НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»
УДК 004.056

«До захисту допущено»
Завідувач кафедри
_____ Дмитро ЛАНДЕ
“___”_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою «Системи, технології та математичні методи кібербезпеки»

зі спеціальності 125 «Кібербезпека»

на тему: Модель кіберзахисту автентифікації на основі негативної логічної системи

Виконав здобувач ступеня магістра ІІ курсу, групи ФБ-11мн

Харченко Владислав Олександрович
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент кафедри інформаційної безпеки, канд. техн. наук, Гальчинський Л. Ю.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Здобувач ступеня магістра
_____
(підпис)

Київ – 2023 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – другий (магістерський)
Спеціальність – 125 «Кібербезпека»
Освітньо-наукова програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Дмитро ЛАНДЕ
(підпис)

«___»_____2023 р.

ЗАВДАННЯ
на магістерську дисертацію здобувачу ступеня магістра

Харченко Владислав Олександрович
(прізвище, ім'я, по батькові)

1. Тема дисертації Модель кіберзахисту автентифікації на основі негативної логічної системи
науковий керівник дисертації Гальчинський Леонід Юрійович,
канд. техн.наук, доцент, доцент кафедри інформаційної безпеки,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 2023 р.

2. Термін подання здобувачем дисертації ---------

3. Предмет дослідження Безпека автентифікації на основі паролю з використанням негативної логіки.

4. Вихідні дані: технічна документація, реалізація моделі захисту на основі негативної логічної системи.

5. Перелік завдань, які потрібно розробити
1. Проаналізувати предметну область.

2. Проаналізувати існуючі методи захисту процедури автентифікації користувача.
3. Виявити недоліки перелічених реалізацій.
4. Розробити модель додаткового захисту парольної автентифікації з використанням негативної логічної системи.
5. Оцінити складнощі та недоліки реалізації представленої системи.
6. Запропонувати вдосконалення представленої моделі.

6. Орієнтовний перелік ілюстративного матеріалу -- ілюстрацій

7. Орієнтовний перелік публікацій Харченко В., Гальчинський Л. (2023). Модель кіберзахисту автентифікації на основі негативної логічної системи ---

8. Дата видачі завдання ---

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз предметної області. Ознайомлення з процесом та методами парольної автентифікації.		Виконано
2	Аналіз вразливостей паролів		Виконано
3	Виявлення недоліків існуючих систем автентифікації		Виконано
4	Розробка моделі негативної логічної системи		Виконано
5	Реалізація бінарного методу захисту		Виконано
6	Оформлення пояснювальної записки		Виконано
7	Підготовка до захисту		Виконано

Здобувач ступеня магістра _____ Владислав ХАРЧЕНКО
(підпис) (власне ім'я, ПРІЗВИЩЕ)

Науковий керівник Леонід) ГАЛЬЧИНСЬКИЙ
(підпис) (власне ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Робота обсягом 84 сторінок включає 17 ілюстрацій, 11 таблиць, 25 джерел літератури та 1 додаток.

Об’єктом дослідження є безпека системи автентифікації від спроб проникнення в інформаційну систему.

Предметом дослідження є створення алгоритму, який базується на негативній логічній системи для підвищення безпеки від атак на систему автентифікації.

Методи дослідження – є аналіз існуючих систем автентифікації, вивчення літератури та матеріалів за заданою темою, аналіз використання автентифікації на основі негативної логіки.

Метою роботи є підвищення рівня захищеності системи парольної автентифікації.

Результати роботи можуть бути використані для впровадження алгоритму захисту на основі негативної логічної системи в хмарних застосунках, інтернет речей або інших типів інформаційних ресурсів з парольною автентифікацією.

Ключові слова: автентифікація, негативна логічна система, детектор, атака та захист, безпека хмарних технологій, хешування.

ABSTRACT

The 84-page work includes 17 illustrations, 11 tables, 25 literature sources and 1 appendix.

The object of the study is the security of the authentication system against attempts to penetrate the information system.

The subject of the study is the creation of an algorithm based on a negative logic system to increase security against attacks on the authentication system.

Research methods include analysis of existing authentication systems, study of literature and materials on a given topic, analysis of the use of authentication based on negative logic.

The purpose of the work is to increase the level of security of the password authentication system.

The results of the work can be used to implement a protection algorithm based on a negative logic system in cloud applications, the Internet of Things, or other types of information resources with password authentication.

Keywords: authentication, negative logic system, detector, attack and defense, cloud technology security, hashing.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

- NLS (Negative Logic System) - негативна логічна система, яка виявляє аномалії під час авторизації за допомогою анті-паролів(детекторів).
- PLS (Positive Logic System) - позитивна логічна система, яка не використовує додатковий прошарок з негативними значеннями.
- IoT (Internet of Things) - концепції, за якою фізичні пристрої, які ми використовуємо в повсякденному житті, підключені до Інтернету та між собою, щоб обмінюватися даними і забезпечувати взаємодію.
- Captcha (Completely Automated Public Turing test to tell Computers and Humans Apart) - це тест, який розроблено для того, щоб відрізнити людей від комп'ютерних програм або ботів.
- Self-points – умовне позначення множини реальних паролів, які зберігаються у базі даних.
- Detector-points – умовне позначення множини анті-паролів(детекторів), які будуються з використанням Self-points
- V-Detector - це програмне забезпечення для виявлення та аналізу вразливостей безпеки в мережах та на комп'ютерах.
- MD5 (Message-Digest Algorithm 5) - це криптографічна хеш-функція, яка перетворює будь-який вхідний текст або даний невеликого розміру (повідомлення) на фіксований вихідний хеш-код фіксованої довжини 128 біт.
- OTP (One-Time Password) - унікальний пароль, який використовується лише один раз для автентифікації або підтвердження транзакції.

ВСТУП

Зі стрімким розвитком комп'ютерів і комп’ютерних мереж одна за одною створюється, розвиваються і поширюються нові технології та послуги, які приносять велику зручність, змінюють життя людей та роблять його комфортнішим. Серед них такі технології як мобільний інтернет, інтернет речей (IoT), хмарні технології та штучний інтелект. Саме ці речі стали темами, що викликають все більший інтерес у суспільства.

Під час використання цих технологій і сервісів є одна важлива річ без якої у звичайного користувача не буду доступу до виконання певних дій, а саме - автентифікація особи. Автентифікація є необхідною умовою для забезпечення безпечного доступу до застосунку.

Якщо взяти за приклад хмарне середовище, то воно пропонує певні можливості та переваги, серед яких доступність до вашої особистої інформації з будь-якого комп'ютера, який підключений до інтернету, здатність переглядати та оброблювати інформацію з різних пристроїв (портативні комп’ютери, телефони, планшети тощо), гнучка конфігурація і багато іншого.

В той же час ключовим питанням при наданні послуг і спільному використанні ресурсів у відкритому хмарному середовищі постає потреба забезпечення конфіденційності, цілісності та доступності системних ресурсів і даних користувачів у системі, в цьому питанні саме автентифікація є необхідною умовою для досягнення цієї мети.

В даній роботі буде детально розглянуте питання автентифікації, аналіз та виявлення сильних та слабких сторін автентифікації на основі паролю. Проаналізовано як впливає довжина та складність паролю на його міцність до атак на систему автентифікації. В результаті буде запропоноване використання методу в основі якого лежить протилежність до існуючих систем заснованих на позитивній логічній системі, а саме – негативна логічна система (NLS – Negative Logic System)[1]-[5]. Даний підхід не виключає

використання існуючих моделей захисту таких як шифрування або двофакторона авторизація.

Шляхом підвищення складності обчислення для проведення успішної атаки на системи автентифікації дана модель збільшує степінь захищеності застосунку без додаткового впливу на звичайного користувача.

1 ТЕОРЕТИЧНІ ОСНОВИ ІЗ СФЕРИ АВТЕНТИФІКАЦІЇ

1.1 Доступ до інформаційного ресурсу

Для успішного використання вище згаданих систем, таких як IoT або хмарні технології, потрібно пройти декілька етапів без яких робота із переліченими технологіями є неможлива.

По-перше, це ідентифікація особи, вона представляє собою те що перевіряє ким ви є в даній системі, чи існуєте ви як користувач в обраній системі. Зазвичай користувачі звикли до використання email або username для проходження етапу ідентифікації.

Далі для надання доказів, що ви насправді є той, ким ідентифікувалися використовують автентифікація (від слова "authentic" – істинний, справжній), зазвичай це підтвердження паролем від вашого облікового запису. На цьому пункті буде побудована дана робота і приділена увага саме до забезпечення безпеки даних і контролю доступу до ресурсів системи.

І на завершення потрібно перевірити, чи має користувач право на відвідування та виконання дій на тому чи іншому ресурсі, дана дія відбувається в рамках процесу який називається авторизації. Послідовність етапів зображена на рисунку 3.1.

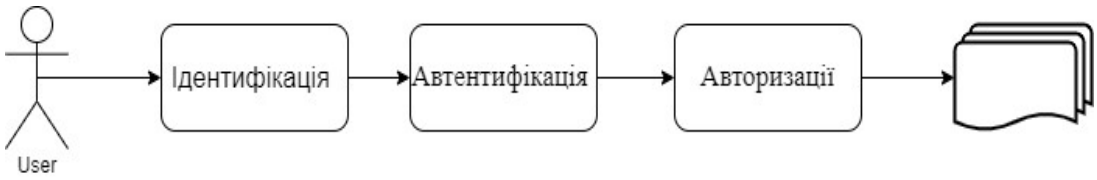


Рисунок 3.1

Всі перелічені вище етапи вкрай важливі для забезпечення конфіденційності даних і контроль доступу до ресурсів, тому це є важливою складовою частиною будь-якої системи, яка працює із обробкою, читанням та збереженням даних.

На цей час у відкритому мережевому середовищі не існує повністю захищеного та на 100% надійного шляху між ідентифікованими та авторизованими сторонами. Це означає, що дані, які передані суб'єктом, і дані, отримані та використані для автентифікації, можуть бути викрадені, обробленні та збережені для подальшого використання зловмисником. Виходячи з цього, постає потреба у тому, щоб забезпечити захист від пасивного та активного мережевого прослуховування, тобто перехоплення, модифікації та копіювання даних.

Передача паролів у відкритому вигляді є незадовільним підходом, бо нічого не буде заважати шахраю викрасти особисті дані користувача. Шифрування паролів також не є панацею, оскільки не запобігає копіюванню та протидії до існуючих методів злому, про які буде йти мова далі. З цього випливає потреба у більш складних протоколах автентифікації.

Варто ще зазначити, що надійна ідентифікація та автентифікація є вкрай важливою не лише через мережеві загрози, але й з інших причин.

По-перше, майже всі об'єкти автентифікації можна побачити, вкрасти або підробити. До таких об’єктів належить ідентифікатор користувача, пароль, ключі, біометричні дані, токени тощо. Зазвичай в ролі унікального ідентифікатору виступає логін, який використовується для ідентифікації користувача в системі. Останнім часом стає все більш і більш популярним об’єкт автентифікації з використанням біометричних даних. Зі слова біометрія зрозуміло, що мова йде про фізичні характеристики користувача, такі як відбиток пальця, голос, обличчя або розпізнавання радужки ока, які використовуються для ідентифікації. Перелічені об'єкти автентифікації дозволяють забезпечити надійну ідентифікацію користувачів в системі та підвищити захист інформації від несанкціонованого доступу.

По-друге, існує протиріччя між надійністю автентифікації, з одного боку, і зручністю для користувачів і системних адміністраторів, з іншого. Наприклад, з міркувань безпеки, користувачі можуть бути зобов'язані повторно вводити свої облікові дані з певною періодичністю, що не тільки не

зручно для звичайного користувача системи, але й збільшує ймовірність того, що хтось вкрав введені дані. Проблема зручності автентифікації для користувачів і системних адміністраторів полягає в тому, що додаткові заходи безпеки, такі як складні паролі, двофакторна автентифікація і біометричні технології, можуть вимагати додаткових зусиль та часу від користувачів. Це може стати проблемою, якщо процес автентифікації стає надто складним або буде займати багато часу, що може викликати відчуття незручності у кінцевих користувачів. В той же час, система має бути спроможною протидіяти до викликів, які створюють зловмисники кожного дня і зберегти у цілісності чутливі дані користувача.

В даній магістерській дисертації буде досліджено саме безпеку автентифікації, оскільки вона захищає конфіденційну інформацію користувачів і забезпечує безпеку систем і даних. На це є декілька причини, чому саме безпека автентифікації така важлива. Одна з них являє собою захист від несанкціонованого доступу або виконувати дії від імені користувача, що може призвести до негативних наслідків. Несанкціонований доступ може призвести до розголошення конфіденційної інформації або злому безпеки інфраструктури всієї системи. Також безпека автентифікації допомагає захистити дані користувачів від втрати або крадіжки, оскільки забезпечує контроль над тим, хто має доступ до цих даних шляхом перевірки користувача на те, що він є тим ким являється. Наприклад, якщо зловмисники отримають доступ до банківської системи без належної автентифікації, вони можуть виконувати фінансові операції від імені користувачів, які не мають на це права. Це може призвести до крадіжки грошей або порушення фінансової стабільності.

Також важливо зазначити, що автентифікація є важливою для захисту конфіденційної інформації. Якщо зловмисники отримають доступ до облікових записів користувачів без належної автентифікації, вони можуть отримати доступ до конфіденційних даних, таких як персональна інформація,

фінансові дані або комерційна інформація, що може призвести до порушення конфіденційності інформації та завдати шкоди користувачам або компанії.

До прикладу інцидентів з порушення цілісності системи можна віднести випадок з однією відомою мережею готелів, який демонструє важливість надійної системи автентифікації. Унаслідок інциденту з атакою на системи бронювання Starwood, яка належить компанії Marriott International, особисті дані 500 мільйонів клієнтів, які перебували в готелях по всьому світу, опинилися в руках хакерів. Сюди входять різні комбінації інформації про гостей, такі як: ім'я, номер телефону, номер паспорта, адреса електронної пошти, адреса проживання, дата народження та стать.

Як результат, акції компаній одразу впали на 6% та мільйони особистої інформації було викладено у мережі Інтернет. Отже, безпека автентифікації є важливою для захисту даних та запобігання несанкціонованому доступу до них.

