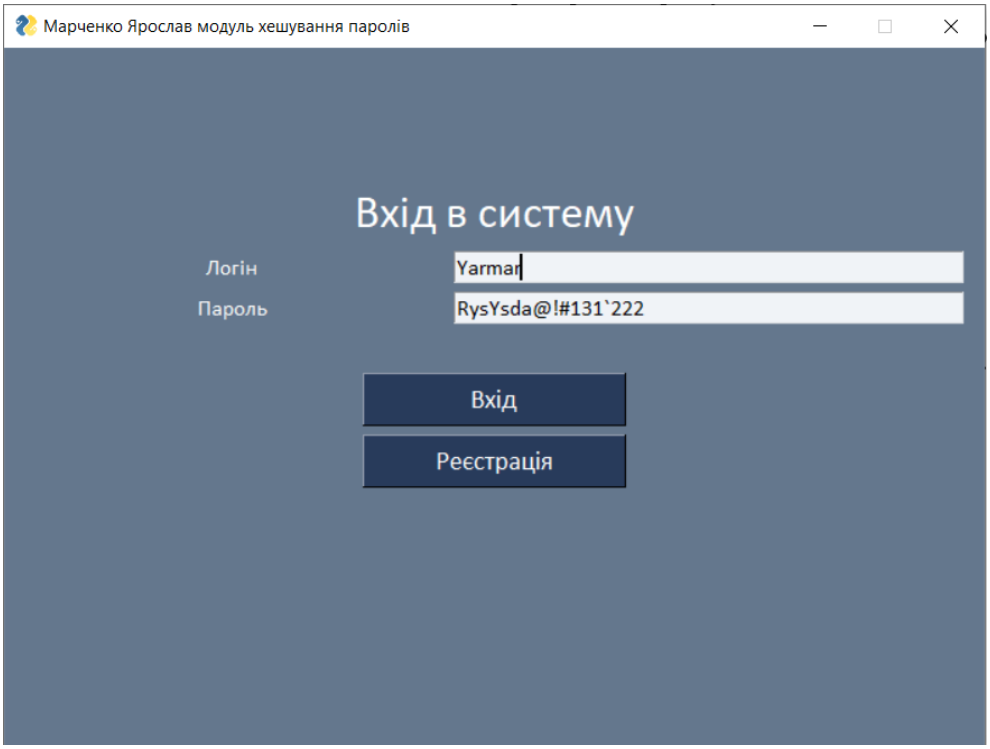


Рис. 3.16. Спроба входу в систему

Якщо, значення hmac рівні, то на електрону скриньку користувача, був надіслан лист з кодом автентифікації.

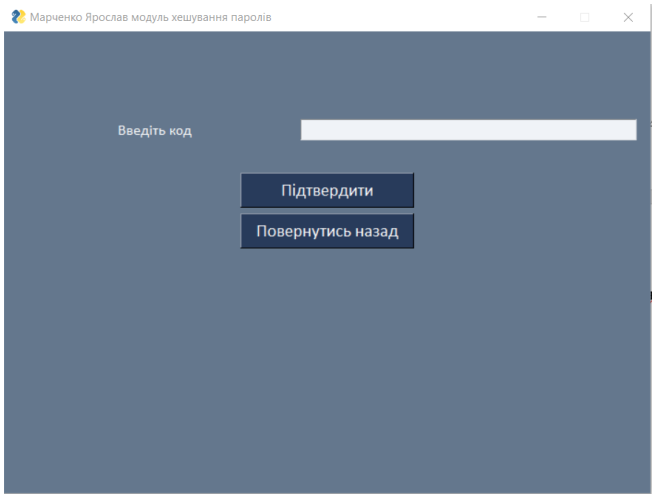


Рис 3.17 Вікно вводу коду

Перевіряємо почтову скриньку, та вводимо у поле отриманий код

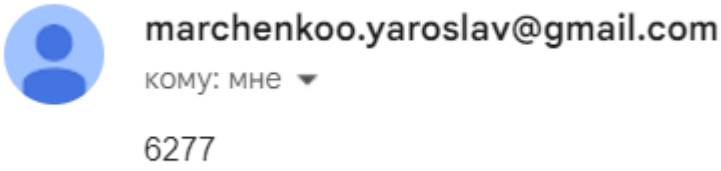


Рис 3.18 Надісланий код

І звісно у разі правильного введення коду, користувач успішно проходить автентифікацію.

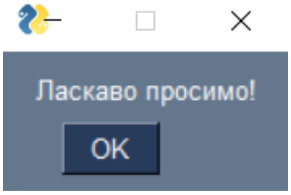


Рис. 3.19. Успішна автентифікація користувача

3.3 Оцінка ефективності та порівняння з існуючими системами.

Щоб оцінити ефективність мого модуля гешування паролів, я порівняю різні реалізації HMAC, а саме: HMAC-MD5, HMAC-SHA1, HMAC-SHA256,HMAC-SHA512.

Та нижче наведу таблицю з отриманими результатами.

