ЖИТОМИРСЬКИЙ ВІЙСЬКОВИЙ ІНСТИТУТУ ІМЕНІ С.П.КОРОЛЬОВА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

здобувача вищої освіти ступеня «бакалавр» заочної форми навчання
за спеціальністю «Кібербезпека»

Тема: «АНАЛІЗ ТЕХНОЛОГІЙ ПРОТИДІЇ СОЦІАЛЬНІЙ ІНЖЕНЕРІЇ»

Виконавець: Олег ГЕОРГІЦА

м. Житомир

2026

АНОТАЦІЯ

Кваліфікаційна робота містить 50 сторінок тексту, 11 рисунків, 9 таблиць, 3 додатки, 26 використаних джерел.

Мета роботи – розробити програмний засіб для підтримки протидії соціальній інженерії шляхом авторизації користувачів, збереження даних у базі Microsoft Access та аналізу повідомлень на наявність ознак соціально-інженерних атак.

Об’єкт дослідження – процес протидії атакам соціальної інженерії в інформаційних системах організацій.

Предмет дослідження – методи, технології та програмні засоби виявлення ознак соціальної інженерії, організації захисту користувачів і зниження ризиків, пов’язаних із людським фактором.

Методи дослідження: аналіз джерел використано для вивчення сучасного стану проблеми соціальної інженерії; класифікацію – для систематизації видів атак; порівняльний аналіз – для оцінювання організаційних, технічних і програмних засобів протидії; системний підхід – для визначення взаємозв’язку між заходами захисту; алгоритмізацію, програмування та тестування – для розроблення і перевірки консольного застосунку.

У роботі проаналізовано сучасні форми соціально-інженерних атак, зокрема фішинг, спір-фішинг, вішинг, смішинг, претекстинг, бейтинг і тейлгейтинг. Розглянуто організаційні технології, технічні засоби, програмні рішення та інструменти моделювання атак. Розроблено консольний програмний засіб мовою Python з авторизацією користувачів, хешуванням паролів, блокуванням після невдалих спроб входу, розподілом ролей, аналізом повідомлень за індикаторами ризику, збереженням результатів у базі Microsoft Access і журналюванням подій безпеки. Новизна результатів полягає в подальшому розвитку навчально-демонстраційного підходу до виявлення ознак соціальної інженерії на основі поєднання організаційних, технічних і програмних заходів.

Галузь застосування результатів – навчальні та демонстраційні процеси у сфері інформаційної безпеки й кібербезпеки, а також підготовка користувачів до розпізнавання типових ознак соціально-інженерних атак. Економічна ефективність окремо не розраховувалась, оскільки робота має навчально-демонстраційне спрямування.

СОЦІАЛЬНА ІНЖЕНЕРІЯ, ФІШИНГ, КІБЕРБЕЗПЕКА, ІНФОРМАЦІЙНА БЕЗПЕКА, АВТОРИЗАЦІЯ, MICROSOFT ACCESS, PYTHON, ІНДИКАТОР РИЗИКУ, ЖУРНАЛЮВАННЯ, АНАЛІЗ ПОВІДОМЛЕНЬ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних.

ІБ – інформаційна безпека.

ШІ – штучний інтелект.

URL – адреса вебресурсу в мережі Інтернет.

SMS – коротке текстове повідомлення мобільного зв’язку.

MFA – багатофакторна автентифікація.

SPF – механізм перевірки права поштового сервера надсилати листи від імені певного домену.

DKIM – механізм криптографічної перевірки справжності електронного листа.

DMARC – механізм перевірки електронної пошти на основі SPF і DKIM.

EDR – засіб виявлення та реагування на підозрілу активність на кінцевих пристроях.

DLP – система запобігання витоку конфіденційної інформації.

ODBC – інтерфейс підключення програм до баз даних.

ВСТУП

У сучасних умовах розвитку інформаційних технологій питання захисту інформації набуває особливого значення для державних установ, військових організацій, підприємств та окремих користувачів. Поширення електронного документообігу, дистанційної роботи, онлайн-сервісів і цифрових каналів комунікації створює не лише нові можливості для обміну даними, а й додаткові ризики для інформаційної безпеки [1; 2].

Однією з небезпечних загроз у сфері кібербезпеки є соціальна інженерія. Її особливість полягає в тому, що об’єктом впливу виступає не стільки технічна система, скільки людина, її довіра, необізнаність, емоційний стан або помилки під час прийняття рішень [4; 7; 10]. Зловмисники можуть використовувати психологічний тиск, авторитет, терміновість, страх або імітацію офіційного звернення для отримання конфіденційної інформації, облікових даних чи доступу до інформаційних ресурсів [7; 9; 10].

Актуальність теми зумовлена тим, що атаки соціальної інженерії залишаються поширеним способом реалізації кіберзагроз. До основних форм таких атак належать фішинг, спір-фішинг, вішинг, смішинг, претекстинг, бейтинг та інші методи маніпулятивного впливу. Значна частина інцидентів інформаційної безпеки пов’язана саме з людським фактором, недостатньою обізнаністю користувачів і порушенням правил безпечної роботи з інформацією [20; 21; 22].

Проблема соціальної інженерії є особливо важливою для організацій, які працюють із конфіденційною, службовою або критично важливою інформацією. Успішна соціально-інженерна атака може призвести до компрометації облікових записів, витоку даних, несанкціонованого доступу до інформаційних систем, фінансових втрат або створення умов для подальших кібератак. Для України ця проблема має додаткове значення у зв’язку з необхідністю підвищення стійкості державних, військових та інших організацій до інформаційних і кібернетичних загроз.

Ефективна протидія соціальній інженерії потребує комплексного підходу [17; 18; 19]. Вона має поєднувати організаційні заходи, навчання персоналу, технічні засоби контролю, багатофакторну автентифікацію, фільтрацію повідомлень, журналювання подій та використання програмних рішень для виявлення ознак потенційних атак. Саме тому доцільним є розроблення навчально-демонстраційного програмного засобу, який дозволяє перевіряти повідомлення на наявність характерних індикаторів соціальної інженерії.

Метою кваліфікаційної роботи є аналіз технологій протидії соціальній інженерії та розроблення програмного засобу, який забезпечує авторизацію користувачів, збереження даних у базі Microsoft Access і перевірку повідомлень на наявність характерних ознак соціально-інженерних атак.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

1. Проаналізувати сучасний стан реалізації атак соціальної інженерії.
2. Розглянути основні підходи та методи протидії соціальній інженерії.
3. Обґрунтувати актуальність запровадження сучасних технологій протидії соціальній інженерії.
4. Дослідити організаційні технології протидії.
5. Проаналізувати технічні засоби і програмні рішення для захисту від соціально-інженерних атак.
6. Розглянути інструменти моделювання та тестування атак соціальної інженерії.
7. Визначити підходи до створення системи протидії соціальній інженерії.
8. Розробити консольний програмний засіб на Python з авторизацією, елементами безпеки, базою даних Microsoft Access і функцією аналізу повідомлень.
9. Сформувати рекомендації щодо застосування технологій протидії соціальній інженерії.

Об’єктом дослідження є процес протидії атакам соціальної інженерії в інформаційних системах організацій.

Предметом дослідження є методи, технології та програмні засоби виявлення ознак соціальної інженерії, організації захисту користувачів і зниження ризиків, пов’язаних із людським фактором.

Ступінь новизни одержаних результатів полягає в подальшому розвитку підходу до навчально-демонстраційного аналізу повідомлень на ознаки соціальної інженерії. У роботі поєднано аналіз організаційних, технічних і програмних засобів протидії із практичною реалізацією консольного застосунку, який забезпечує авторизацію користувачів, журналювання подій, збереження результатів у базі Microsoft Access і визначення рівня ризику повідомлення за набором індикаторів соціально-інженерної атаки.

У роботі використано методи аналізу джерел, класифікації, порівняльного аналізу, системного підходу, алгоритмізації, програмування та тестування.

Практичне значення роботи полягає у розробленні консольного застосунку, який може використовуватися як навчальний або демонстраційний інструмент для аналізу підозрілих повідомлень, перевірки рівня ризику, збереження результатів у базі даних і демонстрації базових механізмів авторизації та безпеки. Вибір консольного застосунку на Python із використанням Microsoft Access є доцільним для навчальної реалізації, оскільки не потребує складної серверної інфраструктури та дозволяє наочно продемонструвати базові механізми захисту.

Галузь застосування результатів роботи – навчальні та демонстраційні процеси у сфері інформаційної безпеки й кібербезпеки, а також підготовка користувачів до розпізнавання типових ознак соціально-інженерних атак.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, переліку використаних джерел і додатків. У першому розділі обґрунтовується необхідність застосування технологій протидії соціальній інженерії. У другому розділі досліджуються сучасні організаційні, технічні та програмні засоби протидії. У третьому розділі розглядаються підходи до створення системи протидії соціальній інженерії, описується програмна реалізація та формуються практичні рекомендації.

1. ОБҐРУНТУВАННЯ НЕОБХІДНОСТІ ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ ПРОТИДІЇ СОЦІАЛЬНІЙ ІНЖЕНЕРІЇ

1.1 Аналіз сучасного стану питання реалізації атак соціальної інженерії

Соціальна інженерія є одним із поширених напрямів реалізації кіберзагроз, оскільки вона спрямована не лише на технічні вразливості інформаційної системи, а передусім на людину як учасника процесу обробки, передавання та захисту інформації. У межах такого підходу зловмисник використовує психологічні прийоми, довіру, необізнаність, поспіх або страх користувача для отримання конфіденційних даних, доступу до облікових записів чи спонукання до виконання небезпечних дій [4; 7; 10].

На відміну від класичних технічних атак, соціальна інженерія часто не потребує складної експлуатації програмних помилок. Для успішної атаки достатньо переконати користувача самостійно надати пароль, перейти за шкідливим посиланням, відкрити вкладення, повідомити службову інформацію або виконати інструкцію, що надходить від нібито довіреної особи. Саме тому людський фактор залишається однією з найбільш складних для усунення вразливостей у системі інформаційної безпеки [7; 10].

Психологічний вплив у соціальній інженерії часто ґрунтується на принципах переконання, які дозволяють зловмиснику впливати на рішення користувача. До таких принципів належать взаємність, зобов’язання та послідовність, соціальний доказ, авторитет, симпатія, а також дефіцит і терміновість. У соціально-інженерних атаках ці механізми можуть використовуватися для формування довіри, створення відчуття боргу, підпорядкування авторитету або примушення користувача до швидкого виконання дії без додаткової перевірки [9; 16]. Узагальнення основних психологічних прийомів соціальної інженерії наведено на рисунку 1.1.