1.2 Автентифікація

Автентифікація - це процедура перевірки, яка підтверджує справжність осіб або суб'єктів. Щоб забезпечити надійну ідентифікацію, використовуються унікальні ознаки або характеристики. Точність автентифікації залежить від складності підробки або фальсифікації цих ознак.

1.2.1 Методи автентифікації

Існує декілька методів автентифікації, які використовуються в різних мережевих середовищах і застосунках. Нижче наведено приклади з найпоширеніших стандартів автентифікації[6]-[8]:

1. Протокол запит-відповідь. В даному методі користувачі вводять свої облікові дані – логін та пароль – для отримання доступу до ресурсу. Даний тип є одними з найпоширеніших методів автентифікації в інформаційних системах. Цей метод має декілька переваг у забезпеченні кібербезпеки зокрема він є простим у використанні, але в той же час має і мінуси, бо наявність слабкого пароля або використання одного й того ж пароля на декількох різних ресурсах впливають на їх ефективність під час протидії кібератакам. Тому рекомендується додатково використовувати двох факторну, біометричну автентифікацію або іншу систему, яка буде описана в даній магістерській дисертації, на якій побудована автентифікація задля того щоб підвищити рівень безпеки доступу.
2. Протокол авторизації OAuth, який використовується в соціальних мережах та інших онлайн-сервісах. Користувачі можуть надати дозвіл стороннім додаткам доступу до своїх облікових даних на певному сайті без необхідності надавати свій логін та пароль.
3. Разовий пароль (OTP) - це унікальний пароль, який використовується лише один раз для автентифікації або підтвердження транзакції. Основна ідея OTP полягає в тому, що кожен раз, коли користувач потребує доступу до системи або здійснює певну операцію, генерується новий унікальний пароль, який діє лише один раз і має обмежений строк дії.
4. Biometric Authentication. Цей метод став доволі популярний з розвиток мобільних пристроїв і використовує біометричні дані, такі як відбиток пальця, розпізнавання обличчя тощо.

1.2.2 Елементи системи автентифікації

Елементи системи автентифікації – це різні компоненти, які перевіряють справжність користувачів під час доступу до захищених ресурсів. До основних елементів системи автентифікації належать:

- Дані ідентифікації користувача. До даних можна віднести ім'я користувача, паролі, ключі, сертифікати та інші об’єкти підтвердження особи.
- Механізми автентифікації. Певні програмні або апаратні пристрої, які можуть перевіряти логін і паролі, біометричні дані, ключі та сертифікати.
- Системи управління та зберігання облікових записів. До даних компонентів можна віднести елементи, які зберігають інформацію про облікові записи користувачів, наприклад, паролі, ключі та сертифікати. Також ці системи дозволяють адміністраторам управляти обліковими записами користувачів, включаючи створення, редагування та видалення облікових записів, а також призначення рівнів доступу до різних ресурсів в залежності від ролі та обов'язків користувачів.
- Механізм контролю доступу. Під цим мається на увазі ті компоненти, які визначають, що користувачі можуть отримати доступ до певних ресурсів і які права доступу має кожен користувач.
- Системи аудиту та протоколювання. Це компоненти, які реєструють усі спроби автентифікації та доступу для забезпечення безпеки та відстеження зловмисників.
- Системи управління та оновлення. Важливий компонент, який оновлюють паролі, ключі та сертифікати та керують обліковими записами користувачів. До основних функцій таких систем можна віднести формулювання вимог до складності пароля, такі як мінімальна довжина, вимога до використання різних символів, таких як числа, букви верхнього та нижнього регістрів, спеціальні символи тощо. Ці системи є важливим елементом безпеки інформації в будь-якій організації, оскільки вони

дозволяють зменшити ризик злому або витоку даних через слабкі паролі.

- Механізми захисту від атак. Це компоненти, що забезпечують захист від різних типів атак, як-от вгадування пароля та фішинг.

Разом ці елементи забезпечують безпечну автентифікацію користувачів та захищають доступ до захищених ресурсів.

1.2.3 Типи автентифікації

Сувора автентифікація, двостороння автентифікація та тристороння автентифікація є різними методами забезпечення безпеки та перевірки автентичності в обміні даними.

Сувора автентифікація використовується, коли одна сторона бажає підтвердити свою ідентичність перед іншою стороною, не розголошуючи секретний ключ. Це може бути досягнуто за допомогою однобічних хеш-функцій або шляхом володінням таємним ключем, наприклад, за допомогою цифрового підпису.

Двостороння автентифікація вимагає, щоб обидві сторони обміну перевіряли та підтверджували свою автентичність одна перед одною. Це забезпечує взаємну довіру та перевірку сторін, що беруть участь у комунікації.

Тристороння автентифікація включає участь третьої сторони, яка може бути сервером автентифікації або сервером видачі ключів. Це дозволяє додатково підтверджувати автентичність сторін та забезпечувати безпеку комунікації шляхом включення довіреної третьої сторони.

Всі ці типи автентифікації використовуються для забезпечення безпеки ідентифікації та перевірки автентичності в системах обміну даними. Вибір конкретного типу залежить від потреб та вимог конкретного сценарію використання.

1.3 Модель використання захищеного пароля

Швидкий розвиток технологій, зокрема розвиток інтернету та цифрових пристроїв, ставить перед нами нові виклики з питань безпеки інформації, яка включає в себе безпеку паролів.

Одним із таких найбільших викликів стає стрімке збільшення потужності обчислювальної техніки, яка може допомогти кіберзлочинцям зламати пароль. Наразі вже існують програмні засоби, які використовують потужність сучасних процесорів, зокрема графічних, щоб швидко перебрати всі можливі комбінації паролів.

Для того щоб забезпечити безпеку паролів у таких умовах, було розроблено нові методи шифрування та автентифікації, такі як двофакторна автентифікація, біометрична ідентифікація, та в даній роботі представлений метод автентифікація за допомогою негативної логічної системи (Negative Logic System). Двофакторна автентифікація, наприклад, вимагає від користувача введення не тільки пароля, але й додаткового підтвердження, наприклад, коду, який надсилається на мобільний телефон. Проте використання негативної логічної системи жодним чином не спонукає користувача робити додаткові кроки так підтверджувати належність до системи через такі засоби як Captcha, двофакторну або біометричну автентифікацію. Детальний опис та алгоритм вищезгаданого методу буде розкрито більш детально у наступних розділах.

Також було розроблено багато програмних засобів для керування та оновлення паролів, які дозволяють організаціям краще контролювати безпеку паролів. Ці засоби можуть автоматично генерувати паролі, перевіряти їх безпеку та розсилати повідомлення про не коректні паролі.

Загалом, в зв'язку зі стрімким розвитком технологій, безпека паролів стала більш актуальною та складною проблемою, яку потрібно вирішувати в реальному часі. Однак, не зважаючи на те, що не можливо гарантувати абсолютну захищеність, можна забезпечити належний рівень безпеки паролів та захисту інформації завдяки новим технологіям та інструментам.

1.3.1 Модель захищеного пароля

Надійність пароля визначається довжиною пароля і складністю символів, що використовуються в паролі. Крім того, надійність пароля також залежить від унікальності пароля. Пароль, який легко вгадати, є недостатньо надійним.

Для забезпечення високої надійності рекомендується використовувати довгі паролі з комбінацією великих і малих літер, цифр і спеціальних символів. За останніми показниками найнадійнішими є паролі, що складаються з 9 і більше символів.

Також не рекомендується використовувати один і той же пароль для різних облікових записів. Якщо хакер зламає один з ваших паролів, вони зможуть легко отримати доступ до всіх акаунтів, що використовують цей пароль.

Тому, щоб забезпечити надійність ваших паролів, рекомендується використовувати довгі і складні паролі, не використовувати паролі, які легко зрозуміти, і не використовувати один і той же пароль для різних акаунтів.

Однак багато людей недостатньо замислюються про збереження свого пароля або про те, щоб мати хороший пароль. Про даний факт може свідчити статистика за 2022 рік із найпопулярнішими паролям в Україні. А саме наступні 10 паролів є найпопулярнішими за версією Nord Security:

1. qwerty
2. 123456
3. 123456789
4. 1234567890
5. 12345678
6. qwerty123
7. qwertyuiop
8. 1234567
9. Password

10.1qaz2wsx

Як можна побачити зі списку вище, то абсолютна більшість паролів складає найпростіша послідовність, яку користувач може ввести на клавіатурі.

Виходячи з цієї проблеми, яка ще не була вирішена, є те, яку стратегію побудувати для того, щоб паролі були більш успішними в боротьбі з реальними атаками. Однак людина схильна обирати найпростіший шлях і більшість з цих людей не будуть відмовлятись від простих паролів з швидким запам'ятовуванням, а ніж вигадувати складний пароль для того щоб підвищити рівень складності до злому. Рішення, яке буде рекомендовано, це впровадження Negative Logic System. Дана модель орієнтована саме на ускладнення та підвищення рівня обчислювальної потужності під час намагань отримати зловмисником доступ до ресурсу. З цього випливає, що зловмисникові знадобиться багато часу, щоб зламати пароль і як результат – система буде залишатись цілісною. В той же час це не означає, що даний спосіб є панацеєю від усіх кібератак і потрібно притримуватися політик безпеки, такі як хеш-функція, які існують на сьогодні і показали себе в надійному плані.

1.3.2 Хеш-функція

Хеш-функція - це функція, яка перетворює вхідні дані будь-якої довжини в послідовність фіксованої довжини. Результат застосування хеш-функції до вхідних даних називається хеш-кодом або хеш-значенням. На вихід цієї функції отримується послідовність чисел, як найчастіше є шістнадцяткової або послідовності бітів. Саме цей отриманий результат від функції служить унікальним значенням вхідних даних.

Хеш-функції дуже розповсюджена у криптографії, особливо для захисту паролів та інших чутливих даних. В той же час можна зустріти

використання вищезгаданої функції в системах зберігання даних для порівняння та отримання швидкого доступу до потрібних даних. Тобто використання хеш-функції є необхідною умовою для безпечного зберігання паролів.

Хеш-функції можна порівнювати за наступними критеріями:

- Стійкість до колізій. Колізія відбувається коли для різних вхідних даних відбувається перетворення у одне і те саме хеш-значення. Стійкість до даного явища вказує на те, що хеш-функції залишається унікальною для різних вхідних даних.
- Швидкість обчислення. Час, необхідний для обчислення хеш-функції для заданого обсягу вхідних даних. Чим швидше функція, тим ефективніше її можна використовувати в додатках з великими обсягами даних.

- Довжина хеш-значення. Довжина значення отриманого хеш рядка, яка виражена в бітах. Хеш-функції з біль довшим значеннями можуть мати більшу стійкість до колізій, але можуть бути менш ефективними у використанні.

- Розмір даних. Розмір блоку даних, які обробляються одночасно теж впливають на швидкодію усього алгоритму. Великі блоки даних можуть прискорити обчислення, але можуть призвести до вразливостей в системі безпеки.

- Криптографічні властивості. Під цим терміном мається на увазі здатність хеш-функції протистояти різним типам атак. Хеш-функції, які відповідають стандартам криптографічної безпеки, вважаються більш надійними.

Тут варто зазначити більш детально, що колізія - це ситуація, під час якої два різні вхідні дані, оброблені хеш-функцією, дають однаковий вихідний хеш. Іншими словами, коли два різних повідомлення мають однаковий хеш-код. Колізії в хеш-функціях звісно не є бажаною, оскільки вони можуть призвести до вразливостей в системі, яка використовує хеш-функції для зберігання і перевірки цілісності даних. Наприклад, якщо зловмисник може знайти два різних записи з однаковим хеш-кодом, він може підмінити дані і пройти перевірку цілісності. Тому хеш-функції з меншою ймовірністю зіткнення є більш надійними і безпечними у використанні.

Нижче наведена порівняльна таблиця з варіантами різних типів хеш-функції.

Хеш-функція	Стійкість до колізій	Швидкість обчислення	Довжина хеш-значення	Розмір даних	Криптографічні властивості
MD5	Не стійкий	Швидко	128 біт	Невеликий	Застаріла
SHA-1	Не стійкий	Швидко	160 біт	Невеликий	Застаріла
SHA-256	Стійкий	Середня швидкість	256 біт	Середній	Рекомендується
SHA-384	Стійкий	Середня швидкість	384 біт	Великий	Рекомендується
SHA-512	Стійкий	Повільно	512 біт	Великий	Рекомендується
Blake2b	Стійкий	Швидко	256 або 512 біт	Невеликий	Рекомендується

1.3 Методи зламу пароля

Кожен пароль p хешується для створення його хеш-коду h за допомогою функції шифрування f, яка визначена у q. Задача злому пароля полягає у тому, щоб використати метод c таким чином, щоб c(h) = p, оскільки h, на відміну від p, зберігається лише на сервері ресурсів бази даних. Таким

чином, хакер повинен з'ясувати, який інструмент с використовувати, щоб допомогти знайти р.

Оскільки злом пароля є серйозною загрозою безпеці, важливо використовувати складні та унікальні паролі, які буде важко зламати за допомогою цих методів. Однак жоден пароль не може гарантувати абсолютну захищеність системи.

Існує кілька методів, які зловмисники використовують для зламу паролів:

- Атака методом перебору. Цей метод передбачає перебір усіх можливих комбінацій символів до того, поки не буде знайдено пароль, який дозволить увійти у систему. Із недоліків можна визначити, що цей метод займає багато часу, але ефективний для слабких паролів.
- Підміна суб'єкта. Ця атака відбувається, коли зловмисник підміняє справжнього суб'єкта (користувача) під час процесу автентифікації. Зловмисник може намагатись набути доступ до захищених ресурсів або отримати привілеї, видаючи себе за іншу особу.
- Атака словником. Цей метод заздалегідь створює список поширених паролів, і кожен пароль перебирають до того, доки не буде знайдено правильний. Цей метод швидший, ніж атака перебором, але ефективний тільки для слабких паролів.
- Дзеркальна атака. Ця атака використовується для стеження за взаємодією між суб'єктом і сервером і отримання конфіденційної інформації. Зловмисник створює "дзеркальну" копію сервера або середовища, до якого суб'єкт намагається отримати доступ Потім зловмисник перехоплює дані автентифікації, отримує доступ до конфіденційних даних або втручається у взаємодію між суб'єктом і справжнім сервером.