Оцінку почну з реалізації HMAC-MD5

- HMAC-MD5

На рисунку 3.20 наведено принцип створення HMAC-MD5, з використанням, динамічної солі та ключа розміром в 32 байти.

```
1. Реєстрація нового користувача
2. Аутентифікація користувача
3. Вихід
Виберіть опцію (1/2/3): 1
Введіть логін:
user1
Введіть свою електроннну адресу
marchenkoo.yaroslav@gmail.com
Щоб створити пароль введіть: створити
Щоб згенерувати пароль введіть: згенерувати
згенерувати
Бажаєте налаштувати генератор паролів : так чи ні
ні
Генерація паролю :
згенерований пароль: gGQ,9xUA#eo>fum*
згенерована сіль: b'\x94\x80\x82\x04\xec\x82%w{\x03\xf3\xfe@\xd1\t\xf8\x1b{\x18\xad{]\x0b\xd0\xa4r\xc1\x16\xd8D\xa91'
додавання солі до паролю : b'\x94\x80\x82\x04\xec\x82%w{\x03\xf3\xfe@\xd1\t\xf8\x1b{\x18\xad{]\x0b\xd0\xa4r\xc1\x16\xd8D\xa91gG
Q,9xUA#eo>fum*'
згенерований ключ: b'\xd6\xde\xc9\x0e\xf6\x05\x13\x01\x1e\\\xb0&B\xe3B\xddCI-\xad2.\xd1sI\x99\x8f1\xa7\xf5Q\xb3'
HMACmd5: 22ae19a9bb7e3d73f52b5e0a7319b1fa
```

Рис 3.20. Отримання значення HMAC-MD5

```
1. Реєстрація нового користувача
2. Аутентифікація користувача
3. Вихід
Виберіть опцію (1/2/3): 2
Введіть логін:
user1
Введіть пароль:
gGQ,9xUA#eo>fum*
newHMACmd5: 22ae19a9bb7e3d73f52b5e0a7319b1fa
```

Рис 3.21. Автентифікація з використанням HMAC-MD5

На рисунку 3.22. наведено результат автентифікації при порушенні значення хешу

```
1. Реєстрація нового користувача
2. Аутентифікація користувача
3. Вихід
Виберіть опцію (1/2/3): 2
Введіть логін:
user1
Введіть пароль:
sadadasda
newHMACmd5: 572feadbe7b7ee6b029dec4aa6073382
Неправильно введена інформація
```

Рис 3.22. Автентифікації при порушенні значення хешу

На рисунку 3.23 наведено принцип створення HMAC-SHA1

- HMAC-SHA1

```
1. Реєстрація нового користувача
2. Аутентифікація користувача
3. Вихід
Виберіть опцію (1/2/3): 1
Введіть логін:
user2
створіть свій логін:
user2
Введіть свою електроннну адресу
marchenkoo.yaroslav@gmail.com
Щоб створити пароль введіть: створити
Щоб згенерувати пароль введіть: згенерувати
згенерувати
Бажаєте налаштувати генератор паролів : так чи ні
ні
Генерація паролю :
згенерований пароль: f3M?7c&QOwTxcKBY
згенерована сіль: b'\xb0&l\xdd:\x13\xdbhtW\x98P+\x9b\xb2F\x80L\xdc\xe5f}\xe3Q\xf2\x9d\rZ\x9b2|\xfe'
додавання солі до паролю : b'\xb0&l\xdd:\x13\xdbhtW\x98P+\x9b\xb2F\x80L\xdc\xe5f}\xe3Q\xf2\x9d\rZ\x9b2|\xfef3M?7c&QOwTxcKBY'
згенерований ключ: b'\xb47\x9a\xbe\xee,\xcb\x8b\x8a\x01*\xea\xf9\xcd;Z\xf4\x8cS\xaeS\xd2\x80\xa2S\x9a!\xaf\x90\xbb\xeax'
HMACsha1: 130eb2304504db1ce76d558c85092e36d7334e53
```

Рис 3.23. Отримання значення HMAC-SHA1

```
1. Реєстрація нового користувача
2. Аутентифікація користувача
3. Вихід
Виберіть опцію (1/2/3): 2
Введіть логін:
user2
Введіть пароль:
f3M?7c&QOwTxcKBY
newHMACsha1: 130eb2304504db1ce76d558c85092e36d7334e53
```

Рис 3.24. Автентифікація з використанням HMAC-SHA1

На рисунку 3.25 наведено принцип створення HMAC- SHA256

- SHA256

```
1. Реєстрація нового користувача
2. Аутентифікація користувача
3. Вихід
Виберіть опцію (1/2/3): 1
Введіть логін:
user3
створіть свій логін:
user3
Введіть свою електроннну адресу
marchenkoo.yaroslav@gmail.com
Щоб створити пароль введіть: створити
Щоб згенерувати пароль введіть: згенерувати
згенерувати
Бажаєте налаштувати генератор паролів : так чи ні
ні
Генерація паролю :
згенерований пароль: 9?BB0iM!WbD#A-n
згенерована сіль: b'\xea\x1b\x0c2U}\xd2\xcb1\xe7U\x9f!\xed\xbcbV\xda\xa6\x9e:T[#q\xe6[\xb5\xf5a\xa1\x97'
додавання солі до паролю : b'\xea\x1b\x0c2U}\xd2\xcb1\xe7U\x9f!\xed\xbcbV\xda\xa6\x9e:T[#q\xe6[\xb5\xf5a\xa1\x979?BB0iM!WbD#A-
n'
згенерований ключ: b'X\xd2\xec\x88\xfc\xa8_\xdb\xd3\xb4\xfd\x1b\xc8z\x00n\xc2\x17\xe4\r\xa6\xc1\x1e@q6\x8c\x80\xed\xf8\xa9\xda'
HMACsha256: 36af2ea6fc48d17758c3e02080ba9c735ce3642281d68c565046f9460fdec54e
```

Рис 3.25 Отримання значення HMAC-SHA256

```
1. Реєстрація нового користувача
2. Аутентифікація користувача
3. Вихід
Виберіть опцію (1/2/3): 2
Введіть логін:
user3
Введіть пароль:
9?BB0iM!WbD#A-n
newHMACsha256: 36af2ea6fc48d17758c3e02080ba9c735ce3642281d68c565046f9460fdec54e
```