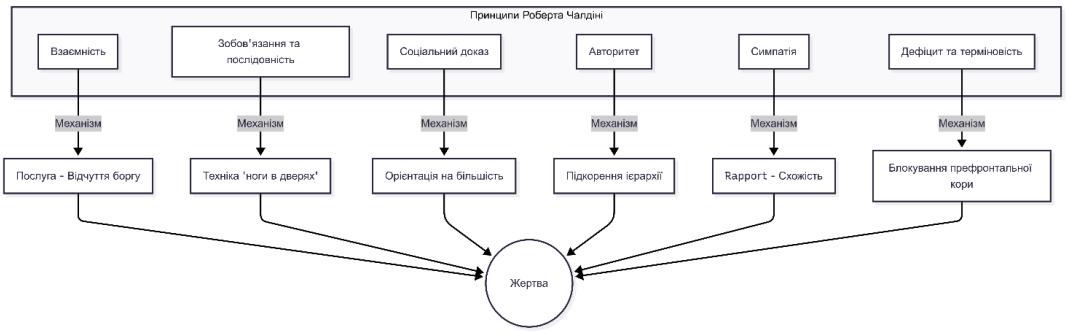

Рисунок 1.1 – Основні психологічні прийоми соціальної інженерії

Як видно з рисунка 1.1, психологічні прийоми соціальної інженерії спрямовані на те, щоб вплинути на поведінку жертви та знизити рівень критичного мислення. Наприклад, принцип авторитету може використовуватися через імітацію керівника, адміністратора або представника банку, а принцип дефіциту та терміновості – через повідомлення про обмежений час, блокування акаунта або необхідність негайного підтвердження даних. Саме тому протидія соціальній інженерії має враховувати не лише технічні, а й психологічні аспекти захисту користувачів.

Сучасні атаки соціальної інженерії реалізуються через різні канали комунікації. Найпоширенішим видом є фішинг, що передбачає надсилання електронних листів або повідомлень, які імітують офіційні звернення банків, державних установ, служб підтримки або відомих онлайн-сервісів. Метою таких повідомлень є отримання облікових даних, платіжної інформації або встановлення шкідливого програмного забезпечення [8; 19; 22].

Окремим різновидом фішингу є спір-фішинг, який спрямований на конкретну особу або організацію. На відміну від масових фішингових розсилок, спір-фішинг базується на попередньому збиранні інформації про жертву, її посаду, коло обов’язків, контакти та особливості професійної діяльності. Такий підхід підвищує переконливість повідомлення, оскільки воно виглядає персоналізованим і відповідає реальному контексту роботи користувача [6; 8].

До поширених форм соціальної інженерії також належать вішинг і смішинг. Вішинг реалізується через телефонні дзвінки або голосові повідомлення, під час

яких зловмисник може представлятися працівником банку, служби безпеки, технічної підтримки чи керівником. Смішинг здійснюється через SMS-повідомлення або месенджери та часто містить короткі посилання, повідомлення про блокування акаунта, виграш, штраф або необхідність термінового підтвердження даних [4; 6; 15].

Основні види атак соціальної інженерії доцільно класифікувати за каналом реалізації та способом впливу на користувача. Узагальнену класифікацію наведено на рисунку 1.2.

Рисунок 1.2 – Класифікація атак соціальної інженерії

Як видно з рисунка 1.2, атаки соціальної інженерії не обмежуються електронною поштою. Вони можуть реалізовуватися через голосові канали, SMS і месенджери, фізичну взаємодію, фальшиві сценарії спілкування та спеціально підготовлені психологічні прийоми.

Крім цифрових атак, соціальна інженерія може реалізовуватися через безпосередню взаємодію із жертвою. До таких методів належать претекстинг, бейтинг, квід про кво та тейлгейтинг. Претекстинг передбачає створення вигаданої ситуації або легенди для отримання інформації. Бейтинг використовує приманку, наприклад USB-носій або фальшивий файл. Квід про кво ґрунтується

на пропозиції певної вигоди або допомоги в обмін на інформацію. Тейлгейтинг пов’язаний із несанкціонованим фізичним доступом до приміщень шляхом проходу слідом за авторизованою особою [4; 7].

У сучасних умовах соціальна інженерія ускладнюється через активне використання відкритих джерел інформації. Соціальні мережі, професійні платформи, публічні реєстри, корпоративні сайти та витоки даних дозволяють зловмисникам збирати відомості про працівників, структуру організації, внутрішні процеси та комунікаційні зв’язки. На основі цих даних створюються більш переконливі сценарії атак, які складніше відрізнити від легітимних повідомлень [6; 13].

Окрему небезпеку становить використання технологій штучного інтелекту для підготовки соціально-інженерних атак. Автоматизована генерація текстів дозволяє створювати грамотно оформлені фішингові повідомлення різними мовами, адаптувати їх до конкретної аудиторії та швидко масштабувати кампанії. Крім того, технології синтезу голосу та відео можуть використовуватися для імітації керівників, співробітників або представників довірених організацій, що значно ускладнює перевірку особи співрозмовника [21].

Актуальність проблеми підтверджується тим, що значна частина інцидентів інформаційної безпеки пов’язана з діями користувачів, які стали жертвами маніпуляції або порушили встановлені правила безпечної роботи з інформацією. За даними аналітичних звітів у сфері кібербезпеки, соціальна інженерія і фішинг залишаються одними з основних векторів початкового проникнення до інформаційних систем організацій [20; 21; 22].

Таким чином, сучасний стан реалізації атак соціальної інженерії характеризується різноманітністю методів, активним використанням цифрових каналів комунікації, персоналізацією атак і поєднанням психологічного впливу з технічними засобами. Це обумовлює необхідність застосування комплексних технологій протидії, які мають включати не лише технічні механізми захисту, а й організаційні заходи, навчання персоналу та програмні інструменти для виявлення ознак потенційних атак.

1.2 Основні підходи та методи протидії соціальній інженерії

Протидія соціальній інженерії має базуватися на комплексному підході, оскільки атаки цього типу поєднують психологічний вплив, технічні засоби та використання організаційних недоліків. Неможливо повністю усунути ризик соціальної інженерії лише за рахунок антивірусного захисту, поштових фільтрів або навчання персоналу. Ефективна система захисту повинна одночасно охоплювати технічні, організаційні, навчальні та програмні заходи [3; 5; 13; 14].

Організаційний підхід передбачає створення чітких правил безпечної роботи з інформацією. До нього належать політика інформаційної безпеки, порядок обробки конфіденційних даних, правила перевірки підозрілих запитів, регламенти взаємодії з електронною поштою, месенджерами та зовнішніми контрагентами. Важливим елементом є розмежування прав доступу, за якого користувач отримує лише ті повноваження, які необхідні для виконання його службових обов’язків. Такий підхід дозволяє зменшити можливі наслідки атаки навіть у випадку, якщо один із користувачів став жертвою маніпуляції.

Одним із ключових організаційних методів протидії є встановлення процедур перевірки особи та підтвердження критичних дій. Наприклад, якщо користувач отримує повідомлення з проханням передати пароль, здійснити фінансову операцію, відкрити доступ до файлів або терміново виконати дію від імені керівника, така вимога має перевірятися через незалежний канал зв’язку. Це може бути телефонний дзвінок на офіційний номер, особисте підтвердження або звернення до адміністратора безпеки. Такий метод знижує ризик виконання дій на основі підроблених повідомлень або фальшивих звернень.

Навчання персоналу є одним із найбільш важливих напрямів протидії соціальній інженерії, оскільки саме користувач часто є основною ціллю атаки. Працівники повинні знати типові ознаки фішингових повідомлень, небезпечних посилань, підроблених сторінок входу, маніпулятивних формулювань і підозрілих запитів. Доцільним є не лише проведення теоретичних занять, а й використання практичних прикладів, тестування знань та моделювання типових ситуацій.

Регулярне навчання дозволяє сформувати у користувачів стійку навичку перевіряти повідомлення перед виконанням будь-яких дій [17; 26].

Технічний підхід до протидії соціальній інженерії включає використання засобів, які знижують імовірність успішного проведення атаки або обмежують її наслідки. До таких засобів належать фільтрація електронної пошти, перевірка вкладень, блокування шкідливих посилань, антивірусний захист, багатофакторна автентифікація, контроль доступу, журналювання подій та моніторинг підозрілої активності. Особливе значення має багатофакторна автентифікація, оскільки вона ускладнює використання викраденого пароля без додаткового підтвердження особи користувача [18].

Важливим технічним методом є захист електронної пошти, оскільки саме цей канал часто використовується для фішингових атак. Для цього застосовуються антиспам-фільтри, перевірка репутації відправника, аналіз вкладень, блокування небезпечних доменів, а також протоколи автентифікації поштових повідомлень, зокрема SPF, DKIM і DMARC. Їх використання дозволяє зменшити кількість підроблених листів, які доходять до користувачів, хоча повністю не усуває ризик персоналізованих атак [19; 22].

Програмні засоби протидії соціальній інженерії можуть виконувати допоміжну роль у виявленні потенційно небезпечних повідомлень. Такі засоби можуть аналізувати текст повідомлення, наявність підозрілих посилань, запити на введення пароля, термінові формулювання, погрози блокування акаунта, згадки про фінансові операції або інші індикатори ризику. На основі виявлених ознак програма може визначати рівень небезпеки повідомлення та надавати користувачу рекомендації щодо подальших дій [11; 12; 15].

Основні підходи до протидії соціальній інженерії доцільно подати у вигляді узагальненої схеми, що показує взаємозв’язок між організаційними, навчальними, технічними та програмними заходами. Така схема наведена на рисунку 1.3.

Рисунок 1.3 – Основні підходи до протидії соціальній інженерії

Як видно з рисунка 1.3, ефективний захист від соціальної інженерії не може обмежуватися лише одним видом заходів. Організаційні правила визначають порядок безпечної роботи, навчання формує обізнаність користувачів, технічні засоби зменшують кількість небезпечних повідомлень, а програмні інструменти допомагають виявляти ознаки ризику та фіксувати результати перевірок.

Окреме значення має журналювання подій безпеки. Записи про входи користувачів, невдалі спроби авторизації, блокування облікового запису, перевірку повідомлень і виявлені індикатори ризику дозволяють аналізувати поведінку користувачів та вчасно помічати підозрілу активність. У межах практичної частини роботи цей підхід буде враховано під час розроблення консольного застосунку, який передбачає авторизацію користувачів, збереження результатів аналізу повідомлень і ведення журналу подій.

Таким чином, основні методи протидії соціальній інженерії включають поєднання управлінських рішень, навчання персоналу, технічних засобів контролю і програмних інструментів аналізу. Найбільш ефективним є багаторівневий підхід, за якого користувач не залишається єдиною лінією

захисту, а його дії додатково контролюються політиками безпеки, засобами автентифікації, фільтрації, моніторингу та аналізу повідомлень.

1.3 Обґрунтування актуальності запровадження сучасних технологій протидії соціальній інженерії

Актуальність запровадження сучасних технологій протидії соціальній інженерії зумовлена тим, що цей вид атак залишається одним із найскладніших для повного усунення. На відміну від суто технічних загроз, соціальна інженерія використовує не лише недоліки програмного забезпечення, а й особливості поведінки людини. Навіть за наявності сучасних технічних засобів захисту користувач може самостійно надати зловмиснику облікові дані, перейти за небезпечним посиланням або виконати інструкцію з підробленого повідомлення [4; 7; 10].

Проблема посилюється тим, що сучасні атаки стають більш персоналізованими. Зловмисники активно використовують відкриті джерела інформації, соціальні мережі, професійні платформи та витоки даних для підготовки повідомлень, які виглядають правдоподібно. У таких умовах користувачу складніше відрізнити реальне службове повідомлення від шахрайського, особливо якщо атака містить елементи терміновості, посилання на авторитетну особу або загрозу негативних наслідків [13].