- Атака за допомогою таблиці. Цей метод використовує заздалегідь обчислену таблицю, що містить хеші паролів і відповідні їм значення відкритого тексту. Якщо хеш пароля знайдено в таблиці, зловмисник може легко відновити пароль у відкритому вигляді.
- Повторення (reply). Ця атака відбувається, коли зловмисник перехоплює та повторює повідомлення або запит автентифікації між суб'єктом і сервером. З цим видом атаки зловмисник може скомпрометувати безпеку протоколу, отримати доступ до захищених даних або виконати небажані дії в системі.
- Людина посередині (подвійна гра): Ця атака відбувається, коли зловмисник вставляється між суб'єктом і сервером, стаючи посередником в комунікації. Зловмисник може перехоплювати та змінювати дані, впливаючи на взаємодію між сторонами.

Загалом, злам паролів є серйозною загрозою безпеці і може бути використаний зловмисниками для отримання несанкціонованого доступу до конфіденційної інформації. Важливо використовувати надійні та унікальні паролі, вмикати двофакторну автентифікацію, коли це можливо, і залишатися пильними щодо фішингових атак та інших атак соціальної інженерії.

1.3.4 Аналіз довжини паролю

Перелічені в попередньому розділі атаки на паролі, за допомогою шпигунського програмного забезпечення, дозволяють зловмисникам отримати несанкціонований доступ до системи шляхом вгадування паролів. Незважаючи на те, що атаки можуть мати різний спосіб виконання, вони мають одну спільну мету - отримати доступ до системи.

Якщо визначити загальну мету та підхід, то підбір паролів полягає у спробі різних комбінацій символів до отримання відповідності правильному паролю. Іншими словами, підбір паролів - це процес методом спроб та

помилок. Незалежно від типу атаки, зловмисники будуть намагатися вгадати пароль, перебираючи різні комбінації символів, поки не знайдуть правильний.

Для ефективного та успішного злому пароля атакуюча сторона використовує певні стратегії. Перед використанням методу перебору всіх можливих варіантів паролів, спочатку відбувається перебір усіх очевидних та популярних комбінації. Через ту проблему, що люди не прикладають зусиль для створення стійкого до атак та міцного паролю, цей підхід має право на успіх.

Для початку, варто розглянути вартість атаки методом перебору, яка буде визначати кількість часу для злому. Без знання складу пароля, його довжини та множини символів, що використовуються для створення пароля, перебір усіх можливих комбінацій символів у всіх можливих довжинах є дуже дорогим. Для аналізу впливу довжини пароля на його час варто зауважити, що в паролях можна використовувати 93 символу (у разі англомовної середи), а саме:

- 26 літер маленького регістру
- 26 літер великого регістру
- 10 цифр
- 31 спеціальний символ

Довжина паролю	Можливі комбінації	Час для злому
3	804,357	8 секунд
4	74,805,201	12 хвилин
5	6,956,883,693	19 годин
6	646,990,183,449	2.5 місяці
7	60,170,087,060,757	19 роки
8	5,595,818,096,650,401	1,799 років
9	520,411,082,988,487,296	167,313 років
10	48,398,230,717,929,316,352	155 століть

У табличці вище наведено приблизний час для отримання правильного паролю різної довжини. Варто зауважити, що в даному підрахунку часу використовуються паролі, які містять усі можливі типи символів.

Проте звідси постає фундаментальне питання про стійкість та міцність паролю. Для демонстрації наведено приклад, як впливає міцність паролю на його час злому.

Пароль	Час для злому
Kq@9.	11 годин
12345	0.001 секунда

У таблиці вище наведено приклад двох паролів, які складаються з однакової кількості символів, проте їх міцність відрізняється тим, що вони складається з різної множини символів. Під час аналізу вибірки з 30000 тисяч реальних паролів користувачів сервісу листування Gmail, було виявлено, що 686 паролів складаються з послідовності “12345” і як бачимо по результатам, цей пароль без додаткового захисту можна зламати менше ніж за одну секунду. Через те, що користувач не прагне використовувати міцний пароль, з’являються нові механізму захисту, які мають на меті підвищити рівень захищеності інформаційних. До даних механізмі можна віднести двофакторну автентифікацію та використання негативної логічної системи.

1.3.5 Аналіз міцності паролю

Для аналізу міцності пароля вводиться термін, як FAT-стійкість. Це показник того, наскільки складно вгадати або зламати пароль за допомогою грубої сили. Представлена формула, яка надає оцінку складності паролю, щоб забезпечити належний рівень безпеки. Вона враховує обчислювальну потужність зловмисника, час, який необхідний для злому пароля, та інформацію про користувача, доступну в Інтернеті. Коли користувач створює пароль, він може використовувати загальнодоступні приватні дані, і

представлена система враховує цю інформацію, щоб створити набір правил для оцінки надійності пароля.

Більшість систем класифікують надійність паролів наступним чином: дуже слабкі, слабкі, сильні та дуже сильні. Однак вони не враховують такі важливі фактори, як час, необхідний для злому пароля, обчислювальні потужності зловмисника або інформацію користувача, яка є загально доступною в мережі. Оцінка надійності пароля повинна враховувати міцність пароля по відношенню до атак грубої сили. Для цього використовують певні фактори, які впливають на обчислення FAT-стійкості пароля. Наприклад, під час створення пароля часто використовується дата народження користувача або особистий номер мобільного телефону[9]-[12].

Групуючи наведені вище факти, отримуємо формулу яка розраховує міцність паролю, як показано нижче:

$$F \times s(t) \times \frac{1}{u(t)} \times \mu > T, \text{ де}$$

- F – типу функції зберігання пароля.
- s(t) – обчислювальна потужність, яку використовує зловмисник, щоб зламати пароль.
- u(t) – кількість паралельних процесорів
- μ – множина додаткової інформації про користувача
- T – час, який потрібен зловмиснику, аби зламати пароль. Зазвичай це час визначений політикою зміни паролів.

Якщо отримане значення більше за T, то вважається, що пароль є міцним до атак, якщо ж ця змінна менше за T, то можна зробити висновок, що у зловмисника буде достатньо часу для того, щоб отримати пароль користувача та проникнути у систему.

Загальна тенденція полягає у зростанні обчислювальної потужності s(t) та кількості паралельних процесорів u(t), які можна оброблювати одночасно. Це створює додаткові можливості для зловмисників у розшифровці паролів, оскільки вони мають доступ до більш потужних обчислювальних ресурсів. Для прикладу у 2019 році, Google оголосив про досягнення у випробуванні свого квантового комп'ютера під назвою Sycamore. За результатами випробування, Sycamore виконав обчислення, на які звичайному суперкомп'ютеру потрібно було б приблизно 10 000 років. Це демонструє значний прогрес у вирішенні обчисленні складних завдань, до яких входить злам паролів.

Проте опираючись на статистику по найпопулярнішим паролям користувачів, де часто можна зустріти використання лише цифр або дат народження, номерів телефонів, як висновок маємо, що абсолютно більшість не отримує статус міцного паролю, а отже гостро постає питання впроваджувати додаткові механізми захисту, які будуть запропоновані в наступному розділі.

1.4 Існуючі методи захисту

У даному розділі було розглянуто актуальну тему доступу до інформаційних ресурсів. Було проаналізовано існуючі методи автентифікації, елементи системи автентифікації та модель використання захищеного пароля.

Для запобігання атак відтворення, одним з ефективних механізмів є використання запит-відповідь з елементом, який ніколи не повторюється. Два широко використовуваних методи цього підходу - відмітка часу та монотонно зростаюча функція.

Механізм з відміткою часу передбачає, що кожен запит автентифікації або відповідь містить у собі відмітку часу. Відмітка часу представляє собою значення, яке збільшується з кожною новою транзакцією або запитом. При отриманні нового запиту, система перевіряє відмітку часу і порівнює її з

попередніми значеннями, щоб переконатися, що вона ще не була використана. Це запобігає повторенню запитів або відповідей.

Використання випадкових чисел без повторень в рамках сеансу є ще одним механізмом для запобігання атакам. Принцип полягає у генерації унікальних випадкових чисел для кожного запиту або сеансу, які використовуються як частина даних. Це допомагає уникнути повторення запитів або підроблення, оскільки кожне випадкове число може використовуватись лише один раз.

Обидва ці методи дозволяють системі впевнитися, що отриманий запит або відповідь є унікальним і не було повторень. Це забезпечує надійну автентифікацію і захищає від атак відтворення.

Висновок до першого розділу

У даному розділі було розглянуто актуальну тему доступу до інформаційних ресурсів. Було проаналізовано існуючі методи автентифікації, елементи системи автентифікації та модель використання захищеного пароля.

Досліджено методи зламу паролю та наведено приклади міцності паролів різної довжини та складності. Висновок, який можна зробити на основі проведеного дослідження - використання довгого та складного пароля є необхідністю для забезпечення безпеки віртуальних просторів. Проте не всі користувачі дотримуються правил та політик використання надійного паролю і з цього постає гостре питання про підвищення рівня захищеності системи від атак на систему автентифікації.

2 НЕГАТИВНА ЛОГІЧНА СИСТЕМА

Існуючі механізми атаки та захисту безпеки базуються на механізмах системи позитивної логіки, тобто стан опису атаки та захисту є позитивним по відношенню до логічного опису атаки та захисту. Суть атаки і захисту безпеки полягає у вартості і витратах як для атакуючої, так і для захищаючої сторони, які вони здійснюють під час атаки і захисту. На основі інформаційної невідповідності, ступеня протистояння, статусу переваги і неповноцінності, а також активного і пасивного стану як атакуючої, так і захисної сторони можна покладатися тільки на вартість і витрати тактики кібератаки і захисту. Таким чином, недоліком існуючих механізмів нападу і захисту на основі PLS є обмеження інформаційної невідповідності між атакуючим і захисними діями.

По-перше, на основі PLS інформація зберігається з відношенням один до одного (1:1). Іншими словами, запит від користувача або зловмисника йде напряму до місця, де зберігаються зашифровані паролі. Опираючись на це знання зловмисник може використовувати велику кількість атакуючих груп для здійснення атаки. Під атакуючою групою тут розуміється як власне самі зловмисники, так і будь-який хост, пристрій або комп'ютерну мережеву систему, які можуть бути використані в мережі.
По-друге, існуючі механізми атаки і захисту збільшують складність саме захисту чутливих даних в інформаційній мережі.

Коли мова йде про мережу або систему, що намагається протидіяти зловмиснику, її можна захищати та оборонятися тільки на стороні даної

системи. Щодо децентралізованих або навпаки централізованих методів атак, то захист системи залежить від посилення з боку сторони, що захищається, проти зловмисника, що зумовлює значні витрати на те, щоб залишити систему не ураженою. Для розв'язання цієї проблеми запропоновано інноваційні механізми атаки та захисту безпеки на основі системи негативної логіки.

Механізми атаки та захисту на основі NLS можуть порушити дану рівність інформаційної еквівалентності між атакуючою та стороною, яка захищається. Це забезпечує тим самим перевагу стороні, яка захищається, і знижуючи обчислювальні витрати на кіберзахист. Крім того, запропонований метод автентифікації на основі NLS може бути застосований не тільки в хмарному середовищі, але і в середовищі IoT, і в іншим інформаційних ресурсах що має велике значення для безпечного доступу.

В цій роботі продемонстрована систему негативної логіки у сфері кібербезпеки, а також побудовано алгоритм захисту, заснований на даній системі.

2.1 Імунна система та принципи функціонування Т-клітин

Ідея системи негативної автентифікації базується на алгоритмі негативного відбору[13]-[15]. Ця ідея була заснована з огляду на процесом зрілості Т-клітин у тимусі імунної системи. Тимус - це орган імунної системи, що знаходиться в грудній порожнині людини та грає важливу роль у дозріванні та формуванні клітин саме імунної системи, зокрема, в розвитку Т-клітин. Найголовніше слід відзначити, що Т-клітина видаляється до виконання своїх функцій, якщо вона впізнає будь-яку клітину свого організму.

Т-клітини є важливим компонентом імунної системи. Вони утворюються у кістковому мозку, а дозрівають та формуються в тимусі - органі, розташованому за грудною кліткою. Т-клітини мають різні функції,

наприклад, вони можуть розпізнавати та атакувати інфіковані або змінені вірусами, бактеріями та іншими патогенами клітини в організмі.

Процес дозрівання Т-клітин у тимусі є важливим, оскільки він дозволяє клітинам відрізнити власні клітини від клітин, що належать іншим організмам. У процесі дозрівання, тимоцити - недозрілі клітини Т-клітин - проходять крізь різні стадії вибору клонів, що дозволяє отримати популяцію Т-клітин з різними рецепторами для розпізнавання антигенів.

Одним з ключових етапів дозрівання Т-клітин у тимусі є процес негативного відбору. Під час негативного відбору, Т-клітини, що розпізнають та атакують власні клітини організму, піддаються апоптозу - програмованій клітинній смерті. Цей процес дозволяє уникнути розвитку аутоімунних захворювань, коли імунна система атакує власний організм.

Таким чином, процес дозрівання Т-клітин та негативний відбір є важливими компонентами імунної системи, які дозволяють клітинам Т-лінії розпізнавати та атакувати змінені клітини в організмі, не пошкоджуючи при цьому здорових клітин. Якраз цей принцип було застосовано в Negative Logic System, який використовує алгоритми негативного відбору для моніторингу та захисту комп'ютерних систем. В цьому методі зберігається позитивний простір, що представляє собою множину користувачів, та генерується набір детекторів, які не повинні збігатися з жодним елементом позитивного простору. Якщо будь-яка зміна в позитивному просторі додається або видаляється, то набір детекторів автоматично коригується. Цей підхід дозволяє ефективно виявляти та запобігати несанкціонованому доступу до системи. Нижче в таблиці наведено порівняльний аналіз Т-клітина на Negative Logic System.

Т-клітини	Negative Logic System
Клітини імунної системи	Система безпеки
Розпізнають та атакують змінені клітини	Розпізнає підозрілі запити на автентифікацію в систему
Формуються в тимусі	Генерується в програмі
Проходять процес дозрівання	Проходять процес негативного відбору

Розпізнавання на основі клітинних маркерів	Розпізнавання на основі негативного простору
Захист організму від захворювань	Захист інформаційної системи від шкідливих атак

Таблиця містить порівняння між Т-клітинами та Negative Logic System, двома системами, які мають різні призначення, але використовують схожі концепції для розпізнавання та відхилення шкідливих клітин або небажаних запитів на автентифікацію.