Рис 3.26. Автентифікація з використанням HMAC-SHA256

На рисунку 3.27 наведено принцип створення HMAC-SHA512

- SHA512

```
1. Реєстрація нового користувача
2. Аутентифікація користувача
3. Вихід
Виберіть опцію (1/2/3): 1
Введіть логін:
user4
створіть свій логін:
user4
Введіть свою електроннну адресу
asdaasada
Щоб створити пароль введіть: створити
Щоб згенерувати пароль введіть: згенерувати
згенерувати
Бажаєте налаштувати генератор паролів : так чи ні
ні
Генерація паролю :
згенерований пароль: &4eXbWr),b(zedtM
згенерована сіль: b'\xbd\x8e\x95K\xa7\xbc\x0e)\xc8\xecxS+\xc0-\x91\xe3\x16\xbe\xad\xd2\xf1\x92\xf3\x9f\x1a\x07\x13V\x9b\xde\x9
e'
додавання солі до паролю : b'\xbd\x8e\x95K\xa7\xbc\x0e)\xc8\xecxS+\xc0-\x91\xe3\x16\xbe\xad\xd2\xf1\x92\xf3\x9f\x1a\x07\x13V\x9
b\xde\x9e&4eXbWr),b(zedtM'
згенерований ключ: b'L\xa7\xbbe\x95\xa6\xb3P=\xab\xee\xe3&\xce\xc9&MZZ\x8bq\xaf\xb7\x9e;\xbb\x82\x0b\n\xb5L$'
HMACsha512: e44e992c581f6c5a2ef111c885d0833287bf95d188ef7b5ff80727afeccb8fdf3e899ba8a69445ecd72432c75123bd080898eebf1b8ce6b2bd1
52f0a7c8c9d24
```

Рис 3.27 Отримання HMAC-SHA512

```
1. Реєстрація нового користувача
2. Аутентифікація користувача
3. Вихід
Виберіть опцію (1/2/3): 2
Введіть логін:
user4
Введіть пароль:
&4eXbWr),b(zedtM
newHMACsha512: e44e992c581f6c5a2ef111c885d0833287bf95d188ef7b5ff80727afeccb8fdf3e899ba8a69445ecd72432c75123bd080898eebf1b8ce6b2
bd152f0a7c8c9d24
```

Рис 3.28. Автентифікація з використанням HMAC-SHA256

А зараз на практиці подивимось, як реагує HMAC, на змінений пароль

Для цього, ми створимо користувача, та введемо пароль YARmar123!@#$%qwe. Отримаємо Hmac. При процесі автентифікації користувача, в паролі замінюлітеру A з англійського алфавіту на літеру з українського алфавіту, та подивимось, як на це буде реагувати система автентифікації.

Почнемо зі створення користувача user1 з паролем YARmar123!@#$%qwe.

```
1. Реєстрація нового користувача
2. Аутентифікація користувача
3. Вихід
Виберіть опцію (1/2/3): 1
Введіть логін:
user1
створіть свій логін:
user1
Введіть свою електроннну адресу
marchenkoo.yaroslav@gmail.com
Щоб створити пароль введіть: створити
Щоб згенерувати пароль введіть: згенерувати
створити
Бажаєте створити пароль з додатковою теоретичною інформацією: так чи ні?
ні
Створення паролю: великі літери, малі літери, спеціальні символи, цифри, довжина 10, або більше символів
YARmar123!@#$%qwe
5 із 5
згенерований пароль: YARmar123!@#$%qwe
згенерована сіль: b'W!+\xf9\xd5o\x8ez6\x95\x7f38"/\x7f/w\x8c*\xfc\x08\xd4\x96\xcaI\x14\xac\x9a\xb7Y\x13'
додавання солі до паролю : b'W!+\xf9\xd5o\x8ez6\x95\x7f38"/\x7f/w\x8c*\xfc\x08\xd4\x96\xcaI\x14\xac\x9a\xb7Y\x13YARmar123!@#$%q
we'
згенерований ключ: b'\x820,\xd7\xe9n\xd5\xdb\x97\xcc\x9d)\x81\x1b\x1b\x90\x1cg\xb2}\xad\x10[\x9f\xcb[U\x89\xd8\xea\x04A'
HMACmd5: f7a3a00f48f39c4eee7ff4701a936182
```

Рис 3.29 Створення облікового запису

Отриманий HMACmd5 : f7a3a00f48f39c4eee7ff4701a936182

Спробуємо пройти автентифікацію, зі зміненим паролем

```
1. Реєстрація нового користувача
2. Аутентифікація користувача
3. Вихід
Виберіть опцію (1/2/3): 2
Введіть логін:
user1
Введіть пароль:
YARmar123!@#$%qwe
newHMACmd5: 50c4ea9cd09b271267985bffaa63e12f
Неправильно введена інформація
```

Рис 3.30 Введення паролю зі змінами

В результаті отримуємо зовсім інший HMAC, що звичайно говорить нам про наявні зміни в паролі.

Зараз проведемо автентифікацію, не змінюючи пароль.

```
Введіть логін:
user1
Введіть пароль:
YARmar123!@#$%qwe
newHMACmd5: 50c4ea9cd09b271267985bffaa63e12f
Неправильно введена інформація
1. Реєстрація нового користувача
2. Аутентифікація користувача
3. Вихід
Виберіть опцію (1/2/3): 2
Введіть логін:
user1
Введіть пароль:
YARmar123!@#$%qwe
newHMACmd5: f7a3a00f48f39c4eee7ff4701a936182
Перевірте свою почту та введіть отриманий код
Введіть отриманий код
9064
Ласкаво просимо аутентифікація успішна
```

Рис 3.31 Використання паролю без змін

В таблиці 3.1 наведено порівняння отриманих результатів

Таблиця 3.1

Отримані релультати реалізацій HMAC

Реалізація HMAC	Довжина блоку	Довжина Дайджесту	Час формування хеш функції	Час автентифікації
MD5	512	128	3.70 мс	4.04 мс
SHA-1	512	160	4.11 мс	4.33 мс

Продовження таблиці 3.1

Реалізація HMAC	Довжина блоку	Довжина Дайджесту	Час формування хеш функції	Час автентифікації
SHA-256	512	256	4.46 мс	4.98 мс
SHA-512	1024	384	4.11 мс	5.92 мс

Отримані результати, були досягнуті, з використанням 1000 викликів.