Додатковим фактором актуальності є розвиток технологій штучного інтелекту, які можуть використовуватися для автоматизованого створення переконливих текстів, підроблених голосових повідомлень або персоналізованих сценаріїв атак. Якщо раніше фішингові листи часто можна було розпізнати за грубими мовними помилками або неправдоподібним оформленням, то сьогодні такі повідомлення можуть бути грамотно написані, стилістично адаптовані до конкретної організації та схожі на реальну ділову комунікацію. Це знижує ефективність традиційних способів розпізнавання загроз лише за зовнішніми ознаками [21].

Важливим аспектом є також поширення дистанційної та змішаної форми роботи. У таких умовах значна частина комунікації між працівниками, керівниками, партнерами та клієнтами відбувається через електронну пошту, месенджери або інші цифрові канали. Це створює додаткові можливості для підміни особи, надсилання фальшивих запитів, поширення шкідливих посилань і маніпуляції користувачами. Відсутність особистого контакту ускладнює перевірку справжності звернення, тому роль процедур підтвердження та технічних засобів контролю суттєво зростає.

Особливого значення протидія соціальній інженерії набуває для державних установ, військових організацій, об’єктів критичної інфраструктури та підприємств, які працюють з конфіденційною інформацією. Для таких організацій наслідками успішної атаки можуть бути не лише фінансові втрати, а й витік службових даних, порушення роботи інформаційних систем, компрометація облікових записів або створення умов для подальших кібератак. Тому захист від соціальної інженерії має розглядатися як складова загальної системи інформаційної та кібербезпеки [1; 16].

Сучасні технології протидії соціальній інженерії мають поєднувати кілька рівнів захисту. Перший рівень становлять організаційні заходи: політики безпеки, правила перевірки запитів, порядок реагування на інциденти та розмежування прав доступу. Другий рівень охоплює навчання персоналу, спрямоване на формування навичок розпізнавання підозрілих повідомлень. Третій рівень включає технічні засоби: фільтрацію пошти, багатофакторну автентифікацію, блокування небезпечних посилань, моніторинг подій та журналювання дій користувачів. Четвертий рівень можуть становити програмні інструменти, які допомагають аналізувати повідомлення та визначати наявність ознак ризику [18; 19].

Запровадження програмних засобів аналізу підозрілих повідомлень є доцільним, оскільки такі інструменти можуть використовуватися як допоміжний елемент системи захисту. Вони не замінюють повноцінні засоби кібербезпеки, але дозволяють продемонструвати принципи виявлення індикаторів соціальної

інженерії: термінових формулювань, запитів на введення паролів, підозрілих посилань, фінансових вимог, імітації офіційного звернення або психологічного тиску. У межах цієї роботи такий підхід реалізується через консольний застосунок на Python з авторизацією користувачів, збереженням даних у базі Microsoft і фіксацією результатів перевірки повідомлень.

Отже, актуальність запровадження сучасних технологій протидії соціальній інженерії визначається високою поширеністю таких атак, посиленням ролі людського фактора, розвитком персоналізованих фішингових сценаріїв, використанням штучного інтелекту та зростанням залежності організацій від цифрових каналів комунікації. Ефективний захист потребує не одного окремого засобу, а комплексного поєднання організаційних, навчальних, технічних і програмних заходів.

Висновки до першого розділу

У першому розділі обґрунтовано необхідність застосування технологій протидії соціальній інженерії як одному з актуальних напрямів забезпечення інформаційної та кібербезпеки. Встановлено, що соціальна інженерія є небезпечною через орієнтацію не лише на технічні вразливості, а передусім на людський фактор, зокрема довіру, неуважність, страх, поспіх, необізнаність або помилки користувачів під час роботи з інформацією.

Проаналізовано сучасний стан реалізації атак соціальної інженерії. Визначено, що найбільш поширеними формами таких атак є фішинг, спір-фішинг, вішинг, смішинг, претекстинг, бейтинг, квід про кво та тейлгейтинг. Показано, що сучасні атаки можуть реалізовуватися через електронну пошту, телефонні дзвінки, SMS, месенджери, фальшиві вебсторінки, фізичні носії та безпосередню взаємодію із жертвою.

Розглянуто основні підходи та методи протидії соціальній інженерії. Встановлено, що ефективний захист має базуватися на поєднанні організаційних, навчальних, технічних і програмних заходів. До організаційних заходів належать

політики інформаційної безпеки, правила обробки конфіденційних даних, розмежування прав доступу та процедури перевірки критичних запитів. Навчальні заходи спрямовані на підвищення обізнаності користувачів і формування навичок розпізнавання підозрілих повідомлень. Технічні засоби включають фільтрацію пошти, багатофакторну автентифікацію, блокування небезпечних посилань, моніторинг подій і журналювання дій користувачів.

Обґрунтовано актуальність запровадження сучасних технологій протидії соціальній інженерії. Визначено, що розвиток цифрових комунікацій, дистанційної роботи, відкритих джерел інформації та засобів штучного інтелекту підвищує складність виявлення соціально-інженерних атак. У таких умовах особливого значення набувають програмні засоби, які можуть допомагати користувачам аналізувати підозрілі повідомлення, визначати ознаки ризику, фіксувати результати перевірок і підтримувати базові механізми авторизації та безпеки.

Отже, соціальна інженерія є комплексною загрозою, що потребує багаторівневої системи протидії. Результати першого розділу підтверджують необхідність подальшого дослідження організаційних, технічних і програмних засобів захисту, а також доцільність розроблення консольного програмного засобу для аналізу повідомлень на ознаки соціально-інженерних атак.

2 ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ СУЧАСНИХ ТЕХНОЛОГІЙ І МЕТОДІВ ДЛЯ ПРОТИДІЇ СОЦІАЛЬНІЙ ІНЖЕНЕРІЇ

2.1 Аналіз організаційних технологій протидії

Організаційні технології протидії соціальній інженерії спрямовані на створення таких правил, процедур і умов роботи, за яких імовірність успішного маніпулювання користувачем суттєво знижується. На відміну від суто технічних засобів захисту, організаційні заходи регулюють поведінку персоналу, порядок доступу до інформації, правила перевірки запитів і дії працівників у разі виявлення підозрілої активності. Саме тому вони є важливою складовою комплексної системи захисту від соціальної інженерії [3; 5; 13; 14].

Одним із базових організаційних заходів є розроблення та впровадження політики інформаційної безпеки. Такий документ визначає загальні правила роботи з інформацією, порядок використання облікових записів, вимоги до паролів, правила обробки конфіденційних даних, порядок взаємодії з електронною поштою та зовнішніми інформаційними ресурсами. Наявність політики інформаційної безпеки дозволяє формалізувати вимоги до користувачів і зменшити кількість випадкових порушень, які можуть бути використані зловмисниками.

Важливим напрямом є розмежування прав доступу. Користувачі повинні мати доступ лише до тих інформаційних ресурсів, які необхідні їм для виконання службових обов’язків. Такий принцип дозволяє зменшити можливі наслідки атаки: навіть якщо обліковий запис окремого користувача буде скомпрометований, зловмисник не отримає доступу до всієї інформаційної системи. Особливо важливо контролювати привілейовані облікові записи адміністраторів, керівників і працівників, які мають доступ до фінансової, службової або персональної інформації.

Окреме значення мають процедури перевірки критичних запитів. Соціально-інженерні атаки часто побудовані на створенні термінової ситуації,

коли користувача змушують швидко виконати певну дію: передати пароль, перейти за посиланням, відкрити доступ до файлу, здійснити платіж або встановити програму. Щоб зменшити ризик таких дій, організація повинна встановити правило обов’язкового підтвердження критичних запитів через незалежний канал зв’язку. Наприклад, запит, отриманий електронною поштою, має бути підтверджений телефоном або через офіційний внутрішній канал.

Навчання персоналу є однією з найбільш ефективних організаційних технологій протидії соціальній інженерії. Працівники повинні розуміти, як виглядають типові фішингові повідомлення, які ознаки можуть свідчити про підробку, чому не можна передавати паролі третім особам, як перевіряти посилання та що робити у разі отримання підозрілого повідомлення. Навчання має бути не одноразовим, а регулярним, оскільки методи атак постійно змінюються [17; 26].

Доцільним є використання практичних форм навчання: тестування знань, розбір прикладів фішингових повідомлень, короткі інструктажі, моделювання типових ситуацій і симульовані атаки. Такі методи дозволяють не лише передати теоретичну інформацію, а й сформувати у користувачів практичну навичку перевіряти повідомлення перед виконанням дій. Особливо важливо навчати працівників не боятися повідомляти про підозрілі листи або власні помилки, оскільки приховування інциденту може призвести до значно більших наслідків.

Основні організаційні заходи протидії соціальній інженерії наведено в таблиці 2.1.

Таблиця 2.1

Організаційні заходи протидії соціальній інженерії

Організаційний захід	Призначення	Очікуваний результат
Політика інформаційної безпеки	Визначає правила роботи з інформацією, обліковими записами та конфіденційними даними	Зменшення кількості порушень правил безпеки

Продовження таблиці 2.1

Організаційний захід	Призначення	Очікуваний результат
Розмежування прав доступу	Обмежує доступ користувачів лише необхідними ресурсами	Зменшення наслідків можливої компрометації
Перевірка критичних запитів	Передбачає підтвердження підозрілих або термінових запитів через незалежний канал	Зниження ризику виконання фальшивих інструкцій
Навчання персоналу	Формує навички розпізнавання фішингу та інших атак соціальної інженерії	Підвищення обізнаності користувачів
Моделювання атак	Дозволяє перевірити готовність персоналу до реальних сценаріїв атак	Виявлення слабких місць у поведінці користувачів
Журналювання подій	Фіксує спроби входу, підозрілі дії та інциденти безпеки	Можливість подальшого аналізу та реагування
Порядок реагування на інциденти	Визначає дії користувачів і адміністраторів у разі атаки	Скорочення часу реагування та локалізації загрози

Як видно з таблиці 2.1, організаційні технології протидії соціальній інженерії мають не лише попереджувальний, а й контрольний характер. Вони дозволяють встановити правила безпечної поведінки, зменшити вплив людського фактора, своєчасно виявляти підозрілі дії та швидше реагувати на інциденти.

Важливим елементом організаційної протидії є формування культури інформаційної безпеки. Працівники мають сприймати правила безпеки не як формальну вимогу, а як частину щоденної роботи. Для цього необхідно уникати підходу, за якого користувачів карають за кожну помилку. Набагато ефективнішим є підхід, за якого працівники можуть швидко повідомити про підозріле повідомлення, випадковий перехід за посиланням або іншу потенційно небезпечну ситуацію без страху приховувати інцидент.

Організаційні технології також повинні враховувати специфіку різних категорій працівників. Наприклад, співробітники фінансових підрозділів частіше можуть бути ціллю атак, пов’язаних із фальшивими платежами або підробленими дорученнями керівництва. Працівники кадрових підрозділів можуть отримувати

шкідливі вкладення під виглядом резюме. Адміністратори інформаційних систем є привабливою ціллю через наявність розширених прав доступу. Тому навчання і правила безпеки мають бути адаптовані до посадових обов'язків і рівня ризику.

Таким чином, організаційні технології протидії соціальній інженерії є необхідною основою для побудови ефективної системи захисту. Вони забезпечують регламентацію дій користувачів, формування навичок безпечної поведінки, контроль доступу, перевірку критичних запитів і реагування на інциденти. Однак самих організаційних заходів недостатньо, тому вони мають застосовуватися разом із технічними засобами та програмними рішеннями, які розглядаються в наступному підрозділі.