У першому стовпці наведено опис Т-клітин, клітин імунної системи, які детермінують та атакують змінені або інфіковані клітини в організмі. У другому стовпці описано Negative Logic System, систему безпеки, яка використовує негативний механізм для розпізнавання та відхилення підозрілих запитів на проведення процедури автентифікації.

2.2 Ідея негативної логіки

Система негативної логіки є протилежною до позитивної логіки, а відповідне відношення є не один до одного, а один до багатьох (1:N). Що стосується формального опису мови, він може приймати звичайні двійкові, десяткові, шістнадцяткові формати, але найчастіше використовується значення у двійковій формі.

Фундаментальний принцип алгоритму негативного методу полягає в наступному:

1. Визначити множину S (Self-points, або просто Self). Дана множина складається зі списку позитивної, або правильної, інформації про користувача. До цієї множини буть входити логіни та паролі усіх користувачів системи.
2. Створити множину D (Detector-points, або просто Detector), яка не повинна відповідати жодному елементу з множини S.

3. Множина детекторів D має постійно контролювати зміни в множині S. Якщо в множині S відбулися зміни, то наявні детектори мають змінитися.

Як вже було вказано вище, в цьому підході використовуються негативні детектори для виявлення аномальних подій у системі. Математичні методи, що використовуються для генерації детекторів, включають теорію випадкових графів та еволюційні алгоритми. Детектори можна зберігати як точки в просторі, де кожна точка представляє собою комбінацію різних параметрів, що використовуються для визначення стану системи. Для генерації негативних детекторів використовується алгоритм з гарантованим часом виконання, який дає можливість знайти всі недопустимі комбінації параметрів із складністю O(N), де N - кількість допустимих комбінацій параметрів.

Нижче схематично буде представлено робота NLS. Одразу варто зазначити, що в позитивній один вхід і цьому входу відповідає тільки один вихід. Негативна логіка використовує інший підхід, в якого так сам один вхід, проте виходів вже певна множина n.

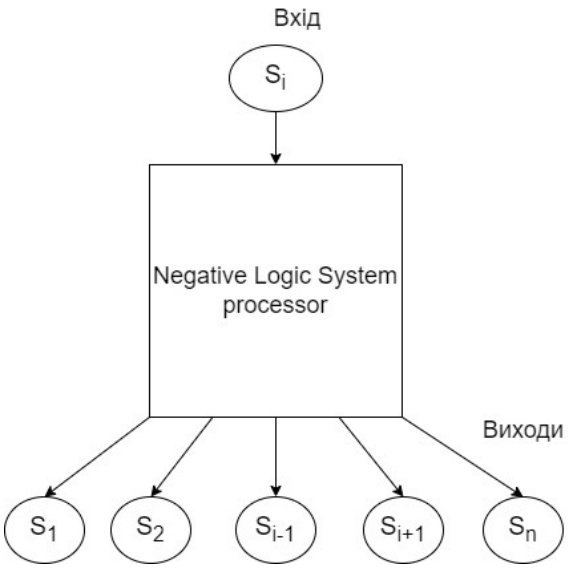


Рисунок 4.1

Відповідно до наведеного вище рис. 4.1, метод NLS поєднує в собі вхід, центр обробки NLS та вихід. Що стосується вхідного елемента, то це значення для введення, яке передається в центр обробки NLS. Вхідним значенням може бути інформація, відформатована в двійковій системі числення, інформація, відформатована в десятковій системі числення тощо.

Що стосується обчислювального центру NLS, то він включає в себе механізми обробки NLS, а саме вибір та перетворення числових систем числення, вибір алгоритму, методу обчислення методу обчислення тощо. Його основна функція полягає у визначенні негативних логічних значень відповідно до вхідних даних і видати результат на вихідну частину. Наприклад, якщо на вхід подано S_i, то від'ємні логічні значення логічні значення знаходяться в наступній множині:

$\{ S_1 \}, \{ S_2 \},\ldots,\{ S_{i-1} \},\{ S_{i+1} \},\{ S_n \}$.

Що стосується вихідного елемента, то одне з негативних логічних значень буде виведено випадковим чином відповідно до методу який було обрано під час обчислення, який працює в центрі обробки Negative Logic System processor, і навіть часу, коли вхідне значення було введено в центр обробки NLS.

2.2.1 Архітектура захисту та нападу на основі NLS

Структура механізмів атаки та захисту безпеки на основі негативної логіки системи на основі негативної логіки наведена нижче на рисунку. Вона складається з модуля атаки, модуля NLS та модуля захисту. Модуль атаки в свою чергу означає механізм, який може поставити під загрозу безпеку системи, використовуючи вразливості або слабкі місця в дизайні або реалізації системи. Модуль захисту означає механізм або техніку, що використовується для захисту системи від потенційних атак і вразливостей.

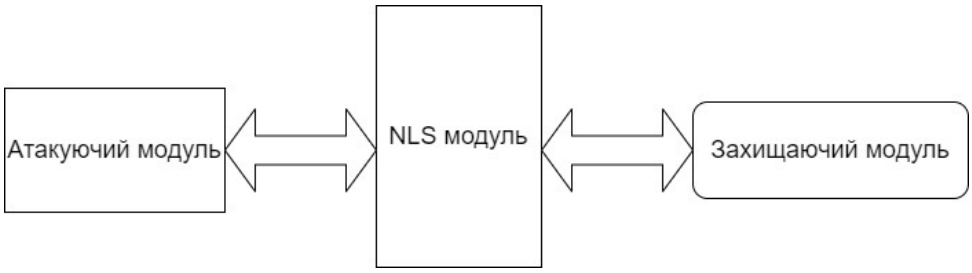


Рисунок 4.2

На рис. 4.2 продемонстровано модуль атаки, модуль NLS та модуль захисту. На даному етапі дуже важливим фактором є те, що враховуючи модель побудови алгоритму та архітектури в цілому, даний підхід не надає атакуючій стороні таку кількість інформації, якої б було достатньо для проведення успішної атаки на систему автентифікації. Тобто в механізмах захисту на основі NLS атакуючий модуль має менше інформації, тому вартість і витрати на проведення атаки набагато вищі, ніж в механізмах захисту на основі PLS, де запит йде на пряму в прошарок зберігання паролів. Модуль NLS в свою чергу зберігає множину із Detector-points і в разі підозрілого запиту із зовні, спрацьовує детектор, та система сповіщає про підозрілу активність і блокує користувача на певний час.

Захисний модуль знаходиться під механізмами безпеки на основі NLS, і його інформація набагато більше через те, що в цьому модулі є доступ до інформації з паролями, так що вартість і витрати на захист набагато нижче, ніж на основі PLS механізмів безпеки.

Аналіз ефективності механізмів атаки та захисту безпеки на основі NLS представлений наступним чином. Згідно з принципом і методом NLS, а також механізмами атаки і захисту на основі NLS, передбачається, що кількість станів системи дорівнює n, які визначаються як:

$\{ S_1 \}, \{ S_2 \},\ldots,\{ S_{i-1} \},\{ S_{i+1} \},\{ S_n \}$, де $S = \{ S_1, S_2, \ldots, S_n \}$.

Опираючись на теорію негативної логіки, всього може бути n – 1 ймовірний варіант значення для буть якого вхідного параметра S_i. Таким чином, щоб отримати значення S_i, потрібно ввести принаймні n - 1 різних

значень. І шляхом перебору та аналізу можна обчислити значення S_i. У порівнянні з PLS, для отримання значення S_i потрібно лише 1 значення, звідси ми можемо побачити, що простір NLS набагато більший, ніж PLS. Що стосується всього простору системи, то простір PLS дорівнює n, тоді як простір NLS дорівнює n(n-1). Коли задано логічне значення, ймовірність успішного судження PLS дорівнює $\frac{1}{n}$, тоді як ймовірність успішного судження NLS дорівнює $\frac{1}{n(n-1)}$.

У механізмах атаки та захисту безпеки, заснованих на NLS, сторона захисту знає число всіх станів, а також обсяг всього простору системи. Саме тому, що інформації для сторони захисту набагато більше, ніж для сторони атаки, а витрати і зусилля, які необхідно докласти, є набагато меншими. Однак, що стосується сторони атаки в механізмах нападу та захисту безпеки на основі NLS, то об'єктивно кажучи, простір безпеки всієї системи спочатку значно розширюється. Він розширюється до другого степеневого відношення для NLS від лінійного відношення для PLS.

По-друге, в реальній атаці і захисті сторона, що атакує, не знає або не може дізнатися про кількість всіх станів, таких як n, так що, навіть якщо зловмисник отримує k видів різних логічних значень, він не може знати, скільки разів йому ще потрібно отримати потрібну інформацію, коли він не знає точну кількість n. Таким чином, складність і потрібна обчислювальні витрати атаки значно зростають. Саме тому інформація для сторони, що атакує, є меншою, ніж для сторони, що захищається, а вартість і витрати, необхідні для сторони, що атакує, є набагато вищими і більшими.

З точки зору сутності атаки та захисту безпеки, то вони полягають у вартості та витратах на проведення атаки або захисту. З наведеного вище аналізу ефективності ми можемо знати, що механізми атаки і захисту на основі NLS можуть істотно збільшити вартість і витрати, необхідні для атаки, і зменшити вартість і витрати, необхідні для захисту. Механізми атаки та

захисту на основі NLS мають важливу практичну цінність і значення в галузі безпеки.

2.2.2 Модель автентифікації на основі NLS

На основі запропонованої системи негативної логіки в даній магістерський дисертації пропонується впровадити автентифікацію особи на основі NLS[16]. Система та метод автентифікації особи на основі NLS можуть бути застосовані не тільки в хмарному середовищі, але й в середовищі Інтернету речей, та інших сервісах, де має велике значення безпечний доступ.

По-перше, порівняємо дану систему та метод автентифікації особи на основі NLS з існуючими системами та методами автентифікації особи.

Існуючі системи та методи автентифікації особистості базуються на системі позитивної логіки. Це означає, що автентифікація облікового запису користувача та інша інформація зберігається в системі автентифікації особи на основі системи позитивної логіки. Іншими словами, під час використання NLS створюється додатковий прошарок, куди спочатку потрапляє запити. В цьому прошарку зберігаються детектори і не зберігаються паролі. Приклад заповнення такого прошарку наведений нижче на рисунку.

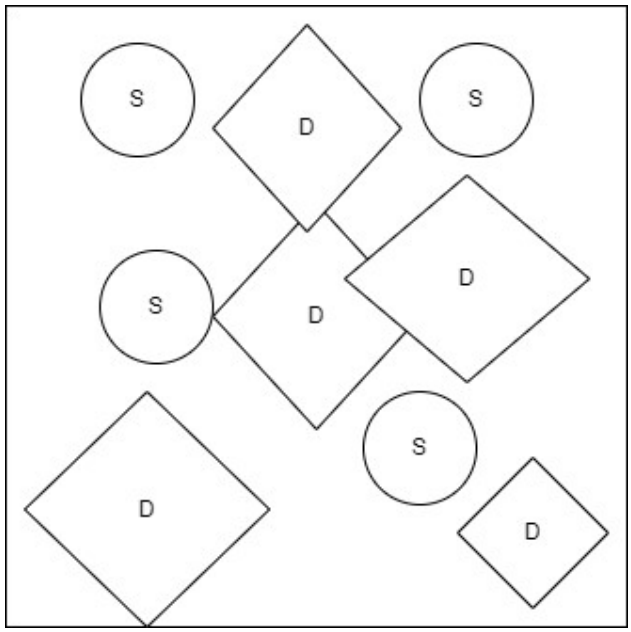
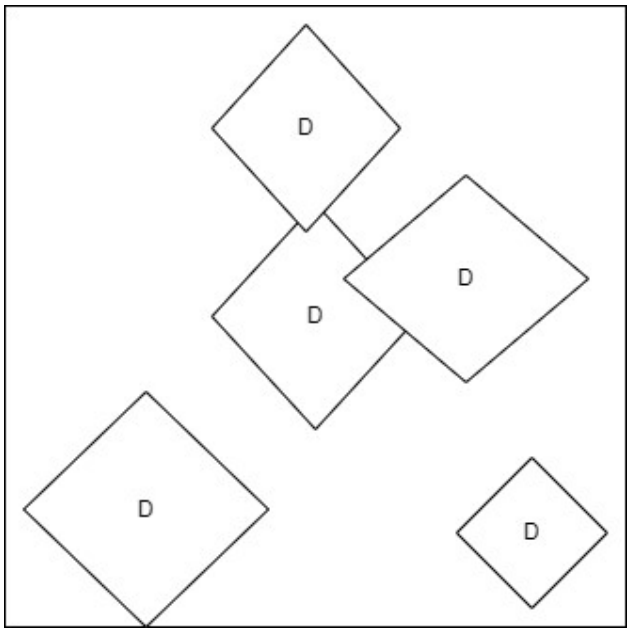


Рис _

Даний рисунок містить круги, ромби та пусту зону. Круги позначають множину S, яка визначає позитивні точки(паролі). Ромби позначають множину D, яка складається з набору детекторів, що не повинні відповідати будь-якому елементу множини S. Дозволяється накладання одного детектору на інший. Ці детектори допомагають виявляти несанкціонований доступ до системи. Пуста зона позначає заплутуючи зону, яка містить випадково створені об'єкти та не відповідає ні множині S, ні множині D. Її функцією є ускладнення завдання для потенційного зловмисника та зменшення ймовірності несанкціонованого доступу до системи.

Якщо станеться виток прошарку, описаного вище, то зловмисник не зможе дізнатися реальні паролі користувачів, бо ця інформація про паролі просто не зберігається у прошарку, і якщо даний прошарок буде скомпрометовано, то для хакера він буде складатися лише з множини



детекторів, як показано на рисунку нижче.

Рис _

Проте якщо станеться виток прошарку, де зберігаються реальні паролі, то наслідки можуть бути набагато гіршими. До них можна віднести викрадення облікового запису користувача, після чого зловмисник може використати отриману інформацію для доступу до системи та ресурсів.

Навіть гірше, зловмисник використовує її як основу для здійснення більшої кібератаки і проникнення, що становить великий ризик для безпеки всієї системи загалом. Крім того, вузьким місцем автентифікації особистості є вибір алгоритму автентифікації, який є цілеспрямованою метою атаки зловмисника.