Від себе можу додати, що цей модуль, ще не є особливо перевірений на практиці. Але результати з урахуванням додавання динамічної солі досить хороші. Щодо реалізації, тут по перше потрібно враховувати, життєвий цикл хеш функцій. По друге правильний вибір розміру солі та секретного ключа. Адже від розміру ключа, залежить і стійкість HMAC.

А зараз виділю основні переваги мого модулю:

Швидкість формування хеш функції:

Всі наведені в таблиці реалізації, генерують формують хеш функції досить швидко, це дуже важливо коли мова заходить про великі об’єми даних

Гнучкість реалізації:

У самому модулі, по перше створити надійний пароль, або його згенерувати, можливість самостійно обирати розмір динамічної солі та ключа, безпосередньо легкість реалізації, та можлива швидка зміна хеш – функції для отримання HMAC.

Двофакторна автентифікація користувача:

В модулі реалізовано двофакторну автентифікацію, за допомогою електронної скринь, що значно підвищює безпеку особистих даних користувача.

Використання динамічної солі:

Звісно, що це додатковий параметр, який безумовно, підвищює безпеку користувачів в цілому, це значення є унікальним для кожного користувача. додатково генерація унікального ключа, для кожного користувача.

3.4 Висновок до третього розділу

У цьому розділі мною було розроблено авторський криптографічний модуль гешування паролів в реальному часі, за рахунок реалізації HMAC в поєднанні з динамічною сіллю, що дозволяє застосувати його в системах автентифікації субєкта інформаційної діяльності на основі застосування додатково згенереваного унікального ключа для кожного користувача.

Гнучкий підхід для створення паролю, перевірка паролю на необхідні параменти безпеки. Можлива генерація паролів

При тестувані різних реалізацій HMAC, був отриманий досить хороший час отримання значення HMAC.

Для підвищення безпеки автентифікації в модулі було додатково реалізовано двофакторну автентифікацію, за допомогою електронної скринь, що значно підвищює безпеку особистих даних користувача.

Тобто мій модуль, це хороше рішення для систем автентифікації користувачів. Ну і головним є звісно легкість реалізіції модуля.

РОЗДІЛ 4. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Хімічне забруднення довкілля

Хімічне забруднення довкілля - це вплив різних хімічних речовин на природне середовище, який може мати шкідливий ефект на здоров'я людей, екосистеми та біорізноманіття. Це один із типів забруднення, який може виникати внаслідок промислової діяльності, використання хімічних речовин у сільському господарстві, викидів автотранспорту та інших джерел.

Основні групи хімічних забруднювачів включають в себе:

- Токсичні речовини

Це речовини, які можуть спричинити серйозні шкоди здоров'ю людини та екосистемам при навіть невеликих концентраціях. Деякі приклади включають в себе важкі метали, такі як свинець та ртуть, хлорорганічні сполуки, пестициди та інші хімічні отрути.

- Рецикловані хімікати

Це речовини, які можуть накопичуватися у природних середовищах та надовго залишатися активними. Наприклад, хлоровані вуглеводні, такі як поліхлоровані біфеніли (ПХБ) і діоксини, можуть бути надто стійкими та накопичуватися в живих організмах, що призводить до серйозних наслідків для здоров'я.

- Газові забруднювачі

Викиди газів, такі як оксиди азоту і сірки, вуглеводні та інші шкідливі речовини, можуть проникати в атмосферу та впливати на якість повітря, викликаючи смог, кислотний дощ і інші проблеми.

- Пластик та мікропластик

Пластикові відходи розкладаються дуже повільно в природних умовах і можуть накопичуватися в океанах, ґрунті та водосховищах. Мікропластик, дрібні частинки пластику менше 5 мм, можуть впливати на морських тварин і навіть потрапляти в ланцюг харчування людей.

Основні причини та джерела хімічного забруднення довкілля включають:

- Промислові викиди

Багато виробничих процесів включають в себе використання і викид хімічних речовин у повітря, воду та ґрунт. Це може включати в себе викиди токсичних газів, рідких хімікатів і твердих відходів.

- Використання пестицидів та добрив

Сільське господарство використовує хімічні пестициди та добрива для захисту рослин і збільшення врожаю. Вони можуть проникати в грунт і воду, що призводить до забруднення навколишнього середовища.

- Викиди транспорту

Автомобілі, авіація та інший транспорт викидають в атмосферу шкідливі гази, такі як оксиди азоту і вуглецю. Це може призводити до формування смогу та забруднення повітря.

- Обробка відходів

Неконтрольована або ненадійна обробка відходів може призводити до розливу небезпечних хімічних речовин в природу.

- Аварії та розливи

Хімічні аварії на промислових об'єктах або розливи небезпечних речовин можуть мати серйозний негативний вплив на навколишнє середовище та здоров'я людей.

Основні наслідки хімічного забруднення:

- Забруднення води

Хімічні забруднювачі можуть потрапляти в водні джерела, такі як річки, озера та океани, через промислові скиди, агрополівки та інші джерела. Це може призводити до отруєння води та шкоди водним екосистемам. Це також може мати серйозні наслідки для здоров'я людей, які споживають забруднену воду.

- Забруднення повітря

Може мати серйозні наслідки для здоров'я та навколишнього середовища. Інгаляційний ризик, пов'язаний із викидами шкідливих речовин, може спричиняти проблеми з диханням, алергії та респіраторні захворювання, порушуючи здоров'я людей. Крім того, викиди оксидів сірки та азоту можуть призводити до утворення кислотних дощів, що негативно впливає на водні екосистеми, ґрунти та рослини. Ці процеси мають важливий вплив на екологічний баланс та здоров'я всього біорізноманіття.