2.2 Дослідження технічних засобів і програмних рішень

Технічні засоби протидії соціальній інженерії спрямовані на зменшення ймовірності контакту користувача з небезпечним повідомленням, обмеження наслідків можливої помилки та виявлення підозрілої активності в інформаційній системі. На відміну від організаційних заходів, які залежать від поведінки персоналу, технічні засоби можуть автоматично блокувати частину загроз, контролювати доступ, перевіряти повідомлення, фіксувати події та попереджати адміністратора про небезпечні дії [5; 18; 19].

Одним із найпоширеніших технічних напрямів є захист електронної пошти. Оскільки фішингові атаки найчастіше реалізуються через електронні листи, важливе значення мають антиспам-фільтри, системи перевірки вкладень, фільтрація посилань і аналіз репутації відправника. Такі засоби дозволяють виявляти масові фішингові кампанії, блокувати небезпечні домени, перевіряти файли на наявність шкідливого програмного забезпечення та зменшувати кількість підозрілих повідомлень, які потрапляють до користувачів.

Важливу роль у захисті електронної пошти відіграють протоколи SPF, DKIM і DMARC. SPF дозволяє перевірити, чи має сервер-відправник право надсилати листи від імені певного домену. DKIM забезпечує криптографічний

підпис повідомлення, що дає змогу перевірити його цілісність. DMARC поєднує механізми SPF і DKIM та визначає політику обробки повідомлень, які не пройшли перевірку. Використання цих протоколів не усуває фішинг повністю, але суттєво ускладнює підробку доменів і масове розсилання листів від імені легітимних організацій.

Окремим напрямом є багатофакторна автентифікація. Вона знижує ризик несанкціонованого доступу навіть у випадку, якщо пароль користувача був викрадений за допомогою фішингу або іншого методу соціальної інженерії. Багатофакторна автентифікація передбачає використання додаткового фактора підтвердження особи: SMS-коду, одноразового коду в застосунку, push-підтвердження, апаратного ключа або біометричної перевірки. Найбільш стійкими до фішингу вважаються апаратні ключі та сучасні механізми автентифікації, які прив’язують підтвердження до конкретного домену або пристрою.

До технічних засобів також належать антивірусні системи, EDR-рішення, системи контролю доступу, DLP-системи, засоби журналювання та моніторингу подій. Антивірусний захист і EDR допомагають виявляти шкідливі файли, підозрілу активність процесів і спроби закріплення зловмисника в системі. DLP-системи спрямовані на запобігання витоку конфіденційної інформації. Системи моніторингу дозволяють виявляти нетипову поведінку користувачів, наприклад вхід у незвичний час, багаторазові невдалі спроби авторизації або доступ до нетипових ресурсів.

Основні технічні засоби захисту від соціально-інженерних атак наведено в таблиці 2.2.

Таблиця 2.2

Технічні засоби захисту від соціально-інженерних атак

Технічний засіб	Призначення	Обмеження
Фільтрація електронної пошти	Блокування фішингових листів, спаму, небезпечних вкладень і посилань	Не завжди виявляє персоналізовані атаки

Продовження таблиці 2.2

Технічний засіб	Призначення	Обмеження
SPF / DKIM / DMARC	Перевірка автентичності домену відправника та зменшення підробки поштових адрес	Потребує правильного налаштування домену
Багатофакторна автентифікація	Захист облікового запису навіть після викрадення пароля	Може бути незручною для користувачів; окремі методи вразливі до маніпуляцій
Антивірусний захист	Виявлення шкідливих файлів і програм	Не запобігає психологічному впливу на користувача
EDR-рішення	Контроль активності кінцевих пристроїв і виявлення підозрілих процесів	Потребує налаштування та аналізу подій
DLP-система	Запобігання витоку конфіденційної інформації	Не завжди визначає контекст передачі даних
Моніторинг і журналювання	Фіксація входів, помилок, підозрілих дій і подій безпеки	Потребує регулярного перегляду журналів

Як видно з таблиці 2.2, кожен технічний засіб має власне призначення і власні обмеження. Тому їх доцільно використовувати не окремо, а як частину багаторівневої системи захисту. Наприклад, фільтрація пошти може зменшити кількість фішингових листів, багатофакторна автентифікація обмежує наслідки викрадення пароля, а журналювання дозволяє виявити підозрілу активність після спроби атаки.

Серед технічних засобів особливе місце займає багатофакторна автентифікація, оскільки вона безпосередньо пов’язана із захистом облікових записів користувачів. У випадку соціальної інженерії зловмисник часто намагається отримати логін і пароль жертви. Якщо додатковий фактор автентифікації відсутній, отриманих даних може бути достатньо для входу в систему. Якщо ж багатофакторна автентифікація налаштована правильно, сам пароль уже не забезпечує повного доступу [18].

Водночас різні методи багатофакторної автентифікації мають різний рівень стійкості. SMS-коди є простими у використанні, але можуть бути вразливими до

атак на мобільного оператора або перехоплення повідомлень. Одноразові коди в застосунках є надійнішими, але все ще можуть бути викрадені через фальшиву сторінку входу. Push-підтвердження зручне, але може використовуватися зловмисниками у схемах багаторазового надсилання запитів, коли користувач підтверджує вхід через втому або неуважність. Найбільш захищеними є апаратні ключі, однак вони потребують додаткових витрат і організаційного впровадження.

Порівняння основних методів багатофакторної автентифікації наведено в таблиці 2.3.

Таблиця 2.3

Порівняння методів багатофакторної автентифікації

Метод автентифікації	Рівень захисту	Основні переваги	Недоліки
SMS-код	Середній	Простота використання, не потребує окремого застосунку	Ризик перехоплення, SIM-swap, залежність від мобільного зв’язку
TOTP-додаток	Високий	Генерує одноразові коди без SMS, працює офлайн	Потребує встановлення застосунку, можливий фішинг у реальному часі
Push-підтвердження	Високий	Зручність для користувача, швидке підтвердження входу	Можливі атаки через втому від багаторазових запитів
Апаратний ключ	Дуже високий	Висока стійкість до фішингу, прив’язка до пристрою	Вартість, потреба у фізичному носії
Біометрична перевірка	Високий	Зручність, складність підробки для звичайного зловмисника	Залежність від пристрою, питання зберігання біометричних даних

Таблиця 2.3 показує, що впровадження багатофакторної автентифікації саме по собі ще не гарантує однакового рівня захисту. Для організацій, які працюють з конфіденційною або службовою інформацією, доцільно використовувати більш

стійкі методи підтвердження особи, особливо для адміністраторів, керівників і користувачів із розширеними правами доступу.

Програмні рішення у сфері протидії соціальній інженерії можуть виконувати як захисну, так і навчальну функцію. До них належать системи аналізу електронної пошти, засоби перевірки URL-адрес, фільтри вкладень, платформи моніторингу подій, інструменти тестування обізнаності персоналу та локальні програми для аналізу підозрілих повідомлень. Такі рішення допомагають автоматизувати частину перевірок і зменшити навантаження на користувача, який не завжди може самостійно оцінити небезпеку отриманого повідомлення.

У межах цієї роботи особливий інтерес становлять програмні засоби, які можуть аналізувати текст повідомлення за набором ознак ризику. До таких ознак належать наявність слів, що створюють терміновість, прохання надати пароль або персональні дані, згадки про блокування акаунта, підозрілі посилання, фінансові вимоги, імітація звернення від керівника або служби підтримки. Подібний підхід не замінює повноцінні корпоративні системи захисту, але може бути використаний як навчальний і допоміжний інструмент для демонстрації принципів виявлення соціально-інженерних атак.

Для практичної реалізації в цій роботі обрано консольний застосунок на Python із використанням бази даних Microsoft. Такий формат відповідає вимогам до програмної частини, є достатньо простим для розгортання на ПК і дозволяє реалізувати базові функції безпеки: авторизацію користувача, перевірку повідомлення, визначення рівня ризику, збереження результатів аналізу та журналювання подій. Такий застосунок може використовуватися як демонстраційний приклад програмного рішення, що підтримує процес протидії соціальній інженерії.

Отже, технічні засоби і програмні рішення є важливою складовою протидії соціальній інженерії. Вони не усувають людський фактор повністю, але дозволяють зменшити кількість небезпечних повідомлень, захистити облікові записи, контролювати дії користувачів, виявляти підозрілу активність і підтримувати процес аналізу потенційних атак. Найбільший ефект досягається

тоді, коли технічні засоби використовуються разом з організаційними заходами та навчанням персоналу.

2.3 Аналіз інструментів моделювання та тестування атак

Інструменти моделювання та тестування атак соціальної інженерії використовуються для перевірки готовності користувачів і організацій до реальних загроз. Їх основне призначення полягає не у завданні шкоди, а у створенні контрольованих навчальних ситуацій, які дозволяють оцінити рівень обізнаності персоналу, виявити слабкі місця в організаційних процедурах і визначити, наскільки ефективно працівники реагують на підозрілі повідомлення або запити.

Одним із найпоширеніших напрямів такого тестування є моделювання фішингових атак. У межах навчальної кампанії користувачам можуть надсилатися спеціально підготовлені повідомлення, які імітують реальні фішингові листи. При цьому система фіксує, чи відкрив користувач лист, чи перейшов за посиланням, чи ввів дані на фальшивій сторінці та чи повідомив про підозріле повідомлення відповідальним особам. Такі результати дозволяють оцінити не лише індивідуальний рівень підготовки працівників, а й загальний стан культури інформаційної безпеки в організації [17; 21; 23; 26].

Важливою перевагою моделювання атак є можливість навчання користувачів без реальних негативних наслідків. Якщо працівник помилився під час симуляції, він може одразу отримати пояснення, які саме ознаки вказували на фішинг: підозрілий домен, термінові формулювання, запит на пароль, нетиповий стиль повідомлення або невідповідність адреси відправника. Такий підхід є ефективнішим, ніж звичайна лекція, оскільки користувач отримує практичний досвід і краще запам’ятовує помилку.

Серед інструментів для моделювання атак соціальної інженерії можна виділити як відкриті, так і комерційні рішення. Відкриті інструменти зазвичай є безкоштовними або умовно безкоштовними, але потребують технічних навичок

для встановлення, налаштування і безпечного використання. Комерційні платформи мають ширший набір функцій, готові шаблони, аналітику, навчальні матеріали та підтримку, але можуть бути фінансово недоступними для невеликих організацій або установ з обмеженим бюджетом.

GoPhish є одним із відомих відкритих інструментів для проведення симульованих фішингових кампаній. Він дозволяє створювати шаблони листів, налаштовувати цільові групи користувачів, надсилати повідомлення та аналізувати результати кампанії. Перевагою GoPhish є простота використання і можливість локального розгортання, що важливо для організацій, які не можуть передавати дані до зовнішніх хмарних сервісів.

SET, або Social Engineering Toolkit, є більш складним інструментом, який використовується переважно для тестування на проникнення та моделювання різних сценаріїв соціальної інженерії. Він має широкий функціонал, але потребує обережного використання, оскільки орієнтований більше на фахівців з кібербезпеки, ніж на звичайні навчальні кампанії для персоналу. У навчальній або дипломній роботі такий інструмент доцільно розглядати саме як приклад засобу тестування, а не як основний варіант для впровадження в організації.