Якщо алгоритм автентифікації піддається атаці, безпеку всього хмарного середовища вже не можливо гарантувати

2.2.3 Модель автентифікації на основі NLS

Система автентифікації особистості та метод, заснований на системі негативної логіки, може змінити ситуацію інформаційної еквівалентності між атакуючою та захисною сторонами. Інформація як для атакуючої, так і для захисної сторони системи автентифікації особистості не є рівноцінною[17]-[19], [24]. Це може збільшити вартість і витрати на кібератаки на систему, і в той же час зменшити вартість і витрати на кіберзахист системи. Нижче наведена схема автентифікації суб'єкта в хмарному середовищі (рис.4.3).

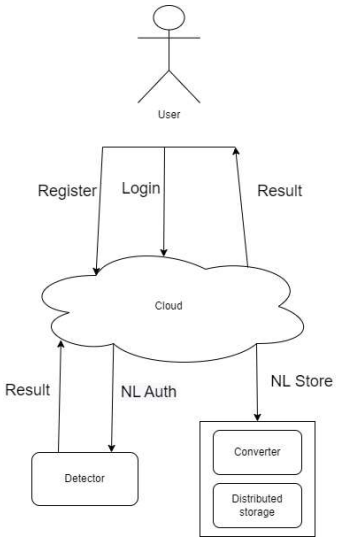


Рисунок 4.3

Життєвий цикл запитів користувача з використанням Negative Logic System складається з наступних етапів:

1. Реєстрація користувача. При реєстрації користувача в системі відбувається оновлення множини детекторів. Додаються нові детектори, або видаляються деякі існуючі, якщо в цьому є потреба.
2. Автентифікація користувача. Під час автентифікації користувач вводить свої ідентифікаційні дані. Система порівнює введені дані із детекторами з множини D (Detector). Якщо є хочаб одне потрапляння на детектор, то запит вважається підозрілим і пропустити його до наступного етапу заборонено, а отже автентифікація вважається не успішною.

3. Видалення користувача. При видаленні користувача з системи його детектор видаляється з множини D (Detector), що призводить до оновлення множини D.

   Кожен з цих етапів тригеррить перетворення множини D і генерацію нових детекторів, які забезпечують відповідність Negative Logic System.

2.3 Варіанти реалізації негативної логічної системи

Існує ряд різних підходів для побудови системи негативної автентифікації, які базуються на представленні позитивного простору паролів. Основною метою цих підходів є генерація негативних детекторів на паролів, які зберігаються в базі даних інформаційного ресурсу. Генерація відбувається з тією метою, щоб охопити якомога більшу частину неправильних запитів на доступ і тим самим підвищити рівень виявлення шкідливих для інформаційного ресурсу запитів на доступ. Кожен з цих підходів використовує своєрідний спосіб представлення та зберігання детекторів у просторі. У даній роботі розглядається три таких варіанти представлення: спосіб з використанням реальних значень, бінарний спосіб та спосіб заснований на сітці. Перші два підходи являють собою недетерміновані з точки зору створення детекторів, тоді як третій підхід є детермінованим підходом до формування набору детекторів. В даному випадку детермінованість означає, що для одного і того самого простору з реальними паролями детектори будуть відрізнятися під час кожної регенерації. Різні підходи створюють детектори різного розміру і надають різний рівень обфускації для запобігання компрометації парольних даних.

Обфускація - це процес приховування та ускладнення розуміння програмного коду або даних, з метою підвищення кіберпезбеки та цілісності системи загалом. Обфускація застосовується для унеможливлення розкриття конфіденційної інформації, запобігання несанкціонованому доступу,

зменшення ризику компрометації та підвищення загального рівня безпеки. У контексті системи негативної автентифікації, яка згадана в попередньому тексті, обфускація застосовується для створення детекторів різних розмірів, що забезпечує додатковий шар ускладнення доступу до парольних даних. Це означає, що шаблони детекторів створюються таким чином, щоб було складніше зрозуміти, які саме паролі або запити на доступ вони виявляють. Це допомагає зменшити ймовірність компрометації таких чутливих та важливих даних, як паролі, та забезпечити більшу безпеку системи автентифікації.

Докладні характеристики кожного з цих підходів розглянуті в наступних підрозділах.

2.3.1 Модель побудови на бінарних значення

Модель простору з бінарними значеннями являє собою недетерміновану модель системи негативної автентифікації, яка ґрунтується на гіперкубі. В основі даної моделі був покладений алгоритм з використанням V- Detector.

Алгоритм V- Detector використовує модель, в якій кожне представлення даних користувача відображається у просторі з певним n виміром. Шляхом аналізу цього простору, V-Detector генерує певні негативні значення, потрапляння на яких може свідчити на потенційну атаку або спробу несанкціонованого доступу.

Важливою особливістю V-Detector є його здатність до недетермінованого створення детекторів, тобто генерувати кожного разу унікальні значення, що дозволяє розпізнавати нові типи атак і адаптуватися до нових типів загроз. Представлений алгоритм заснований на аналізі властивостей існуючих та згенерованих даних і використання певних метрик для визначення відстані між даними.

Повертаючись до реалізації простору з бінарними значеннями варто зазначити, що він є дискретним і використовує лише два можливих значення (0 або 1) для кожного виміру. Кожне інформаційне представлення, в даному випадку розглядається зашифровані ім’я та пароль користувача, відображається у n-вимірному бінарному просторі, де n - довжина рядка представлення. Дана модель гарантує, що різні комбінації імен користувача та паролів не будуть мати спільні точки у бінарному просторі. Кількість елементів у моделі простору з бінарними значеннями становить 2^n, де n - розмірність простору.

Візьмемо для прикладу бінарний простір де розмірність, тобто n, буде дорівнювати 3. Варіанти створення області будуть виглядати так, як зображено у таблиці нижче.

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Для створення детекторів вибираються точки за межами позитивної області бінарного простору, які ще не покриті наявними детекторами. Розмір детектора вибирається таким чином, щоб гіперкуб, що його представляє, не перетиналась з існуючою позитивною областю. Створені детектори покривають певну частину області, в якій не знаходяться реальні дані користувача. В даному випадку можливе перетинання між детекторами, проте детектори не мають права потрапляти у зону, де зберігаються реальній дані користувача. У даній моделі використовується так звана відстань Хеммінга. Дана міра дає можливість вимірювати відстань між будь-якими

двома точками у бінарному просторі. Відстань Хеммінга між двома об'єктами однакової довжини - це кількість позицій, на яких біти мають відмінні значення.

Нижче представлений приклад для розрахунку відстані Хеммінга, де

\+ - співпадіння

− - відмінність

0	0	1	0	1	1	0	1	1	1
+	−	+	−	−	−	−	+	+	−
0	1	1	1	0	0	1	1	1	0

Підрахувавши кількість відмінностей можна визначити, що відстань Хеммінга дорівнює 6. Детальне представлення алгоритму на основі бінарних значень представлено у розділі _.

2.3.2 Модель побудови на дійсних значеннях

Модель простору з дійсними значеннями, також як і бінарна модель, є недетермінованою. Проте має представлення гіперсфери, яка використовує n-вимірний простір дійсних чисел для представлення позитивних та негативних регіонів, іншими словам представлення простору з паролями та детекторами. Позитивні регіони та детектори представлені гіперкубами n-вимірності, де $n \in Z+$. Залишкові регіони є також n-вимірним простором, такі регіони не покриваються жодним детектором.

Кожен вимір представленої моделі відповідає зашифрованій та сегментованій інформації для входу, яку можна побачити нижче на таблиці(4).

Початковою інформацією для входу є ім'я користувача та пароль, які після хешування стають зашифрованим даними. Зашифрована інформація розбивається на n сегментів, які в подальшому будуть розглядатися як виміри. Розмірність простору визначається кількістю отриманих сегментів.

Значення кожного сегменту в моделі простору знаходиться в діапазоні від 0 до 1. Детектори обираються з негативного регіону, який не покривається іншими детекторами, і мають визначений радіус, щоб уникнути перекриття з позитивним регіоном.

Описана вище модель може бути проілюстрована двовимірною проекцією з позитивними регіонами та детекторами, як показано на рисунку.

У даній реалізації моделі простору з дійсними значеннями використовується Евклідова відстань для обчислення відстані між будь-якими двома точками у просторі.

Варто зазначити, що Евклідова відстань - це метрика, яка вимірює відстань між двома точками у просторі. Вона базується на принципах геометрії Евкліда і використовує для цього пряму лінію для визначення найкоротшого шляху між двома точками.

Якщо брати за приклад двовимірний простір, то Евклідова відстань між двома точками (x_1, y_1) і (x_2, y_2) обчислюється за наступною формулою:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

У n-вимірному просторі більшої вимірності формула Евклідової відстані розширюється, де розрахунок відстані залежить від різниці координат у всіх вимірах простору.

2.3.3 Модель побудови на основі сітки

Модель простору на основі сітки є детермінованою на відміну від бінарних та реальних моделей кіберзахисту з використанням негативної логічної системи[25].

Ця модель негативної логічної системи заснована на сітках, і спеціально розроблена таким чином, щоб детектори створювалися без накладання один на інший. Представлена модель використовує квадратну

сітку, іншими словами звичайну матрицю N×N для представлення простору даних користувача.

Згенерована сітка зберігається як двовимірний масив цілих чисел з визначеною розмірністю. Оптимальний розмір сітки може бути вибраний для поділу простору залежно від цільового рівня захисту від злому.

Позитивна інформація, тобто інформація в якій зберігається реальні паролі, відображаються на певних комірках. Вона ідентифікуються за номером рядкова та стовпчика. Значення комірки з позитивною інформацією дорівнює 1. Відповідно ті комірки, які не містять інформацію про пароль, легко виявляються за їхніми значеннями, які у даному випадку дорівнюють нулю.

Для зберігання позитивних точок у даній моделі мережі використовуються координати (x, y) комірок для кожного поєднання ім'я користувача та його пароля. Кількість збережених координат відповідає кількості поєднань, позначеної як N. Розмір сітки є сталим для всього простору і зберігається один раз перед генерацією детекторів.

Складність представленого алгоритму для створення негативних детекторів становить O(N). Починаючи з комірки, що містить позитивну точку, де значення комірки дорівнює 1, визначається послідовність комірок, які з'єднуються одна з одною до тих пір, поки не зустрінеться інша позитивна комірка. Таким чином відбувається запевнення даної сітки детекторами.

2.3.4 Параметр заплутаності

У просторі моделі власні точки зазвичай зображуються як одна точка. Однак, щоб ввести певний ступінь заплутаності, кожна позитивна точка оточена деякими сусідніми областями. У випадку з моделлю двійкового простору кожна позитивна точка утворює гіперсферу з радіусом, що задається в запропонованому алгоритмі, з центром безпосередньо позитивній точці. Збільшення параметра заплутаності зменшує розмір негативної області, проте це не впливає на загальну кількість детекторів.

Необхідно відзначити, що кількість детекторів залежить від проміжків між гіперсферами, а не від загальної площі негативної областю.

Детальний опис впливу параметру заплутаності з прикладами та порівняннями буде розглянуто в наступному розділі під час реалізації алгоритму з використанням негативної логічної системи на бінарній основі.

Висновок до другого розділу

З розділу про негативну логічну систему можна зробити висновок, що така система є одним із варіантів забезпечення безпеки інформаційних систем. Вона базується на принципах функціонування імунної системи, зокрема на роботі Т-клітин. Модель автентифікації на основі NLS дозволяє виявляти зловмисників, які намагаються отримати доступ до захищеної інформації. Ця система має свої переваги, зокрема висока ефективність та не потребує додаткових дій зі сторони користувача. Однак, є й деякі недоліки, наприклад, відносна складність реалізації та відсутність універсального стандарту. Існують різні варіанти реалізації негативної логічної системи, що дозволяє використовувати її в різних контекстах та в залежності від обчислювальної можливості системи.

3 РЕАЛІЗАЦІЯ НЕГАТИВНОЇ ЛОГІЧНОЇ СИСТЕМИ

В даному розділі буде схематично представлена модель захисту інформаційного ресурсу з використанням негативної логічної системи, опираючись на теоретичне підґрунтя, яке було створене у попередніх розділах. Базуючись на представленій схемі, побудований та імплементований алгоритм, який буде фільтрувати підозрілі запити на автентифікацію.

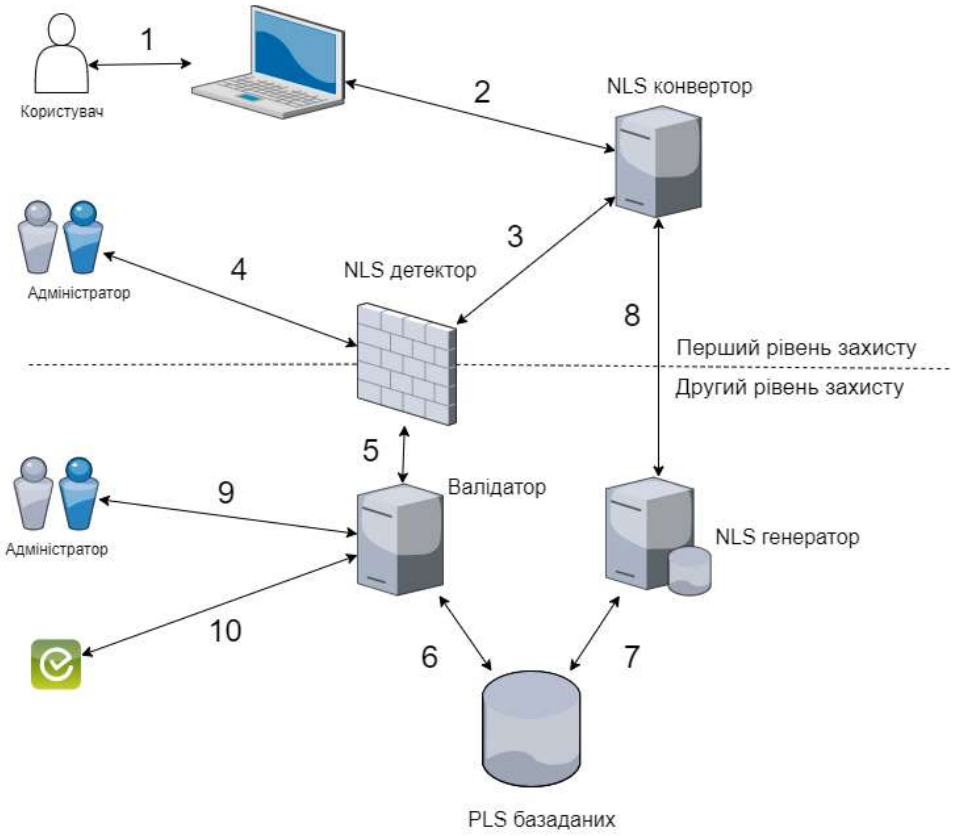
Під час реалізація були обрані наступні параметри системи:

Мова програмування	Java
Алгоритм NLS	На основі бінарних значень
Метод хешування	MD5
Кількість паролів	1000-5000
Кількість детекторів	1000-500000
Confusion parameter	1-15

Під час тестування даної системи буде продемонстроване як впливає кількість детекторів та параметр заплутаності на роботу системи в цілому.