- Зниження біорізноманітості

Хімічне забруднення може призводити до вимирання видів та зниження біорізноманітності. Токсичні речовини можуть вбивати рослини і тварини, що може впливати на їхні популяції та екосистеми в цілому.

- Забруднення ґрунту

Хімічні забруднювачі можуть також потрапляти в ґрунт, що може впливати на якість ґрунту для сільського господарства та рослинництва. Вони можуть також проникати в ґрунтові води, що може впливати на якість питної води.

- Загроза для здоров'я людей

Хімічне забруднення довкілля може мати серйозний вплив на здоров'я людей. Викладені токсичними речовинами або дихаючи забруднений повітря, люди можуть стикатися з різними захворюваннями, включаючи захворювання дихальних шляхів, рак та інші хронічні захворювання.

- Зміни в кліматі

Деякі хімічні забруднювачі, такі як парникові гази, можуть впливати на клімат. Викиди вуглекислого газу, метану та інших газів можуть призводити до глобального потепління та змін клімату.

З вище сказаного стає зрозуміло, що наслідки хімічного забруднення можуть включати зниження якості повітря та води, втрату біорізноманіття, отруєння ґрунту та водойм, а також загрозу здоров'ю людей. Для зменшення хімічного забруднення довкілля важливо вживати заходів для обмеження викидів шкідливих речовин, сприяти переходу до більш чистих технологій та

споживання більш екологічно чистих продуктів. Також важливо проводити освіту та підвищувати обізнаність громадян щодо хімічного забруднення та його наслідків для здоров'я та природного середовища.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи мною було розглянуто ключові поняття, які необхідно розуміти, перед реалізацією модуля хешування паролів в реальному часі:

- Створення надійного паролю

Головні вимоги надійного паролю, це дотримання досить простих пунктів: довжина паролю повинна бути більше 10 символів, використання малих та великих літер, спеціальних символів та цифр, створювати унікальний пароль, та не повторювати пароль для різних сайтів та соціальних мереж

- Атаки на паролі

Серед атак на паролі можна виділити: атака грубої сили, фішинг, та кейлогери.

- Зберігання паролів

Перед зберіганням паролі, хешувати за допомогою хеш функцій, при цьому бажано розуміти всі слабкі сторони обраної хеш функції.

- Враховувати детермінованість хеш функцій

Тобто при введені одного і того же значення, ми отримуємо один і тойже хеш, а зважаючи на те, що користувачі можуть створювати однакові паролі, то при викрадені бази даних це може бути проблемою.

Для вирішення проблеми детермінованості використовують додатковой параметр, званий, як сіль. В моєму випадку використання динамічної солі є удосконалення існуючого рішення, а саме кодів автентифікації, динамічна сіль є унікальним параметром для кожного користувача, тобто використання саме динамічної солі вирішує багато проблем такі, як: атака райдужними таблицями, в першу чергу принцип детемінованості вхідного повідомлення, вирішує проблему створення однакових паролів кількома користувачами. В моєму випадку сіль, є додатковим параметром для отримання HMAC, використання солі створює додатковий шар безпеки і не дасть змоги отримати зловмисником HMAC користувача так просто, а зважаючи на те, що кожного користувача

унікальна сіль, та унікальний ключ, та пароль, який відповідає нормам безпеки, то спроба отримання HMAC значення, хоча б одного користувача буде вимагати досить багато часу, та зусиль. А Якщо це все додатково захистити двофакторною автентифікацією, то на виході ми отримуємо досить хороший модуль для забезпечення безпечної автентифікації користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