Комерційні платформи, такі як KnowBe4, Proofpoint Security Awareness Training і Microsoft Attack Simulator, орієнтовані на комплексне навчання персоналу. Вони можуть містити великі бібліотеки шаблонів фішингових листів, навчальні модулі, аналітику результатів, автоматичне призначення курсів користувачам і звітність для адміністраторів. Їх перевагою є зручність і масштабованість, однак суттєвим обмеженням є вартість, залежність від ліцензій і не завжди достатня адаптація до українського мовного та організаційного контексту.

Порівняння основних інструментів моделювання атак соціальної інженерії наведено в таблиці 2.4.

Таблиця 2.4

Порівняння інструментів моделювання атак соціальної інженерії

Інструмент	Призначення	Переваги	Обмеження
GoPhish	Проведення симульованих фішингових кампаній	Безкоштовний, має веб-інтерфейс, підтримує локальне розгортання	Потребує технічного налаштування, має обмежені навчальні матеріали
SET	Моделювання різних сценаріїв соціальної інженерії	Широкий функціонал, придатний для тестування на проникнення	Складний для звичайного навчального використання, потребує досвіду
King Phisher	Проведення фішингових кампаній і аналіз результатів	Дає змогу створювати кампанії та переглядати статистику	Обмежена підтримка і складніше розгортання
Microsoft Attack Simulator	Симуляція атак у середовищі Microsoft 365	Інтеграція з Microsoft 365, зручність для організацій, які вже використовують цю екосистему	Доступний не в усіх ліцензіях, залежить від хмарної інфраструктури
KnowBe4	Навчання персоналу та симуляція фішингових атак	Велика база шаблонів, навчальні модулі, розвинена аналітика	Висока вартість, залежність від комерційної платформи
Proofpoint Security Awareness Training	Навчання користувачів і тестування їхньої стійкості до атак	Якісна аналітика, інтеграція із захисними рішеннями Proofpoint	Висока вартість, найкраще працює в межах власної екосистеми

Як видно з таблиці 2.4, інструменти моделювання атак мають різне призначення та рівень складності. Відкриті рішення, такі як GoPhish або SET, надають більше гнучкості та можуть бути розгорнуті локально, але потребують вищого рівня технічної підготовки. Комерційні платформи є зручнішими для масштабного навчання персоналу, однак їх впровадження потребує фінансових витрат і залежності від зовнішнього постачальника.

Для організацій, які працюють з конфіденційною інформацією або мають обмеження щодо використання хмарних сервісів, важливим критерієм є можливість локального розгортання. У такому випадку відкриті інструменти

можуть бути більш придатними, оскільки дозволяють контролювати інфраструктуру, дані користувачів і результати тестування. Водночас їх застосування потребує чітких правил, щоб симульовані атаки не порушували права працівників і не створювали зайвого психологічного тиску.

Моделювання атак соціальної інженерії має проводитися відповідально. Перед запуском навчальних кампаній організація повинна визначити мету тестування, категорії користувачів, типи повідомлень, порядок обробки результатів і правила використання отриманих даних. Основною метою таких заходів має бути підвищення обізнаності персоналу, а не покарання окремих працівників. Якщо користувачі сприймають симуляції як спробу “спіймати” їх на помилці, це може знизити довіру до заходів безпеки і погіршити готовність повідомляти про реальні інциденти.

Результати моделювання атак можуть бути корисними для подальшого вдосконалення системи захисту. Наприклад, якщо велика кількість користувачів переходить за посиланням у повідомленнях із терміновими вимогами, це свідчить про необхідність додаткового навчання щодо психологічного тиску. Якщо користувачі не повідомляють про підозрілі листи, необхідно спростити процедуру звітування або пояснити її важливість. Якщо працівники часто реагують на повідомлення від імені керівництва, доцільно посилити процедури підтвердження критичних запитів.

У межах цієї роботи аналіз інструментів моделювання атак є важливим для визначення підходів до практичної реалізації програмного засобу. Оскільки розроблюваний застосунок є консольним і призначений для демонстрації базових принципів протидії соціальній інженерії, він не дублює функції великих платформ симуляції атак. Його завдання полягає в іншому: показати, як можна реалізувати авторизацію користувачів, журналювання подій і аналіз повідомлень за типовими індикаторами ризику.

Отже, інструменти моделювання та тестування атак соціальної інженерії є важливим елементом комплексної системи протидії. Вони дозволяють оцінити рівень підготовки персоналу, виявити слабкі місця в організаційних процедурах,

перевірити ефективність навчання і сформувати практичні навички розпізнавання загроз. Найбільший ефект від таких інструментів досягається тоді, коли вони застосовуються разом з організаційними заходами, технічними засобами захисту та регулярним навчанням користувачів.

Висновки до другого розділу

У другому розділі досліджено можливості сучасних технологій і методів, що застосовуються для протидії соціальній інженерії. Розглянуто організаційні технології, технічні засоби, програмні рішення та інструменти моделювання атак, які у сукупності формують основу комплексного захисту користувачів і інформаційних систем.

Проаналізовано організаційні технології протидії соціальній інженерії. Встановлено, що важливу роль відіграють політика інформаційної безпеки, розмежування прав доступу, перевірка критичних запитів, навчання персоналу, журналювання подій і порядок реагування на інциденти. Такі заходи дозволяють зменшити ризик виконання користувачами небезпечних дій під впливом маніпуляції та формують основу безпечної поведінки в організації.

Досліджено технічні засоби захисту від соціально-інженерних атак. Визначено, що важливими елементами технічного захисту є фільтрація електронної пошти, перевірка вкладень і посилань, протоколи SPF, DKIM і DMARC, багатофакторна автентифікація, антивірусний захист, EDR-рішення, DLP-системи, моніторинг і журналювання подій. Зазначені засоби не усувають людський фактор повністю, однак зменшують імовірність успішної атаки та обмежують її можливі наслідки.

Окрему увагу приділено багатофакторній автентифікації як одному з ключових засобів захисту облікових записів. Встановлено, що різні методи автентифікації мають різний рівень стійкості до атак. SMS-коди є простими у використанні, але менш захищеними, тоді як TOTP-додатки, push-підтвердження, біометрична перевірка та апаратні ключі забезпечують вищий рівень безпеки.

Найбільш доцільним є використання стійких методів автентифікації для облікових записів із підвищеними правами доступу.

Проаналізовано інструменти моделювання та тестування атак соціальної інженерії. Розглянуто відкриті та комерційні рішення, зокрема GoPhish, SET, King Phisher, Microsoft Attack Simulator, KnowBe4 і Proofpoint Security Awareness Training. Встановлено, що відкриті інструменти забезпечують гнучкість і можливість локального розгортання, але потребують технічної підготовки. Комерційні платформи є зручнішими для масштабного навчання персоналу, однак мають вищу вартість і можуть залежати від зовнішньої інфраструктури.

Результати дослідження показали, що жоден окремий метод не забезпечує повного захисту від соціальної інженерії. Найбільш ефективним є поєднання організаційних заходів, технічних засобів, навчання персоналу, моделювання атак і програмних інструментів аналізу підозрілих повідомлень. Саме такий підхід дозволяє зменшити вплив людського фактора, своєчасно виявляти ознаки небезпеки та підвищувати загальну стійкість організації до соціально-інженерних атак.

Отже, проведене дослідження підтверджує доцільність створення програмного засобу, який демонструє базові принципи протидії соціальній інженерії: авторизацію користувачів, перевірку повідомлень на наявність індикаторів ризику, визначення рівня небезпеки, збереження результатів у базі даних і журналювання подій. Це є основою для переходу до третього розділу, присвяченого застосуванню технологій протидії соціальній інженерії та опису програмної реалізації.

3. ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ ПРОТИДІЇ СОЦІАЛЬНІЙ ІНЖЕНЕРІЇ

3.1 Підходи до створення системи протидії соціальній інженерії

Застосування технологій протидії соціальній інженерії передбачає не лише аналіз теоретичних підходів, а й практичну реалізацію окремих механізмів захисту. У межах роботи було розроблено консольний програмний засіб, призначений для демонстрації базових підходів до виявлення ознак соціально-інженерних атак у текстових повідомленнях, організації авторизації користувачів, збереження результатів перевірок і журналювання подій безпеки.

Розроблений застосунок має навчально-демонстраційне призначення. Він не виконує реальну фільтрацію електронної пошти, не надсилає повідомлення, не проводить фішингові кампанії та не збирає чужі облікові дані. Його основна мета полягає в демонстрації того, як за допомогою простого програмного засобу можна реалізувати перевірку повідомлень за типовими індикаторами соціальної інженерії та поєднати її з базовими механізмами безпеки користувацького доступу.

Програмний засіб реалізовано мовою Python у вигляді консольного застосунку для операційної системи Windows. Для зберігання даних використовується база Microsoft Access social_engineering_defense.accdb, а підключення до неї виконується через бібліотеку pyodbc. У базі зберігаються облікові записи користувачів, результати аналізу повідомлень, знайдені індикатори ризику, журнал входів і журнал подій безпеки .

Додатково програмний засіб має модульну структуру, що спрощує його супровід, тестування та подальше розширення. Основні функції застосунку розподілено між окремими файлами відповідно до їх призначення. Модуль запуску відповідає за початкове меню та перехід до реєстрації або авторизації. Модуль авторизації реалізує створення облікових записів, перевірку паролів, хешування, облік невдалих спроб входу та блокування користувачів. Модуль

роботи з базою даних забезпечує підключення до Microsoft Access і виконання запитів для збереження користувачів, повідомлень, індикаторів та журналів.

Окремий модуль аналізу повідомлень відповідає за пошук індикаторів соціальної інженерії, нарахування балів ризику, визначення рівня небезпеки та формування рекомендації користувачу. Меню звичайного користувача містить функції аналізу повідомлення, перегляду історії перевірок і зміни пароля. Меню адміністратора призначене для перегляду користувачів, журналів входів, подій безпеки, усіх перевірених повідомлень і розблокування облікових записів. Такий поділ дозволяє уникнути надмірного ускладнення одного програмного файла та забезпечує логічну організацію коду.

Загальна логіка роботи застосунку передбачає запуск програми, вибір дії у головному меню, реєстрацію або авторизацію користувача, перехід до меню відповідної ролі та виконання доступних функцій. Для звичайного користувача доступні аналіз повідомлення, перегляд власної історії перевірок і зміна пароля. Для адміністратора передбачено перегляд користувачів, журналу входів, подій безпеки, усіх перевірених повідомлень і розблокування облікових записів.

Загальний алгоритм роботи консольного застосунку наведено на рисунку 3.1.