3.1 Архітектурна модель NLS

Запропонована архітектурна модель негативної логічної системи представляє собою два рівні валідацій. Запит від користувача, який не пройшов перший рівень захисту відхиляється та не переходить до наступного етапу перевірок. Схема виглядає наступним чином:



Дана схема умовно поділена на перший рівень захисту та другий рівень захисту. Перший рівень захисту містить логіку по опрацюванню та перевірку запиту від користувача. Саме цей рівень перевіряє так відфільтровує підозрілі та не правильні запити на автентифікацію за допомогою логіну та паролю.

Варто пояснити, що означають умовні позначення перед тим як перейти до описання кроків валідацій.

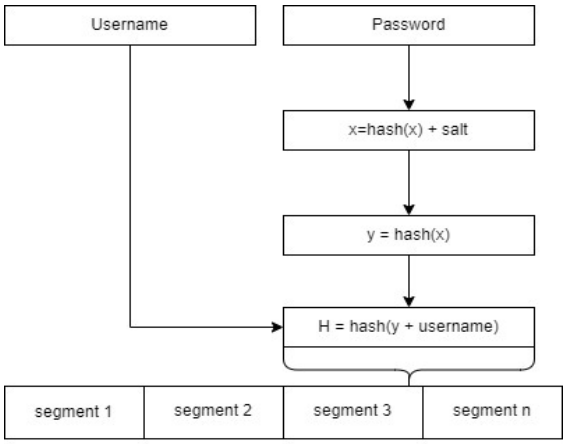
- PLS база даних – це звичайна база даних, де зберігаються паролі користувачів. Коли користувач реєструється, його пароль хешується і хешоване значення зберігається в базі даних. Для хешування паролів рекомендується використовувати "сіль". Сіль - це додатковий випадковий рядок, який додається до пароля перед хешуванням. Це підвищує безпеку, оскільки зловмисник не може використовувати райдужні хеш-таблиці для злому паролів, оскільки кожен пароль буде хешований з унікальною сіллю.
- NLS генератор – це механізм, який зчитує паролі із PLS база даних та будує NLS детектори, які будуть в майбутньому слугувати для ідентифікації підозрілого запиту.

- NLS конвертор – це сервіс, який забезпечує перетворення вхідного запиту від користувача. Спочатку логін та пароль, які надходять у чистому вигляді - хешують. Отримане хеш значення інтерпретується в бінарне значення певної розмірності. У випадку використання методу MD5, довжина вихідного значення буде послідовність із 0 та 1 розміром 128 символів.
- NLS детектор – це безпосередньо той модуль, де відбувається перевірка вхідного запиту від користувача на першому рівні захисту з використанням негативної логіки. Якщо
- PLS валідатор – це модуль, до якого доходить запит вже після того, як пройдений успішно попередній рівень захисту.
- Адміністратор – умовне позначення адміністрування та облік усіх відхилених підозрілих запитів від користувача.

Нижче наведений покроковий опис життєвого циклу валідацій та обробок вхідного запиту:

1: Починається життєвий цикл перевірки логіну та пароля користувача із вводу його через клавіатуру пристрою.

2: Запит з логіном та паролем перенаправляється до NLS конвертеру де перетворюється у 128 значне бінарне число наступним чином:



3: Перетворені параметри автентифікації направляються до множини детекторів, NLS детектор. У цьому кроці відбувається порівняння вхідного значення із кожним значенням із множини детекторів. Запит рахується успішним тільки тоді, коли не було жодного співпадіння із детектором. В цьому випадку вважається, що дані користувача коректні і перший рівень захисту пройдений.

4: Запит рахується не правильним або підозрілим у тому випадку, якщо хоча б один із детекторів виявив аномалії.

5: Даний етап свідчить про те, що запит може рахуватись безпечним і можлива перевірка з реальними паролями у базі даних.

6: На цьому етапі посилається запит на співставлення введеного користувачем логіну та пароля з тими що зберігаються у PLS базі даних.

7: Даний етап формує на самому початку детектори у модулі NLS генераторі, використовуючи реальні логіни та паролі у PLS базі даних. Варто зазначити, що NLS база даних не зберігає жодних згадок про те, які паролі зберігаються у PLS базі даних.

8: Запит на отримання множини детекторів

9: Запит рахується не правильним або підозрілим у тому випадку, якщо значення введені користувачем не дорівнюють тим значенням, які зберігаються у PLS базі даних.

10: Запит рахується успішним, якщо два рівні захисту пройдено успішно. Тільки в цьому випадку користувач отримає доступ до інформаційного ресурсу.

Важливим аспектом усієї представленої системи є те, що для користувача усі перелічені кроки відбуваються без його участі і тим самим не змушують його до виконання певних додаткових перевірок, а все зводитися до звичайного підтвердження автентифікації через логін та пароль.

Згаданий вище факт є великою перевагою використання методу заснованого на негативній логічній системі, через те що рівень захищеності усього інформаційного ресурсу зростає без надавання додаткових перевірок від користувача, таких як двофакторна автентифікація або captcha.

3.2 Імплементація бінарного алгоритму

Для реалізації механізму захисту на основі негативної логічної системи було обрано механізм з використання бінарних значень. Даний механізм був обраний через низку важливих факторів, які краще за інші методи може продемонструвати алгоритм та результати злому системи захисту з використанням негативної логіки. До таких факторів можна віднести наступні позиції:

3.2.1 Обрання тестових даних

Одним із важливих етапів реалізації алгоритму являється створення тестових даних для перевірок тих чи інших тверджень. У випадку тестування алгоритму на проходження автентифікації знадобиться наступні дані:

- Реальні паролі користувачів. Під час реалізації алгоритму будуть використанні знайдені паролі, які використовувалися або ще використовуються по сьогоднішній день. Для демонстрації алгоритму буде використовуватися вибірка із 1000 та 5000 паролів.
- Паролі для демонстрування стійкості даної системи. Буде використана множина із реальних паролів розміром 80000 одиниць.
- Мова розробки. Під час виконання було обрано високорівневу мову програмування Java та в якості середи розробки IDE IntelliJ IDEA.

3.2.2 Приклад сутностей в алгоритмі

Перед тим як перейти до тестування системи, варто продемонструвати як виглядають сутності, які взаємодіють між собою у алгоритмі заснованому на негативній логічній системі.

- Пароль користувача
  Приклад: kharchenko_vlad_2023
- Хешований пароль користувача за допомогою MD5
  Приклад: ec0d7c81346a57e36fb5f1a5c871110d
- 128 бітне представлення хешованого значення
  Приклад:
  11101100000011010111110010000001001101000110101001010111111000110110111111011010111110001101001011100100001110001000100010000011101
- Детектори. Складаються із 128 бітного значення та радіусу гіперсфери, який покриває дане бінарне значення.
  Приклад:
  {"00000100011110001010011011000111100011110100001110011000110100011011001000010010011010000110000000111111111001000000101110110111" -> "72"}
- Параметр заплутаності
  Приклад: 5

3.2.2 Приклад активації детектора на фейковому паролі

Негативна логічна система передбачає, що підозрілі запити на авторизацію, які будуть надходити від користувача, потрапляють у множини

детекторів. У разі спрацювання принаймні одного детектора – запит відхиляється та логується в системі, як підозрілий.

Далі буде наведений приклад “спрацювання” детектору на підозрілому паролі. Нехай база паролів складається із 1000 записів. Для прикладу візьмемо пароль “vlad_hacker”, якого в базі не існує.

1. База детекторів згенерована і приклад одного із детекторів(d_1) наведено нижче: d_1="10010110100111100111001001101011011111010110100110001111110101000010010001010111001010100100010001110101001010000011100101100110" -> "65", де бінарна послідовність це значення, а “65” – це радіус гіперсфери, яку покриває значення детектору.
2. Пароль vlad_hacker хешується та перетворюється у бінарне значенні з такою ж розмірність, як і значення розмірності детектору, дорівнює 128.
3. Вираховується відстань Хеммінга між значення детектора та значенням пароля. В даному прикладі відстань Хеммінга буде дорівнювати 62.
4. Відбувається перевірка на факт потрапляння у площину гіперсфери детектору. Якщо радіус детектору більший за Хеммінгову відстань, то пароль є не підозрілий, бо потрапив у площину детектору.
5. Запит на автентифікацію відхиляється та перенаправляється до адміністратору з поміткою про спробу порушення цілісності системи.

Вище наведений випадок спроби атаки на систему автентифікацію, та продемонстровано як саме система ідентифікую підозрілий запит.

3.3 Тестування системи

В даному розділі буде проведений аналіз впливу різних типів параметрів на стійкість системи, розмір даних та швидкість роботи. До таких параметрів системи відносяться наступні:

- Кількість паролів в базі даних
- Кількість згенерованих детекторів
- Значення параметру заплутаності

3.3.1 Аналіз кількості паролів

Одним із важливих факторів побудови негативної логічної системи є кількість детекторів та наявна кількість паролів у базі даних. Для досліду було обрано дві вибірки паролі, які складаються із 1000 паролів, та 5000 відповідно.

Перед тим, як перейти до аналізу кіберстійкості системи, слід зауважити, що окрім високого рівня захисту вона має бути ще швидкою у використанні. Тому нижче в таблиці наведено дані про час генерації різної кількості детекторів враховуючи існуючі паролі в базі даних.

Паролі / Детектори	1000	5000
1000	0.268 с	0.897 с
5000	0.912 с	3 с
20000	2 с	11 с
50000	6 с	28 с
100000	11 с	58 с
500000	53 с	266 с

На таблиці вище було згенеровано від 1000 до 50000 окремо для вибірки із 1000 та 5000 паролів. Як можна побачити, то час генерації детекторів на малих значеннях майже не відрізняються. Проте суттєві зміни починаються вже після 20000 детекторів і з цього можна зробити наступні висновки:

- Час генерації детекторів прямопропорційно залежить від кількості наявних паролів у базі даних. Можна споглядати дану закономірність через те, що генерація детекторів передбачає той факт, що потрібно перебрати усі значення з простору паролів бази даних з тією метою, щоб визначити чи можна використовувати детектор, чи ні.
- Варто усвідомлювати той факт, що зі зростанням кількості користувачів, що відповідно приводить до зростання кількості

паролів відбувається зростання обчислювальної потужності, яка потрібна для генерації великої кількості детекторів.

- Генерація детекторів не відбувається кожної години або кожного дня. Множина детекторів формується один раз і може бути використаною до тих пір, поки на дану системи не буде застосована атака кіберзлочинцями. Якщо адміністратор інформаційного ресурсу бачить підозрілу активність, то дана множина детекторів може бути перегенерована. Під час реєстрацій нового користувача в системі або видалення існуючого, множину детекторів теж не має потреби генерувати з нуля, а потрібно просто занести зміни в поточну множину.

Кількість детекторів впливає на захищеність від підозрілих запитів під час виконання авторизації. В наступному розділі зазначене прямопропорційне відношення кількості детекторів, та кількості хибно успішних спроб на авторизацію.

Важливо сказати, що під час тестування даної системи всі паролі, які існують в базі даних – успішно проходять перший рівень безпеки, що і очікується від представленого механізму.

3.3.2 Аналіз кількості детекторів

В даному розділі представлений аналіз впливу різної кількості детекторів на захищеність системи під час проведення автентифікації.

Під час експерименту було обрано 30000 паролів реальних користувачів, які будуть використовуватися у атаці грубою силою на перший рівень захисту негативної логічної системи. Задані наступні параметри системи:

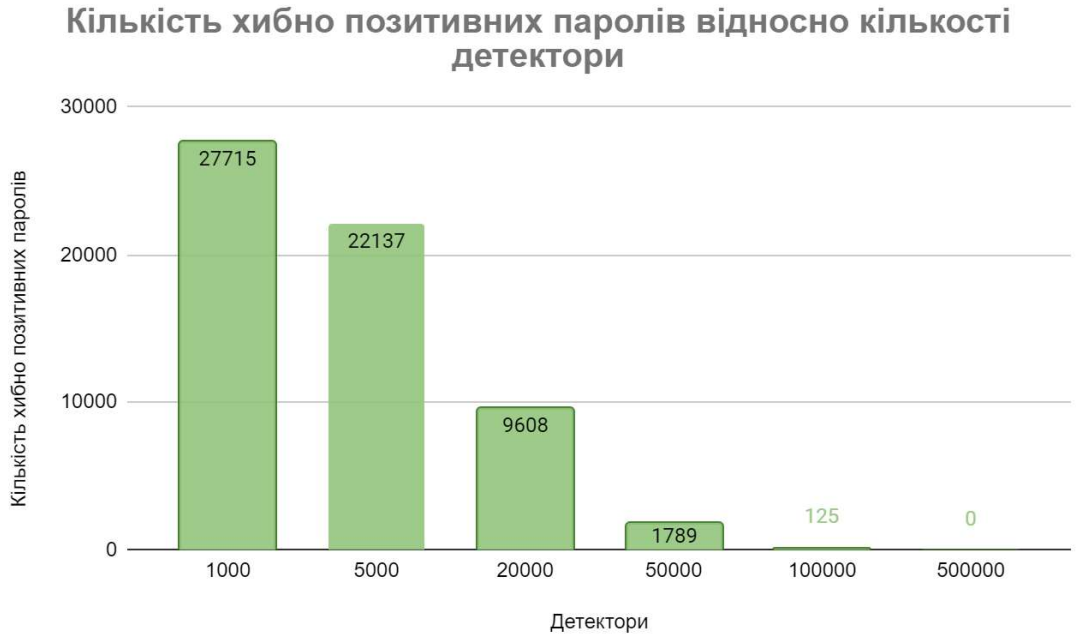
- Кількість реальних паролів в базі даних 1000.
- Кількість детекторів 20000.
- Кількість фейкових паролів 30000.
- Параметр заплутаності 5.