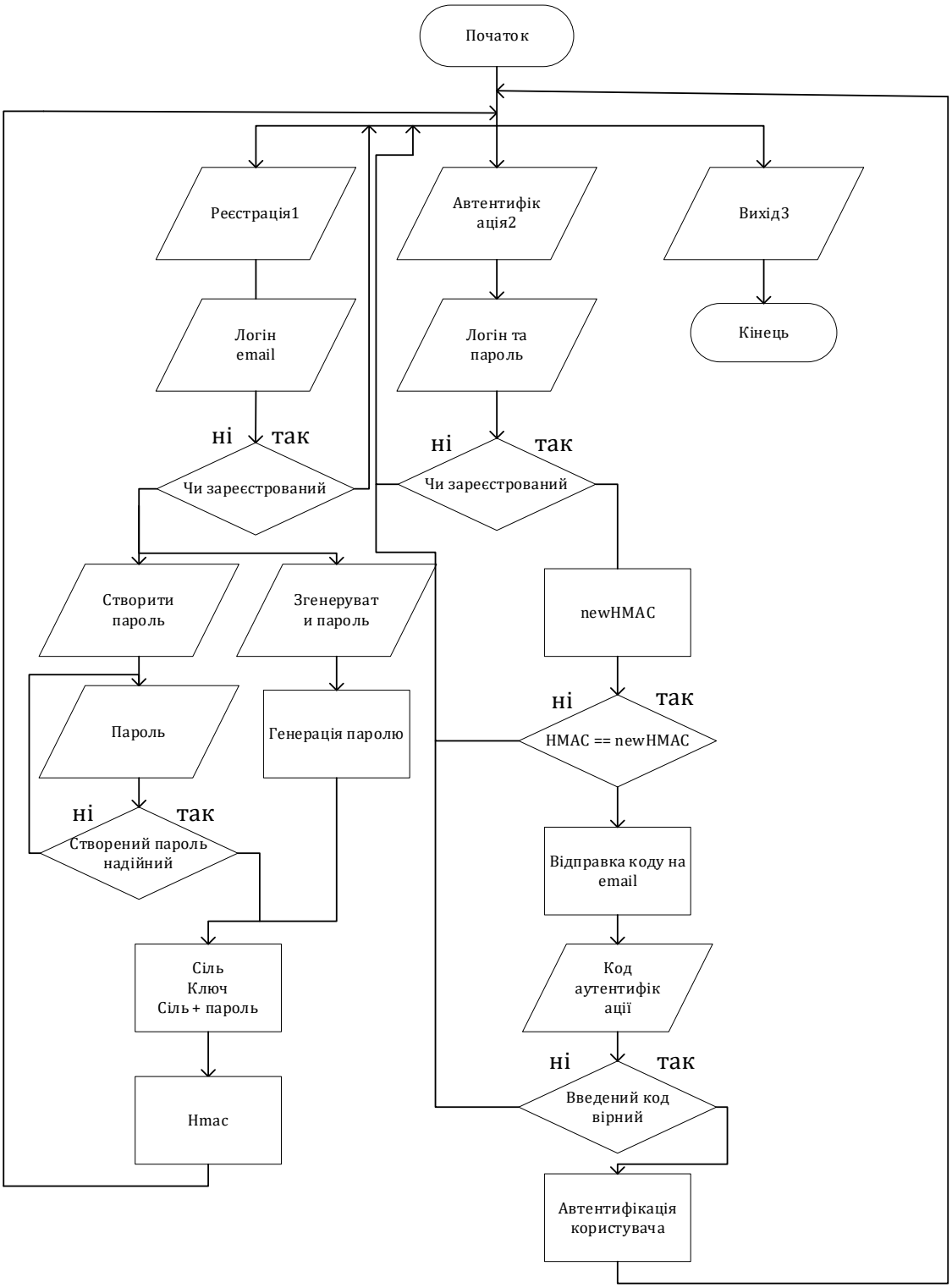
1. Are Your Passwords in the Green? [Електронний ресурс] – Режим 48 доступу : www/ URL: https://www.hivesystems.io/blog/are-your-passwords-inthe-green. – 22.11.2022 р. – Назва з титул. екрана.
2. Як створити надійний пароль — рекомендації спеціалістів ESET. URL: https://eset.ua/ua/news/view/649/nadezhnyy-parol-sposoby-sozdaniya-parolya-otspetsialistov-eset
3. Рейтинг найпопулярніших паролів у 2022 році та небезпека, яку вони несуть. [Електронний ресурс] — URL: https://blog.ipay.ua/uk/rejtyng-najpopulyarnishyh-paroliv-u-2022-roczi-ta-nebezpeka-yaku-vony-nesut/
4. Як поставити пароль на архів RAR, ZIP та 7z [Електронний ресурс] — URL: https://www.habstudia.com/create-password-archive
5. Захист документа паролем [Електронний ресурс] — https://support.microsoft.com/uk-ua/office/%D0%B7%D0%B0%D1%85%D0%B8%D1%81%D1%82-%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0-%D0%BF%D0%B0%D1%80%D0%BE%D0%BB%D0%B5%D0%BC-05084cc3-300d-4c1a-8416-38d3e37d6826
6. VeraCrypt is a free open source disk encryption software for Windows, Mac OSX and Linux. [Електронний ресурс] — URL: https://veracrypt.fr/en/Home.html
7. “10 кращих програм для зберігання паролів” [Електронний ресурс] – Режим доступу: https://root-nation.com/ua/soft-ua/ua-10-krashhix-program-dlyazberigannya-paroliv/
8. Price, Rob (22 лютого 2017). Password managers are an essential way to protect yourself from hackers – here's how they work. Business Insider (англ.).
9. Stallings William, (2008) “Cryptography and Network security” (Fourth edition)Pearson, New Delhi, pg no. 33,35
10. Голубничий Д.Ю., Сєвєрінов О.В., Коломійцев О.В., Місюра О.М., Третяк В.Ф., Власов А.В., Крук Б.М. Аналіз сучасних загроз в інформаційних

системах за складовими загроз: кібербезпеки, інформаційної безпеки та безпеки інформації, 2021.

11. What is a dictionary attack? And how you can easily stop them. CSO Online // URL: https://www.csoonline.com/article/3568794/what-is-a-dictionary-attack-and-how-youcan-easily-stop-them.html (дата звернення: 05.04.2023
12. What is a Brute Force Attack? | Definition, Types & How it Works[Електронний ресурс] — URL: https://www.fortinet.com/resources/cyberglossary/brute-force-attack
13. Chester, John A., "Analysis of Password Cracking Methods & Applications" (2015). Williams Honors College, Honors Research Projects. https://ideaexchange.uakron.edu/honors_research_projects/7
14. Безпека в Інтернеті: що потрібно знати. Профспілка працівників освіти і науки України. URL: https://pon.org.ua/novyny/5427-bezpeka-v-nternetscho-potrbno-znati.html
15. Грайворонський М. В. Безпека інформаційно-комунікаційних систем / М. В. Грайворонський, О. М. Новіков – К.: Видавнича група BHV, 2009. – 608 с.
16. М. Ліндстрем, Я. Стрікер. "Фішингові атаки: підходи та методи боротьби". Видавництво "Логос", 2018 рік
17. Jakobsson, M., & Myers, S. (2016). "Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft". Wiley
18. Захист від фішингу [Електронний ресурс] — URL: https://support.microsoft.com/uk-ua/windows/захист-від-фішингу-0c7ea947-ba98-3bd9-7184-430e1f860a44
19. Security Technology Ltd. Testing and reviews of keyloggers, monitoring products and spyware— http://www.keylogger.org
20. 4 шляхи протидії кейлогерам. [Електронний ресурс]. - Режим доступу: http://www.makeuseof.com/tag/4-ways-protect-keyloggers/
21. What Is a Hash Function in Cryptography? A Beginner’s Guide [Electronic resourse] – Regime of access : www/ URL:

https://www.thesslstore.com/blog/what-is-a-hash-function-in-cryptography-abeginners-guide/. – 23.11.2022 у. – Title from the screen