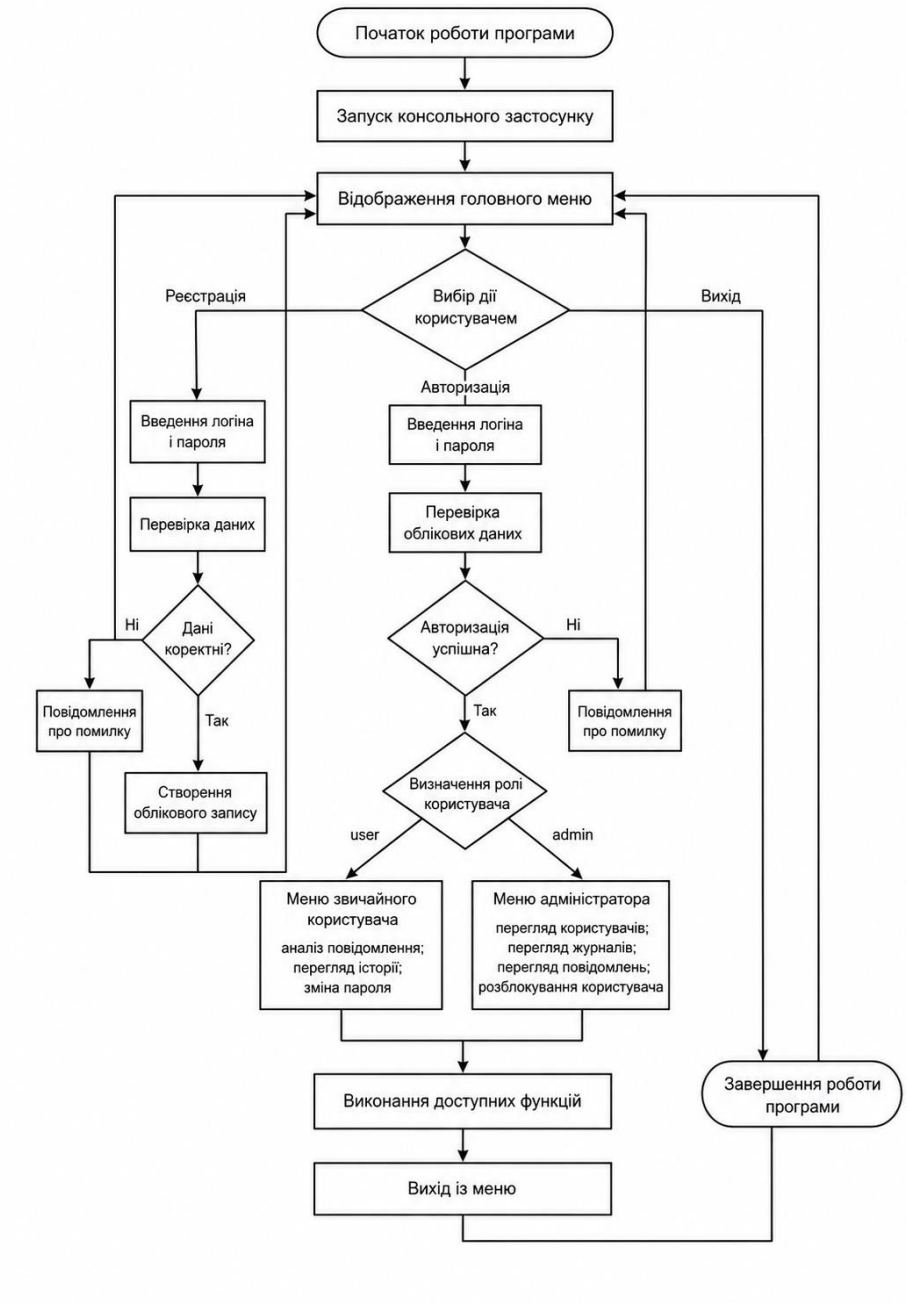

Рисунок 3.1 – Алгоритм роботи консольного застосунку

Як видно з рисунка 3.1, робота програми починається з головного меню, після чого користувач може зареєструватися, авторизуватися або завершити роботу. Після успішної авторизації система визначає роль користувача та відкриває відповідне меню. Такий підхід дозволяє розмежувати функції звичайного користувача й адміністратора.

Основні функціональні можливості розробленого програмного засобу наведено в таблиці 3.1.

Таблиця 3.1

Функціональні можливості консольного застосунку

Функція	Призначення	Реалізація
Реєстрація користувача	Створення нового облікового запису	Введення логіна, пароля та підтвердження пароля
Авторизація користувача	Контроль доступу до функцій програми	Перевірка логіна, пароля та стану блокування
Перевірка складності пароля	Підвищення базового рівня безпеки	Контроль довжини, літер, цифр і спеціальних символів
Хешування пароля	Захист пароля від зберігання у відкритому вигляді	Використання солі та алгоритму `pbkdf2_hmac`
Блокування користувача	Захист від багаторазового підбору пароля	Блокування після 3 невдалих спроб входу
Розподіл ролей	Обмеження доступу до адміністративних функцій	Ролі `admin` і `user`
Аналіз повідомлення	Виявлення ознак соціальної інженерії	Перевірка тексту за групами індикаторів
Розрахунок ризику	Визначення рівня небезпеки повідомлення	Нарахування балів і визначення рівня ризику
Журналювання подій	Фіксація важливих дій користувачів	Запис входів і подій безпеки до бази даних
Перегляд історії	Контроль результатів попередніх перевірок	Виведення збережених повідомлень і рівнів ризику

Розмежування ролей у програмі має важливе значення для демонстрації принципу мінімальних привілеїв. Звичайний користувач отримує доступ лише до тих функцій, які потрібні для перевірки повідомлень і перегляду власних результатів. Адміністратор, навпаки, має доступ до службових функцій, пов’язаних із контролем роботи системи, переглядом журналів і керуванням станом облікових записів. Такий підхід дозволяє показати, що навіть у навчальному програмному засобі доступ до функцій має бути обмежений відповідно до ролі користувача.

Практичне значення такого розмежування полягає в тому, що користувач не може переглядати службову інформацію, журнали інших користувачів або виконувати адміністративні дії. Це зменшує ризик випадкового або навмисного

втручання в роботу системи. У контексті протидії соціальній інженерії така модель є важливою, оскільки навіть у разі компрометації звичайного облікового запису зловмисник не отримує доступу до адміністративних можливостей програми.

У програмі реалізовано дві ролі користувачів: user і admin. Роль user призначена для звичайного користувача, який може аналізувати повідомлення, переглядати власну історію перевірок і змінювати пароль. Роль admin призначена для адміністратора, який має доступ до службових функцій: перегляду користувачів, журналів, усіх перевірених повідомлень і розблокування облікових записів.

Одним із важливих елементів реалізації є механізм авторизації. Під час входу користувач вводить логін і пароль. Програма перевіряє наявність користувача в базі даних, стан блокування облікового запису та правильність пароля. Пароль не зберігається у відкритому вигляді: для кожного користувача створюється окрема сіль, після чого пароль хешується з використанням алгоритму hashlib.pbkdf2_hmac із SHA-256 .

Алгоритм авторизації користувача наведено на рисунку 3.2.

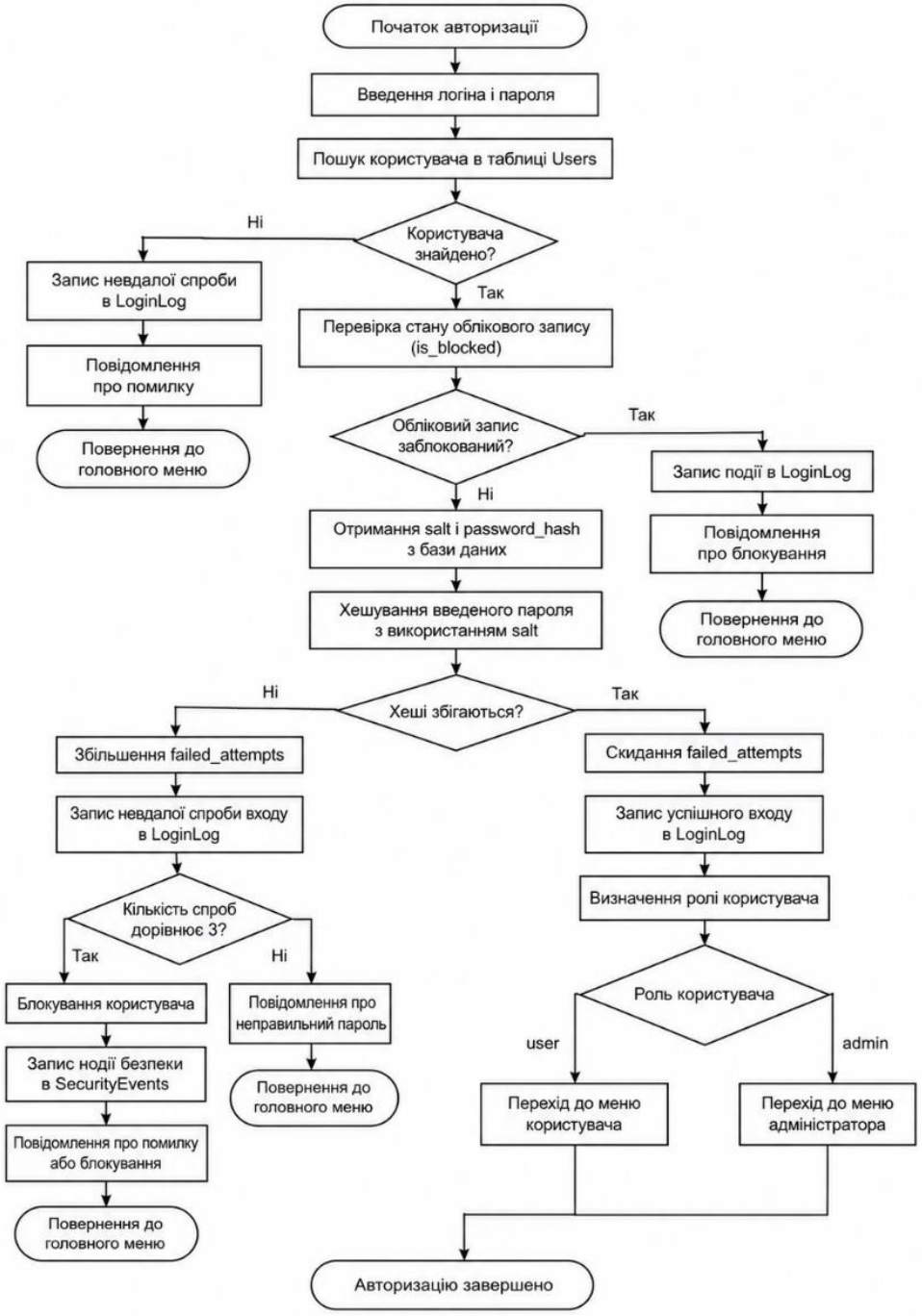

Рисунок 3.2 – Алгоритм авторизації користувача

Як показано на рисунку 3.2, після введення логіна і пароля програма перевіряє, чи існує відповідний користувач, чи не заблокований його обліковий запис, а також чи збігається хеш введеного пароля зі збереженим значенням. У разі успішної авторизації користувач переходить до меню відповідної ролі. У разі помилки збільшується лічильник невдалих спроб входу, а після трьох помилок обліковий запис блокується.