Отримані результати можна побачити нижче:

Детектори	Кількість хибно позитивних паролі
1000	27715
5000	22137
20000	9608
50000	1789

100000	125
500000	0

Візуалізація впливу кількості детекторів на криптографічну цілісність системи.



Важливо зазначити, що в разі проходження негативної логічної системи, що вже свідчить про певні проблеми під час побудови алгоритму, хибний запит проходить до другого рівня захисту. На другому рівні вже йде пряме співставлення з реальним паролем і вже тоді підозрілий запит не завадить шкоди інформаційному ресурсу.

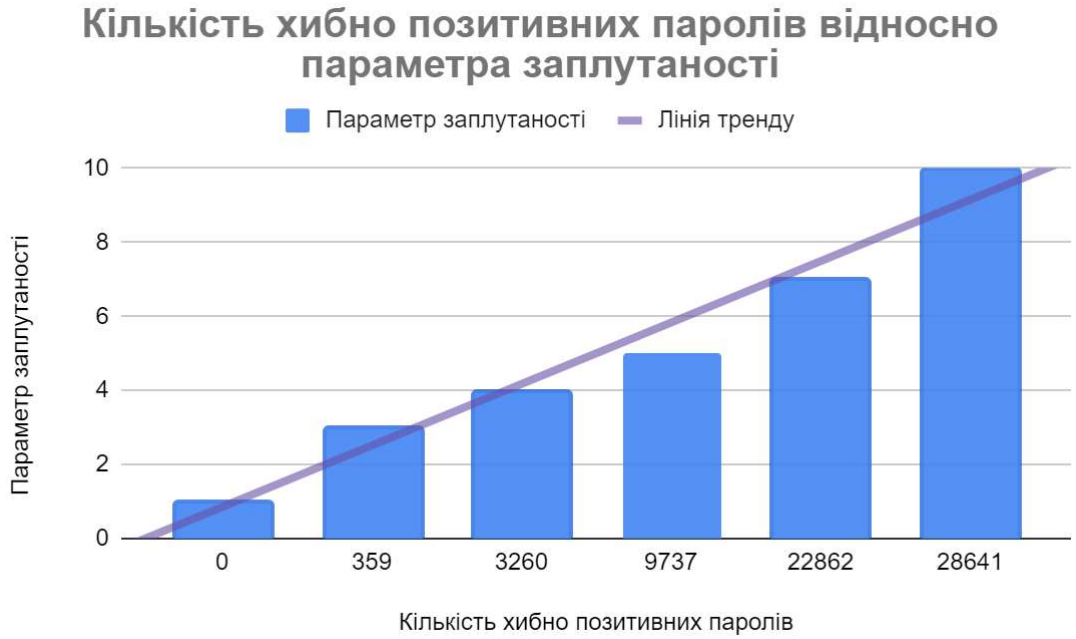
Як можна побачити з графіку вище, то кількість детекторів напряму залежить на стійкість системи до атак. Проте створення великої кількості детекторів потребує виділення певного об’єму пам’яті, що може бути критичним у разі стрімкого зростанням кількості користувачів. І в цей час на допомогу приходить параметр заплутаності, який описаний в наступному розділі.

3.3.3 Аналіз значення параметру заплутаності

Кількість детекторів напряму залежить на стійкість системи до атак. Гостро постає питання пам’яті для зберігання великої кількості детекторів, і в цьому випадку на допомогу приходить параметр заплутаності. Параметр заплутаності це звичайне число, яке впливає на радіус гіперсфери під час створення детекторів. Чим менше значення становить параметр заплутаності, тим строгіше до помилок реагує негативна логічна система.

Кореляція параметру заплутаності та кількості хибно позитивних паролів, які успішно пройшли валідацію, наведено нижче:

Параметр заплутаності	Кількість хибно позитивних паролів
10	28641
7	22862
5	9737
4	3260
3	359
1	0



Як можна побачити вище, то при коефіцієнті заплутаності 1 – жодний хибний запит рахується за підозрілий. З однієї сторони, це показує працездатність даного методу під час протистояння атакам на систему автентифікації. Проте з іншої сторони користувачі інформаційного ресурсу – це люди, які здатні робити помилки під час набору тексту, в тому числі і паролів. З цього можна зробити висновок, що встановлювати строгий параметр заплутаності має користь для безпеки ресурсу, проте може хибно блокувати звичайних користувачів. Натомість краще використати адекватний параметр заплутаності. Приклад: користувач має пароль kharchenko_vlad_2023. При налаштуванні параметру

заплутаності, наприклад 1, введений користувачем пароль kharchenko_vlad_2022 буде рахуватися підозрілим, хоча помилка лише в одному останньому символі. Після підвищення параметру заплутаності останній пароль буде проходити перший рівень безпеки, не відправить повідомлення адміністратору про підозрілу активність і врешті-решт не пройде перевірку з базою даних і користувачу повернеться помилка.

Враховуючи вище наведені експерименти, представлена негативна логічна система буде в рази ускладнювати зловмисникам отримати доступ до інформаційного ресурсу через проходження несанкціонованої автентифікації. За рахунок впровадження додаткового прошарку детекторів, які слідкують за аномальної поведінкою, зловмиснику не вдасться успішно пройти його через спрацювання детекторів, після чого запит з його адреси буду вважатися підозрілим.

Отже, без додаткового впливу на звичайного користувача, дана негативна логічна система підвищує захищеність інформаційного ресурсу під час спроб атак на автентифікацію.

Висновки до третього розділу

У цьому розділі дипломної роботи були розглянуті різні аспекти негативної логічної системи (NLS). Була описана архітектурна модель NLS, яка базується на імунній системі та принципах функціонування Т-клітин. Також було розглянуто модель автентифікації на основі NLS та варіанти її реалізації.

Детально була описана імплементація бінарного алгоритму та приклад активації детектора на фейковому паролі. Було проведено тестування системи та аналізовано кількість детекторів та значення параметру заплутаності.

На основі проведених досліджень можна зробити висновок, що негативна логічна система є ефективним способом захисту від злому паролів.

Вона може бути успішно застосована в різних областях, де потрібна надійна автентифікація, таких як банківська система або корпоративні мережі.

4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

Існуючі механізми атаки та захисту безпеки базуються на механізмах системи позитивної логіки (Positive Logic System), тобто стан опису атаки та захисту є позитивним по відношенню до логічного опису атаки та захисту. Суть атаки і захисту безпеки полягає у вартості і витратах як для атакуючої, так і для захищаючої сторони, які вони здійснюють під час атаки і захисту. На основі інформаційної невідповідності, ступеня протистояння, статусу переваги і неповноцінності, а також активного і пасивного стану як атакуючої, так і захисної сторони можна покладатися тільки на вартість і витрати тактики кібератаки і захисту. Таким чином, недоліком існуючих механізмів нападу і захисту на основі PLS є обмеження інформаційної невідповідності між атакуючим і захисними діями.

4.1 Опис ідеї стартап-проекту

Основною метою стартапу буде розробка і впровадження нових механізмів нападу і захисту, які будуть базуватися на механізмах системи негативної логіки (Negative Logic System).

Система негативної логіки буде використовувати негативний стан опису атаки та захисту, що дозволить знизити інформаційну невідповідність між атакуючими та захисними діями. Замість того, щоб покладатися на вартість і витрати тактики кібератаки і захисту, система буде використовувати інформаційну невідповідність як фактор рішення.

Зокрема, стартап може розробити нову систему виявлення загроз на основі машинного навчання, яка буде використовувати систему негативної логіки для аналізу поведінки користувачів та ідентифікації потенційних загроз.

Також можуть бути розроблені нові методи захисту, які будуть використовувати систему негативної логіки для зниження інформаційної невідповідності та забезпечення більш ефективного захисту від кібератак.

Основною перевагою такого стартапу буде те, що він пропонує нові підходи до захисту від кібератак, які можуть бути більш ефективними за традиційні підходи на основі системи позитивної логіки. Також цей стартап може бути цікавим для різних організацій, які прагнуть забезпечити безпеку своєї інформації та мережі перед можливими кібератаками.

Таблиця 5.1 – Порівняння схожих методів оцінки стійкості

№	Техніко-економічні характеристики ідеї	Мій додаток	NTLM	Слабка сторона	Нейтральна сторона	Сильна сторона
1	Робота з серверами, ПК та БД	Високий	Середня			+
2	Врахування важливих аспектів	Висока	Середня			+
3	Документація	Слабко описана	Слабко описана		+	

4.2 Технологічний аудит ідеї проекту

Щодо технологічного аудиту ідеї, варто врахувати, що розробка системи виявлення загроз на основі машинного навчання може вимагати значних зусиль з боку команди розробників, а також великих витрат на збір та аналіз

великих обсягів даних.
Крім того, варто врахувати можливі ризики, пов'язані зі зберіганням та обробкою великих обсягів даних, що можуть включати конфіденційну та особисту інформацію користувачів. Тому важливо враховувати правові та етичні аспекти розробки та використання такої системи.

Усі ці фактори потрібно уважно взяти до уваги при подальшому розвитку ідеї та створенні бізнес-стратегії для стартапу.

4.3 Аналіз можливості запуску стартап-проекту на ринок

Щоб зробити аналіз ринкових можливостей для запуску стартап проекту, необхідно проаналізувати кілька ключових факторів:

Розмір ринку: Вартість глобального ринку кібербезпеки в 2020 році становила 173 мільярди доларів і очікується, що до 2027 року вона зросте до 270 мільярдів доларів. Це свідчить про те, що ринок кібербезпеки є великим і швидко розвивається, що створює можливості для нових стартапів, таких як запропонований.

Конкуренція: Ринок кібербезпеки є дуже конкурентним, з багатьма гравцями на ринку, включаючи такі великі компанії, як Symantec, Kaspersky Lab, McAfee та інші. Тому стартап повинен бути готовим до сильної конкуренції на ринку.

Потенційні клієнти: Потенційними клієнтами для стартапу можуть бути різні види організацій, включаючи компанії з малим, середнім та великим бізнесом, державні установи, міжнародні організації тощо, які мають високий рівень залежності від цифрових технологій і повинні забезпечувати свою кібербезпеку.

В таблиці 5.2 наведено характеристику потенційного ринку стартап-проекту.

Таблиця 5.2 – Характеристика потенційного ринку стартап-проекту

№	Характеристика стану ринку	Показник стану ринку
1	Динаміка ринку (якісна оцінка)	Зростає
2	Наявність обмеження для входу	Відсутність документації
3	Специфічні вимоги стандартизації	Немає

Кількісну оцінку надати неможливо, але потенційно кожний підрозділ вищих навчальних закладів може використовувати запропонований прототип системи для забезпечення безпеки цифрових даних для своїх цілей та забезпечити веб-стосунок всією необхідною інформацією для взаємодії між студентами та викладачам

Результати представлені в таблиці 5.3

Таблиця 5.3 - Фактори ризику

№	Назва фактору	Опис	Можливе рішення
1	Мала інформованість	Через брак документації та опису в мережі інтернет розробники можуть не знати переваги аутентифікації	Розробка методу аутентифікації з інформацією та документацією про даний застосунок

Продовження таблиці 5.3

2	Кібератаки з метою заволодіння персональними даними	Через недостатні рівень захищеності зловмисник може заволодіти необхідними йому персональними даними, які в майбутньому він може використати в зловмисних цілях	Зробити метод аутентифікації більш безпечним

В таблиці 5.4 описано результати аналізу конкурентного середовища.

Таблиця 5.4 - Аналіз конкурентного середовища

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентно спроможними)
1. Вказати тип конкуренції: Чиста	Багато спеціалістів та систем	Розробити продукт, що вигравatиме у швидкодії
2. За рівнем конкурентної боротьби: міжнаціональна	Системи не мають національної прив’язки	Описувати використання методу міжнародною мовою
3. За галузевою ознакою: міжгалузева	Застосовується в різних цілях	Розирювати середовища застосування
4. Конкуренція за видами товарів:	Наявність конкуренції з іншими методами	Покращувати документацію та опис

товарно-родова		методу в мережі
5. За характером конкурентних переваг: Нецінова	Методи представляють собою різні математичні алгоритми	Збільшувати популярність обраного методу
6. За інтенсивністю: Марочна	Метод надає власні унікальні переваги	Розвивати метод

Далі необхідно провести аналіз конкуренції в даній галузі за Майклом Портером. Результати аналізу наведено в таблиці 5.4

Таблиця 5.4 – Аналіз конкуренції за Майклом Портером

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	3	Стратегія спеціалізації	Середня інтенсивність конкуренції та відносна простота, зберігаючи якість продукту	Стратегія диференційованого маркетингу

Результати аналізу підтверджують, що на ринку сприятливі умови для створення і запуску даного стартап-проекту. Були сформульовані та обґрунтовані перелік факторів конкурентоспроможності. Аналіз оформлено в таблицю 5.5

Таблиця 5.5 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Удосконалення під сьогоденну ситуацію в країні	Застосунок може використовуватись кафедрами для більш деталізованого ведення обліку та для взаємодії між студентами та викладачами
2	Доступність програмного продукту	Програмний продукт може використовуватись на будь-якому пристрої

Після проведення аналізу можна виділити сильні та слабкі (які потребують вдосконалення) сторони продукту (таблиця 5.6).

Таблиця 5.6 – Порівняльний аналіз сильних та слабких сторін системи

№	Фактор конкурентоспроможності	Бали 1-10	Рейтинг товарів-конкурентів						
			-3	-2	-1	0	1	2	3
1	Удосконалення під сьогоденну ситуацію в країні	9							+
2	Доступність програмного продукту	10							+

SWOT-аналіз дозволяє оцінити можливості та загрози бізнесу, а також сильні і слабкі сторони продукту. Результати наведені у таблиці 5.7

Таблиця 5.7 – SWOT-аналіз стартап-проекту

Сильні сторони: Якість та аудиторія	Слабкі сторони: Відсутність документації
Можливості: використання на будь-яких пристроях	Загрози: Вірогідність не виходу в маси

На основі SWOT-аналізу було спроектовано альтернативну ринкову поведінку для інтеграції стартап-проекту (таблиця 5.8).