22. Хеш-функція [Електронний ресурс] — URL: https://uk.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F
23. Алгоритм хешування MD5 [Електронний ресурс] — URL: https://uk.wikipedia.org/wiki/MD5
24. Алгоритм хешування SHA-1[Електронний ресурс] — URL: https://uk.wikipedia.org/wiki/SHA-1
25. Сіль (криптографія) [Електронний ресурс] — URL: https://uk.wikipedia.org/wiki/%D0%A1%D1%96%D0%BB%D1%8C_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D1%96%D1%8F)


Початок
Реєстрація1
Автентифік
ація2
Вихід3
Логін
email
Логін та
пароль
Кінець
ні
так
Чи зареєстрований
ні
так
Чи зареєстрований
Створити
пароль
Згенерувати
пароль
newHMAC
Пароль
Генерація паролю
ні
так
HMAC == newHMAC
ні
так
Створений пароль
надійний
Відправка коду на
email
Сіль
Ключ
Сіль + пароль
Код
аутентифік
ації
Hmac
ні
так
Введений код
вірний
Автентифікація
користувача

```
import smtplib
import ssl
from email.message import EmailMessage
import random
import hashlib
import hmac
import os
import string
import time
from password_generator import PasswordGenerator
users_info = {}

while True:
    print("1. Реєстрація нового користувача")
    print("2. Аутентифікація користувача")
    print("3. Вихід")

    user_ch = input("Виберіть опцію (1/2/3): ")

    if user_ch == "1":
        print('Введіть логін:')
        user_log = str(input())
        if user_log not in users_info:#если логин не зарегестрирован
            print('створіть свій логін:')
            user_log = str(input())
            print('Введіть свою електроннну адресу')
            user_em = str(input())
            print('Щоб створити пароль введіть: створити')
            print('Щоб згенерувати пароль введіть: згенерувати')
```

```
            user_answer1 = str(input())
            pos_answer1 =
['створити','Створити','СТВОРИТИ','згенерувати','Згенерувати','ЗГЕНЕРУВАТИ'
]
            while user_answer1 not in pos_answer1:
               print('Будь ласка, оберіть варіант')
               user_answer1 = str(input())
            if ('створити' in user_answer1) or ('Створити' in user_answer1) or
('СТВОРИТИ' in user_answer1):
               print('Бажаєте створити пароль з додатковою теоретичною
інформацією: так чи ні?')
               user_answer2 = str(input())
               pos_answer2 = ['так', 'Так', 'ТАК', 'ні', 'Ні','НІ']
               while user_answer2 not in pos_answer2:
                  print('Будь ласка, оберіть варіант')
                  user_answer2 = str(input())
               if ('так' in user_answer2) or ('Так' in user_answer2) or ('ТАК' in
user_answer2):#пароль з теоретичною инф
                  print('Для створення надійного паролю:')
                  print('1) Використовуйте великі літери')
                  print('2) Використовуйте малі літери')
                  print('3) Використовуйте спеціальні символи')
                  print('4) Використовуйте цифри')
                  print('5) Довжина паролю повинна становити 10, або більше
символів')
                  print('Не використовуйте часто вживані паролі (qwerty,
123456, pass12345...)')
                  print('Пароль повинен бути унікальним')
```

```
print('При створенні паролю, не використовуйте особисту інформацію')
print('Створіть свій пароль:')
#введение пароля
create_pass = str(input())
upp_case = any([1 if i in string.ascii_uppercase else 0 for i in create_pass])
low_case = any([1 if i in string.ascii_lowercase else 0 for i in create_pass])
special_characters = any([1 if i in string.punctuation else 0 for i in create_pass])
digits = any([1 if i in string.digits else 0 for i in create_pass])
len_pass = len(create_pass)
if len_pass >= 10:
    len_pass = True
else:
    len_pass = False
rules_pass = [upp_case, low_case, special_characters, digits, len_pass]
print(rules_pass)
num_rules = 0
for i in range (len(rules_pass)):
    if rules_pass[i]:
        num_rules += 1
print(num_rules,'із 5')
while num_rules != 5 :
    print('Виконайте необхідні вимоги:')
    print('1) Великі літери')
    print('2) Малі літери')
```

```
                print('3) Спеціальні символи')
                print('4) Цифри')
                print('5) Довжина більше 10 символів')
                create_pass = str(input())
                upp_case = any([1 if i in string.ascii_uppercase else 0 for i in
create_pass])
                low_case = any([1 if i in string.ascii_lowercase else 0 for i in
create_pass])
                special_characters = any([1 if i in string.punctuation else 0 for
i in create_pass])
                digits = any([1 if i in string.digits else 0 for i in create_pass])
                len_pass = len(create_pass)
                if len_pass >= 10:
                    len_pass = True
                else:
                    len_pass = False
                rules_pass = [upp_case, low_case, special_characters, digits,
len_pass]
                print(rules_pass)
                num_rules = 0
                for i in range (len(rules_pass)):
                    if rules_pass[i]:
                        num_rules += 1
                print(num_rules,'із 5')
            else:
                if ('ні' in user_answer2) or ('Hi' in user_answer2) or ('HI' in
user_answer2):
                print('Створення паролю: великі літери, малі літери,
спеціальні символи, цифри, довжина 10, або більше символів')
```

```
create_pass = str(input())
upp_case = any([1 if i in string.ascii_uppercase else 0 for i in
create_pass])
low_case = any([1 if i in string.ascii_lowercase else 0 for i in
create_pass])
special_characters = any([1 if i in string.punctuation else 0 for
i in create_pass])
digits = any([1 if i in string.digits else 0 for i in create_pass])
len_pass = len(create_pass)
if len_pass >= 10:
    len_pass = True
else:
    len_pass = False
rules_pass = [upp_case, low_case, special_characters, digits,
len_pass]
num_rules = 0
for i in range (len(rules_pass)):
    if rules_pass[i]:
        num_rules += 1
print(num_rules,'із 5')
while num_rules != 5 :
    print('Пароль не відповідає необхідним вимогам')
    create_pass = str(input())
    upp_case = any([1 if i in string.ascii_uppercase else 0 for i