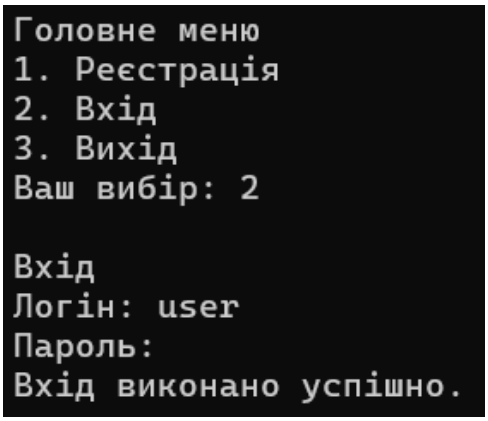

Рисунок 3.3 – Приклад успішної авторизації користувача

На рисунку 3.3 наведено приклад успішної авторизації користувача та переходу до меню відповідної ролі. Цей скріншот підтверджує працездатність механізму входу та розмежування функцій залежно від ролі користувача.

Особливістю реалізованого механізму авторизації є те, що пароль користувача не зберігається у відкритому вигляді. Під час реєстрації для кожного облікового запису формується окрема сіль, яка використовується під час створення хешу пароля. Під час повторного входу програма не порівнює введений пароль напряму, а повторно обчислює хеш із використанням збереженої солі та порівнює отримане значення із записом у базі даних. Такий підхід зменшує ризик розкриття паролів у разі несанкціонованого доступу до бази.

Додатковим елементом захисту є обмеження кількості невдалих спроб входу. Якщо користувач кілька разів вводить неправильний пароль, система збільшує лічильник помилок і після досягнення встановленого порогу блокує обліковий запис. Інформація про невдалі спроби входу фіксується в журналі, що дозволяє адміністратору виявляти підозрілу активність і аналізувати можливі спроби підбору пароля.

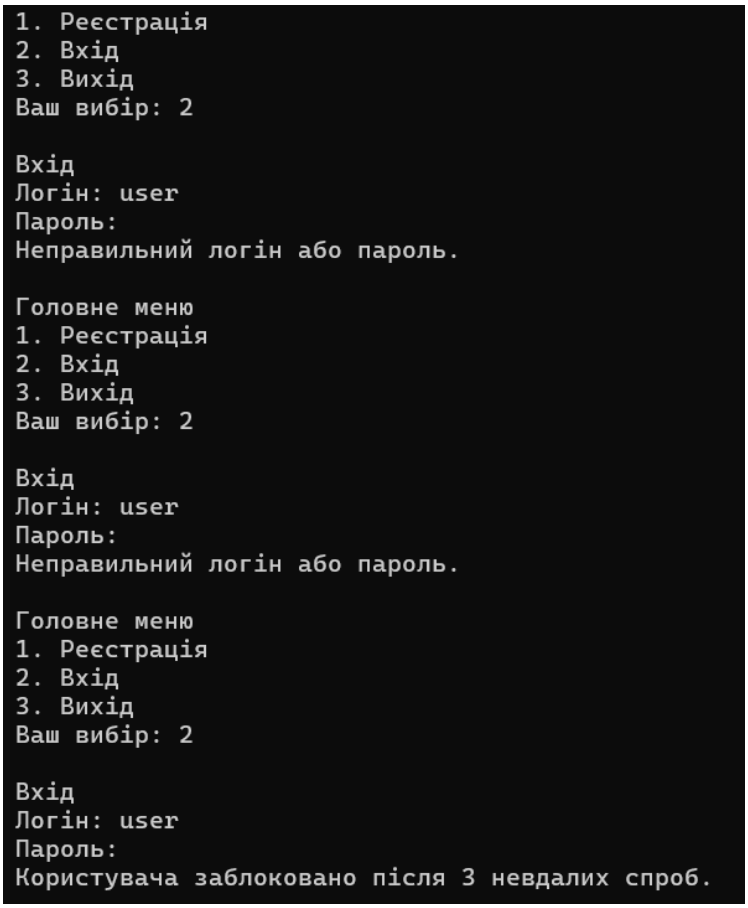

Рисунок 3.4 – Блокування користувача після невдалих спроб входу

Рисунок 3.4 демонструє спрацювання механізму блокування облікового запису після кількох неправильних спроб введення пароля. Такий механізм є базовим засобом захисту від підбору пароля та несанкціонованого доступу до функцій програми.

Для зберігання даних використовується база Microsoft Access. Її структура включає таблиці Users, Messages, Indicators, LoginLog і SecurityEvents. Таблиця Users містить інформацію про користувачів, ролі, хеші паролів, солі, кількість невдалих спроб входу та стан блокування. Таблиця Messages зберігає перевірені повідомлення, рівень ризику та кількість балів. Таблиця Indicators містить знайдені ознаки соціальної інженерії. Таблиці LoginLog і SecurityEvents використовуються для журналювання спроб входу та важливих подій безпеки.

Структуру бази даних програмного засобу наведено в таблиці 3.2.

Таблиця 3.2

Структура бази даних програмного засобу

Таблиця	Призначення	Основні поля
Users	Зберігання облікових записів користувачів	id, username, password_hash, salt, role, failed_attempts, is_blocked, created_at
Messages	Зберігання перевірених повідомлень	id, user_id, message_text, risk_score, risk_level, created_at
Indicators	Зберігання знайдених індикаторів ризику	id, message_id, indicator_type, description, score
LoginLog	Зберігання спроб входу	id, user_id, username_attempt, login_time, status, note
SecurityEvents	Зберігання подій безпеки	id, user_id, event_type, description, created_at

Зв’язки між таблицями дозволяють зберігати результати перевірок у прив’язці до конкретного користувача. Кожне повідомлення пов’язане з користувачем, який виконав аналіз, а знайдені індикатори пов’язані з конкретним повідомленням. Це дає змогу переглядати історію перевірок, аналізувати результати та фіксувати події безпеки.

Структуру бази даних у вигляді схеми наведено на рисунку 3.5.

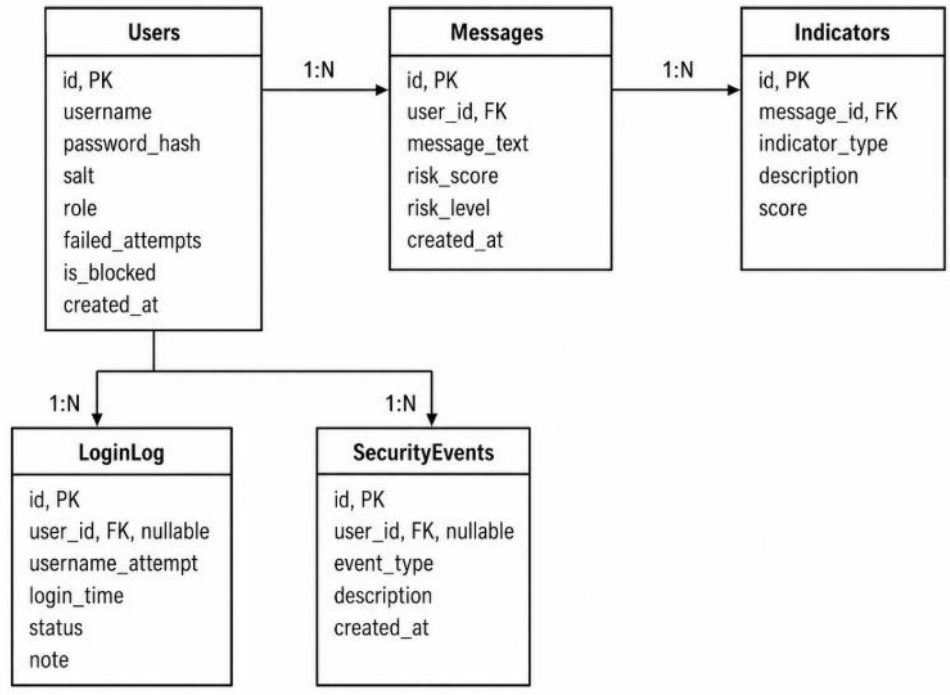

Рисунок 3.5 – Структура бази даних програмного засобу

Як видно з рисунка 3.5, центральною таблицею є Users, з якою пов’язані повідомлення, журнали входів і події безпеки. Таблиця Indicators пов’язана з таблицею Messages, оскільки для кожного перевіреного повідомлення може бути знайдено кілька індикаторів ризику.

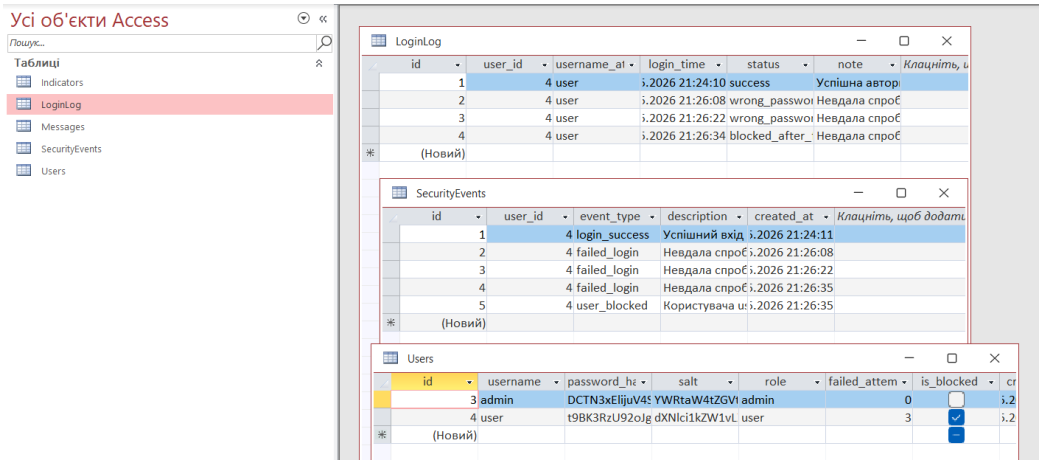

Рисунок 3.6 – Таблиці бази даних Microsoft Access

На рисунку 3.6 показано відкриту базу даних Microsoft Access із таблицями, що використовуються програмою. Наявність таблиць підтверджує реалізацію збереження користувачів, повідомлень, індикаторів ризику, журналу входів і подій безпеки.

Основна функція застосунку полягає в аналізі текстових повідомлень на ознаки соціальної інженерії. Аналіз має правило-орієнтований характер: програма не виконує семантичне розуміння тексту, а перевіряє повідомлення за заздалегідь визначеними групами індикаторів. До таких груп належать терміновість, запит конфіденційних даних, фінансові дії, імітація авторитету, психологічний тиск і підозрілі посилання.

Індикатори соціальної інженерії, що перевіряються програмою, наведено в таблиці 3.3.

Таблиця 3.3

Індикатори соціальної інженерії, що перевіряються програмою

Тип індикатора	Приклади	Кількість балів
Терміновість	терміново, негайно, сьогодні, протягом 24 годин, останній шанс	+2
Запит конфіденційних даних	пароль, логін, код підтвердження, CVV, номер картки	+3
Фінансова дія	переказ, оплата, рахунок, платіж, компенсація, виграш, штраф	+2
Імітація авторитету	служба безпеки, адміністратор, керівник, банк, технічна підтримка	+2
Психологічний тиск	буде заблоковано, ви втратите доступ, санкції, негайне підтвердження	+2
Підозріле посилання	bit.ly, tinyurl, cutt.ly, IP-адреса, login, verify, account, secure, update	+3

Після виявлення індикаторів програма нараховує бали ризику. Якщо повідомлення отримує від 0 до 2 балів, йому присвоюється низький рівень ризику. Якщо кількість балів становить від 3 до 6, ризик визначається як середній. Якщо повідомлення отримує 7 і більше балів, воно класифікується як повідомлення з високим рівнем ризику. Для кожного рівня програма формує рекомендацію користувачу.