Таблиця 5.8– Альтернативна ринкова поведінка

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Робота на різних ПК	50%	6 місяців
2	Участь в конференціях та популярних виступах	50%	6 місяців

4.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії продукту

Розроблення ринкової стратегії розпочинається з визначення стратегії охоплення ринку. Для цього необхідно охарактеризувати цільові групи потенційних споживачів (таблиця 5.9).

Таблиця 5.9 – Вибір цільових груп потенційних споживачів

№	Опис профілю	Готовність	Орієнтовний	Інтенсивність	Простота

	цільової групи потенційних клієнтів	споживачів сприйняти продукт	попит в межах цільової групи (сегменту)	конкуренції в сегменті	входу у сегмент
1	Власники ПК	Висока	90%	Висока	Середня
2	Власники організацій	Висока	10%	Середня	Низька
Які цільові групи обрано:1					

Для роботи в обраних сегментах ринку сформуємо базову стратегію розвитку (таблиці 5.10, 5.11, 5.12).

Таблиця 5.10 – Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	3	Стратегія спеціалізації	Середня інтенсивність конкуренції та відносна простота, зберігаючи якість продукту	Стратегія диференційованого маркетингу

Таблиця 5.11 – Визначення базової стратегії конкурентної поведінки

№	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки *
1	Українському - так	так	Так, реалістичність	Стратегія лідеру

Таблиця 5.12 – Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Швидкодія, реалістичність	Стратегія диференційованого маркетингу	Якість, висока функціональність, велика кількість інформації	Імідж, якість, спеціалізація

4.5 Розроблення маркетингової програми проекту

Розроблення маркетингової програми стартап-проекту

Кінцевим етапом є формування маркетингової концепції товару. Спочатку необхідно визначити ключових переваг концепції потенційного товару (таблиця 5.13). Також опишемо три рівні моделі товару (таблиця 5.14).

Таблиця 5.13 – Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Оцінка Захищеності цифрових даних	Безпека	
2	Документація, наукові праці	Велика аудиторія	Унікальність

Таблиця 5.14 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
I. Товар за задумом	Систем захисту цифрових даних
II. Товар у реальному виконанні	Програмний продукт – прототип веб-застосунку
	Якість: доказана експериментально
	Пакування - відсутнє
	Марка: назва організації-розробника + назва товару

III. Товар із підкріпленням	Після продажу: гарантія якості та оновлення
За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності	

Далі необхідно визначити цінові межі продукту (таблиця 5.15).

Таблиця 5.15 – Цінові межі продукту

№	Рівень цін на товари-замінники	Рівень цін на товари- аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	500	1000	700 млн	500-2000

Висновки до четвертого розділу

У розділі номер 5 було розглянуто ідеї стартап – проекту, також проведено технологічний аудит запланованого проекту, аналіз можливостей ринку, розроблено ринкову та маркетингову стратегію для потенційного продукту. Отже, розвиваючи та розповсюджуючи інформацію про удосконалений метод утентифікації, його доцільно використовувати для створення стартап-проектів, використання даного методу є економічно вигідним, такий проект має перспективи тривалого сталого розвитку.

ВИСНОВКИ

В рамках даної дипломної дисертації було проведено широкий аналіз та дослідження питань, пов'язаних з доступом до інформаційного ресурсу, методами автентифікації та елементами системи автентифікації. Дослідження було спрямоване на з'ясування принципів функціонування цих систем, аналіз існуючих методів та їх недоліків, а також виявлення можливих варіантів їх покращення та оптимізації.

З початку було розглянуто питання доступу до інформаційного ресурсу, включаючи процеси ідентифікації, автентифікації та авторизації. Були проаналізовані різні методи автентифікації, такі як використання паролів, біометричних даних, токенів та інших факторів. Крім того, вивчені були елементи системи автентифікації, такі як цифрові підписи, сертифікати, хеш-функції та протоколи обміну даними.

Дослідження також охопило аналіз використання захищеного пароля, його моделі використання та важливість довжини та міцності пароля для забезпечення безпеки.

У контексті безпеки були розглянуті методи зламу пароля, включаючи атаки перебору, словникові атаки, атаки з використанням розумного словника та інші техніки. Було проведено аналіз їхніх недоліків та рекомендації щодо запобігання таким атакам.

Подальше дослідження було спрямоване на негативну логічну систему (NLS) та її реалізацію в контексті моделі автентифікації. Була описана архітектурна модель NLS, яка базується на імунній системі та принципах функціонування Т-клітин. Також було розглянуто модель автентифікації на основі NLS та варіанти її реалізації.

Детально була описана імплементація бінарного алгоритму та приклад активації детектора на фейковому паролі. Було проведено тестування системи та аналізовано кількість детекторів та значення параметру заплутаності.

На основі проведених досліджень можна зробити висновок, що негативна логічна система є ефективним способом захисту від злому паролів. Вона може бути успішно застосована в різних областях, де

потрібна надійна автентифікація, таких як банківська система або корпоративні мережі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Thomas, S.: Decidability for some justification logics with negative introspection. J. Symb. Log. 78(2), 388–402, 2013.
2. 15th International Annual Conference, Research on Identity Authentication Method, 2018.
3. Wu, L., Zhang, Y., Choo, K.K.R., He, D.: Efficient and secure identity-based encryption scheme with equality test in cloud computing. Future Gener. Comput. Syst. 73, 22–31, 2017.
4. NordPass [Електронний ресурс] – The most popular passwords. https://nordpass.com/most-common-passwords-list/
5. Dasgupta, D., Saha, S.: Password security through negative filtering. In: Emerging Security Technologies (EST), 2010 International Conference on, pp. 83–89, 2010.
6. Techtarget [Електронний ресурс] – Why is authentication important in cybersecurity: https://www.techtarget.com/searchsecurity/definition/authentication
7. Smith, R.E.: Authentication: from passwords to public keys. Addison-Wesley Longman Publishing, 2017.
8. Wu, L., Zhang, Y., Xie, Y., Alelaiw, A., Shen, J.: An efficient and secure identity-based authentication and key agreement protocol with user anonymity for mobile devices. Wirel. Pers. Commun. 94(4), 2017.

9. Bojinov, H., Bursztein, E., Boyen, X., Boneh, D.: Kamouflage: Loss-resistant password management. In: Computer Security, pp. 286–302. Springer, 2015.
10. M. Ayara, J. Timmis, R. De Lemos, L. De Castro, R. Duncan, “Negative Selection: How to generate detectors”, Proceedings of the 1st International Conference on Artificial Immune System, 2018.
11. Camenisch, J., Lehmann, A., Neven, G.: Optimal distributed password verification. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 182–194. ACM, 2015.
12. Ji, Z.: Negative selection algorithms: From the thymus to v-detector. P Ji, Z., Dasgupta, D.: V-detector: An efficient negative selection algorithm with probably adequate detector coverage. Inf. Sci. 179(10), 1390–1406
13. Juels, A., Rivest, R.L.: Honeywords: Making password-cracking detectable. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 145–160. ACM, 2016.
14. Ori, L., João, M., Yoni, Z.: Sequent systems for negative modalities. Log. Univ. 11(3), 345– 382, 2017.
15. Buchman, D., Poole, D.: Negative probabilities in probabilistic logic programs. Int. J. Approx. Reason. 83, 43–59, 2017.
16. Харченко, В., & Гальчинський, Л. (2023). ПІДВИЩЕННЯ РІВНЯ БЕЗПЕКИ АВТЕНТИФІКАЦІЇ З ВИКОРИСТАННЯМ АЛГОРИТМУ НЕГАТИВНОГО ВІДБОРУ. Collection of scientific papers «ΛΌГОΣ», (May 26, 2023; Boston, USA), 158-162.
17. Habib, H., Colnago, J., Melicher, W., Ur, B., Segreti, S.M., Bauer, L., Christin, N., Cranor, L.F, 2017.
18. Furnell, S., Esmael, R., Yang, W., Li, N. Enhancing security behaviour by supporting the user. Computers & Security, 75: 1-9, 2018.

19. Grassi, P.A., Garcia, M.E., Fenton, J.L. Digital Identity Guidelines, 2017.
20. L. Lamport, “Password authentication with insecure communication”, Communications of the ACM.
21. D. de Borde, "Two-factor authentication", Siemens Insight Consulting.
22. Murray, H., Malone, D. Evaluating password advice. In: 28th Irish Signals and Systems Conference, 2017.
23. Knieriem, B., Zhang, X., Levine, P., Breitinger, F., Baggili. An overview of the usage of default passwords. In: Matoušek, P., Schmiedecker, M. (eds.) Digital Forensics and Cyber Crime, 2018.
24. Dasgupta, Dipankar, Abhijit Kumar Nag, Denise Ferebee, Sanjib Kumar Saha, Kul Prasad Subedi, Arunava Roy, Alvaro Madero, Abel Sanchez, and John R. Williams. “Design and Implementation of Negative Authentication System.” International Journal of Information Security 18, no. 1: 23–48, 2017.
25. Yexia Cheng, Yuejin Du1, Jin Peng , Jun Fu and Baoxu Liu1. “Research on Identity Authentication Method Based on Negative Logic System”, CNCERT 2018, CCIS 970, pp. 3–15, 2019.

ДОДАТКИ

ДОДАТОК А

Програмний код алгоритму

```
package com.kharchenko55;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

class NlsImpl {
    private static final int DIMENSION = 128;
    private static final int DETECTORS_COUNT = 20000;
    private static final int CONFUSION_PARAMETER = 1;



    Map<String, Integer> generateDetectors(List<String> selfPoints) {
        List<String> copy = new ArrayList<>(selfPoints);
        Map<String, Integer> detectorPoints = new ConcurrentHashMap<>();
        while (detectorPoints.size() < DETECTORS_COUNT) {

            String x = generateBinaryString(DIMENSION);
            int minimumSelfDistance = getMinimumSelfDistance(copy,
CONFUSION_PARAMETER, x, DIMENSION);
            if (minimumSelfDistance > 0) {
                boolean checkDetector = checkDetector(detectorPoints, x, minimumSelfDistance);
                if (checkDetector) {
                    detectorPoints.put(x, minimumSelfDistance);
                }
            }
```

```
    }

    return detectorPoints;
}

static boolean verifyInputRequest(String password, Map<String, Integer> detectorPoints) {
    int d;
    for (Map.Entry<String, Integer> entry : detectorPoints.entrySet()) {
        String s = entry.getKey();
        Integer radius = entry.getValue();

        d = hammingDist(password, s);
        if (d <= radius) {
            return false;
        }
    }

    return true;
}

private static boolean checkDetector(Map<String, Integer> detectorPoints, String x, int r) {
    if (detectorPoints.isEmpty()) {
        detectorPoints.put(x, r);
    }
    for (Map.Entry<String, Integer> entry : detectorPoints.entrySet()) {
        String s = entry.getKey();
        Integer radius = entry.getValue();
        int dDiff = Math.abs(radius - r);
        int dSum = radius + r;
        int dCenter = hammingDist(s, x);

        return dCenter >= dDiff && dCenter >= dSum / 2;
    }
    return true;
```

```
    }

    private static int getMinimumSelfDistance(List<String> selfPoints, int
confusingParameter, String x, int r) {
        int d;
        List<Integer> dis = new ArrayList<>(selfPoints.size());
        for (int i = 0; i < selfPoints.size(); i++) {
            d = hammingDist(selfPoints.get(i), x);
            if (d - confusingParameter < r) {
                r = d - confusingParameter;
                dis.add(r);
            }

        }

        return Collections.min(dis);
    }

    private static int hammingDist(String str1, String str2) {
        int i = 0, count = 0;
        while (i < str1.length()) {
            if (str1.charAt(i) != str2.charAt(i))
                count++;
            i++;
        }
        return count;
    }

    private static String generateBinaryString(int N) {
        StringBuilder s = new StringBuilder();
        for (int i = 0; i < N; i++) {
            int x = findRandom();
            s.append(x);
        }
```

```
        return String.valueOf(s);
    }

    private static int findRandom() {
        return (1 + (int) (Math.random() * 100)) % 2;
    }
}
```

ДОДАТОК Б

Тестування системи

```
package com.kharchenko55;

import org.junit.jupiter.api.Test;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

import static org.junit.jupiter.api.Assertions.assertEquals;

class NlsImplTest {

    @Test
    void verifyTheRandomPasswordFail() throws IOException {
```

```
        NlsImpl nls = new NlsImpl();
        Map<String, Integer> detectors;
        List<String> self = new ArrayList<>();
        List<String> randomPass =
    Files.lines(Paths.get("30000.txt")).collect(Collectors.toList());
        List<String> barePass = Files.lines(Paths.get("top1000.txt")).collect(Collectors.toList());
        randomPass.removeAll(barePass);

        for (String pass : barePass) {
            String md5Custom = md5Custom(pass);
            String binary = convertStringToBinary(md5Custom);
            self.add(binary);
        }
        Instant start = Instant.now();
        detectors = nls.generateDetectors(self);

        int count = 0;
        for (String randomPass1 : randomPass) {
            String binaryHasedPassword = convertBareStringToBinary(randomPass1);
            boolean result = NlsImpl.verifyInputRequest(binaryHasedPassword, detectors);
            if (result) {
                count++;
            }
        }

        assertEquals(0, count);
    }


    @Test
    void verifyTheAllPasswordsFromFileShouldSuccess() throws IOException {
        NlsImpl nls = new NlsImpl();
        int count = 0;
        List<String> self = new ArrayList<>();
```

```
    Map<String, Integer> detectors;
    List<String> barePass = Files.lines(Paths.get("top1000.txt")).collect(Collectors.toList());
    for (String pass : barePass) {
        String md5Custom = md5Custom(pass);
        String binary = convertStringToBinary(md5Custom);
        self.add(binary);
    }
    detectors = nls.generateDetectors(self);
    for (String correctPass : self) {
        boolean result = NlsImpl.verifyInputRequest(correctPass, detectors);
        if (!result){
            count++;
        }
    }
    assertEquals(0, count);
}

private String convertBareStringToBinary(String password) {
    String md5Custom = md5Custom(password);
    return convertStringToBinary(md5Custom);
}

static String md5Custom(String input) {
    try {
        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] messageDigest = md.digest(input.getBytes());
        StringBuilder hexString = new StringBuilder();
        for (byte b : messageDigest) {
            hexString.append(String.format("%02x", b));
        }
        return hexString.toString();
    } catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
```

```
    }
}
```