in create_pass])
    low_case = any([1 if i in string.ascii_lowercase else 0 for i
in create_pass])
    special_characters = any([1 if i in string.punctuation else 0
for i in create_pass])
```

```
                    digits = any([1 if i in string.digits else 0 for i in create_pass])
                    len_pass = len(create_pass)
                    if len_pass >= 10:
                        len_pass = True
                    else:
                        len_pass = False
                    rules_pass = [upp_case, low_case, special_characters, digits,
len_pass]
                    num_rules = 0
                    for i in range (len(rules_pass)):
                        if rules_pass[i]:
                            num_rules += 1
                    print(num_rules,'із 5')
                else:
                    if ('згенерувати' in user_answer1) or ('Згенерувати' in
user_answer1) or ('ЗГЕНЕРУВАТИ' in user_answer1):
                        print('Бажаєте налаштувати генератор паролів : так чи ні')
                        user_answer3 = str(input())
                        pos_answer3 = ['так', 'Так', 'ТАК', 'ні', 'Ні','НІ']
                        while user_answer3 not in pos_answer3:
                            print('Будь ласка, оберіть варіант')
                            user_answer3 = str(input())
                        if ('так' in user_answer3) or ('Так' in user_answer3) or ('ТАК' in
user_answer3):
                            print('Генерація паролю :')
                            gen_pass = PasswordGenerator()
                            print('Введіть мінімальну кількість символів :(>= 12)')
                            min_len = int(input())
                            while min_len <= 11:
```

```
                print('Введіть мінімальну кількість символів :')
                min_len = int(input())
            gen_pass.minlen = min_len
            print('Введіть максимальну кількість символів :')
            max_len = int(input())
            gen_pass.maxlen = max_len
            print('Введіть мінімальну кількість великих літер :')
            min_uchars = int(input())
            gen_pass.minuchars = min_uchars
            print('Введіть мінімальну кількість малих літер :')
            min_lchars = int(input())
            gen_pass.minlchars = min_lchars
            print('Введіть мінімальну кількість цифр :')
            min_numbers = int(input())
            gen_pass.minnumbers =  min_numbers
            print('Введіть мінімальну кількість спеціальних символів :')
            min_schars = int(input())
            gen_pass.minschars = min_schars
            create_pass = gen_pass.generate()
        else:
            if ('ні' in user_answer3) or ('Hi' in user_answer3) or ('HI' in
user_answer3):
                print('Генерація паролю :')
                gen_pass = PasswordGenerator()
                gen_pass.minlen = 15
                create_pass = gen_pass.generate()

    def add_user(user_log, user_svdata):
        users_info[user_log] = user_svdata
```

```
                user_svdata = {}
                password = create_pass
                print('згенерований пароль:', password)
                salt = os.urandom(32)
                print('згенерована сіль:', salt)
                pass_hmac= salt + password.encode('utf-8')
                print('додавання солі до паролю :', pass_hmac)
                key_hmac = os.urandom(32)
                print('згенерований ключ:', key_hmac)
                user_svdata['hmac_inf'] = salt + key_hmac
                Hmac_pass = hmac.new(key_hmac, pass_hmac,
digestmod=hashlib.md5)
                hashed_pass = Hmac_pass.hexdigest()
                print('HMACmd5:', hashed_pass)
                user_svdata['hmac_digest'] = hashed_pass
                user_svdata['email'] = user_em
                add_user(user_log, user_svdata)
                print(f"Обліковий запис {user_log} створено: {user_svdata}")

            elif user_ch == "2":
                print('Введіть логін:')
                saved_log = str(input())
                if saved_log in users_info:
                    print('Введіть пароль:')
                    ent_pass = str(input())
                    sv_salt = user_svdata['hmac_inf'][:32]
                    salt_pass = sv_salt + ent_pass.encode('utf-8')
                    sv_key = user_svdata['hmac_inf'][32:]
```

```
                ent_pass_hmac = hmac.new(sv_key, salt_pass,
digestmod=hashlib.md5)
                new_hased_pass = ent_pass_hmac.hexdigest()
                print('newHMACmd5:',new_hased_pass)
                if new_hased_pass == user_svdata['hmac_digest']:
                    print('Перевірте свою почту та введіть отриманий код')
                    send_us = 'marchenkoo.yaroslav@gmail.com'
                    sender_pass = 'ytdo zqcl edqb scxx'
                    receiv_us = user_svdata['email']
                    sub = 'authentication code'
                    gen_code = random_number = random.randint(1000, 9999)
                    body = str(gen_code)

                    em = EmailMessage()
                    em['From'] = send_us
                    em['To'] = receiv_us
                    em['Subject'] = sub
                    em.set_content(body)

                    context = ssl.create_default_context()

                    with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context)
as smtp:
                        smtp.login(send_us, sender_pass)
                        smtp.sendmail(send_us, receiv_us, em.as_string())
                        print('Введіть отриманий код')
                        enter_code = str(input())
                        if enter_code == body:
                            print('Ласкаво просимо аутентифікація успішна')
```

```
                else:
                    print('введений код невірний')
            else:
                print('Неправильно введена інформація')

elif user_ch == "3":
    print("Вихід.")
    break

else:
    print("Невірний вибір. Спробуйте знову.")
```