Алгоритм аналізу повідомлення наведено на рисунку 3.7.

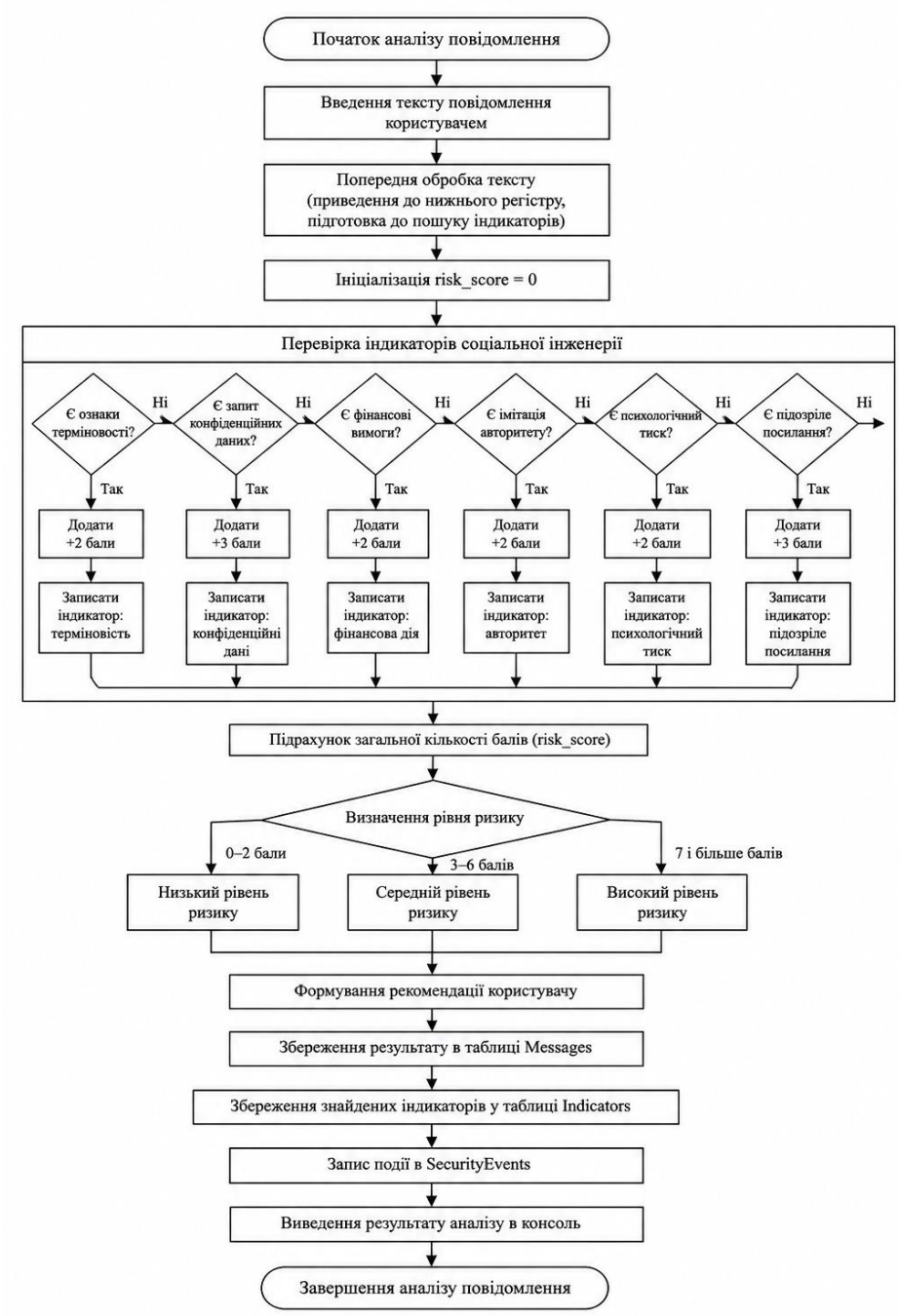

Рисунок 3.7 – Алгоритм аналізу повідомлення на ознаки соціальної інженерії

Як видно з рисунка 3.7, після введення тексту програма послідовно перевіряє повідомлення за групами індикаторів, нараховує бали, визначає рівень ризику, зберігає результат у базі даних і виводить користувачу рекомендацію.

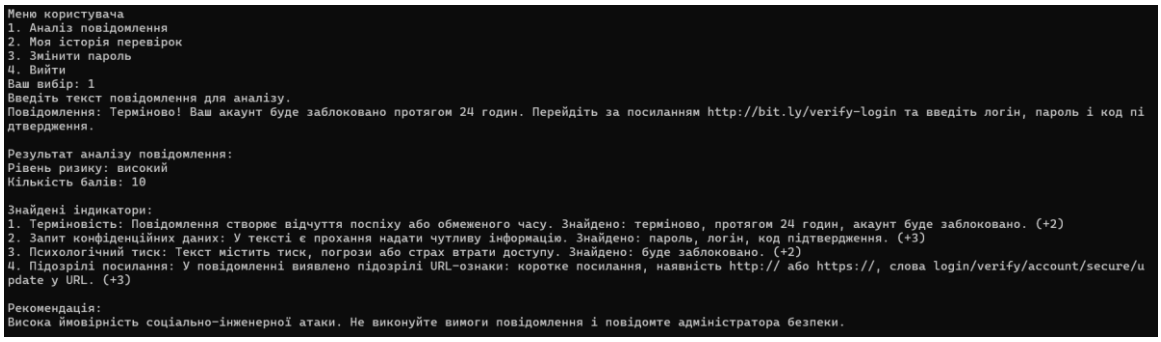

Рисунок 3.8 – Результат аналізу повідомлення з високим рівнем ризику

На рисунку 3.8 наведено приклад аналізу повідомлення, яке містить ознаки соціальної інженерії: терміновість, запит конфіденційних даних, психологічний тиск і підозріле посилання. Програма визначає рівень ризику, виводить знайдені індикатори та надає користувачу рекомендацію щодо подальших дій.

Отже, розроблений консольний застосунок демонструє практичне застосування окремих технологій протидії соціальній інженерії. У ньому поєднано авторизацію користувачів, безпечне зберігання паролів, блокування після невдалих спроб входу, рольову модель доступу, аналіз повідомлень за індикаторами ризику, збереження результатів у базі Microsoft Access і журналювання подій безпеки.

3.2 Рекомендації із застосування технологій протидії соціальній інженерії

Застосування технологій протидії соціальній інженерії має базуватися на комплексному підході, який поєднує організаційні правила, технічні засоби, навчання персоналу та програмні інструменти. Окремий засіб захисту не може повністю усунути ризик соціальної інженерії, оскільки атаки цього типу спрямовані не лише на технічну інфраструктуру, а й на поведінку користувачів. Тому основним завданням є зменшення ймовірності помилки користувача та обмеження наслідків у разі її виникнення.

Першим напрямом є удосконалення організаційних процедур. В організації мають бути визначені правила обробки конфіденційної інформації, порядок перевірки підозрілих запитів, правила використання електронної пошти, месенджерів і зовнішніх інформаційних ресурсів. Особливу увагу потрібно приділяти критичним діям: передачі паролів, зміні прав доступу, відкриттю файлів від невідомих відправників, оплаті рахунків або переходу за посиланнями з повідомлень. Такі дії мають виконуватися лише після перевірки через незалежний канал зв’язку.

Другим напрямом є навчання персоналу. Користувачі повинні знати типові ознаки соціально-інженерних атак: термінові формулювання, погрози блокування, запити на введення паролів, підозрілі посилання, повідомлення від імені керівника або служби безпеки. Навчання має бути регулярним і практично орієнтованим. Доцільно використовувати приклади реальних або навчальних повідомлень, тестові завдання, короткі інструктажі та симуляції типових атак.

Третім напрямом є застосування технічних засобів. До них належать фільтрація електронної пошти, перевірка вкладень, блокування небезпечних посилань, багатофакторна автентифікація, розмежування прав доступу, журналювання подій і моніторинг підозрілої активності. Такі засоби дозволяють зменшити кількість небезпечних повідомлень, які потрапляють до користувача, а також обмежити наслідки можливої компрометації облікового запису.

Четвертим напрямом є використання програмних засобів, які допомагають аналізувати повідомлення на наявність ознак соціальної інженерії. Розроблений у межах роботи консольний застосунок може використовуватися як навчальний або демонстраційний інструмент. Він дозволяє показати користувачу, які саме ознаки можуть свідчити про ризик: терміновість, запит конфіденційних даних, фінансові вимоги, психологічний тиск, імітація авторитетного джерела або підозрілі посилання. Аналіз у програмі є правило-орієнтованим, тобто виконується на основі заздалегідь визначених індикаторів, а не повноцінного семантичного аналізу тексту .

Для перевірки працездатності програмного засобу було проведено ручне тестування основних функцій. Перевірялися сценарії реєстрації, авторизації, блокування користувача, аналізу повідомлення, збереження результатів і перегляду журналів. Результати тестування наведено в таблиці 3.4.

Таблиця 3.4

Результати тестування основних функцій програми

Тестовий сценарій	Очікуваний результат	Фактичний результат	Статус
Реєстрація нового користувача	Створення нового облікового запису в базі даних	Користувач створюється, дані зберігаються в таблиці Users	Успішно
Реєстрація з уже наявним логіном	Виведення повідомлення про помилку	Програма повідомляє, що логін уже існує	Успішно
Авторизація з правильним паролем	Перехід до меню відповідної ролі	Користувач входить у систему та бачить меню	Успішно
Авторизація з неправильним паролем	Збільшення кількості невдалих спроб	Спроба входу фіксується в журналі	Успішно
Блокування після 3 невдалих спроб	Обліковий запис блокується	Користувач блокується після трьох помилок	Успішно
Аналіз безпечного повідомлення	Визначення низького рівня ризику	Програма виводить низький рівень ризику	Успішно
Аналіз підозрілого повідомлення	Визначення середнього рівня ризику	Програма виявляє окремі індикатори ризику	Успішно
Аналіз небезпечного повідомлення	Визначення високого рівня ризику	Програма виводить високий ризик і рекомендацію	Успішно
Збереження результату аналізу	Запис повідомлення до бази даних	Дані зберігаються в таблиці Messages	Успішно
Збереження індикаторів	Запис знайдених ознак до бази даних	Індикатори зберігаються в таблиці Indicators	Успішно

Продовження таблиці 3.4

Тестовий сценарій	Очікуваний результат	Фактичний результат	Статус
Перегляд журналу входів адміністратором	Виведення записів про входи	Адміністратор бачить записи LoginLog	Успішно
Розблокування користувача адміністратором	Зняття блокування облікового запису	Користувач розблоковується	Успішно

Як видно з таблиці 3.4, основні функції програми виконуються відповідно до очікуваних результатів. Тестування підтвердило працездатність механізмів реєстрації, авторизації, блокування користувача, аналізу повідомлень, збереження результатів і журналювання подій. Це дозволяє використовувати розроблений застосунок як демонстраційний приклад програмного засобу підтримки протидії соціальній інженерії.

Разом із тим необхідно враховувати обмеження реалізації. Програма не є повноцінною корпоративною системою кіберзахисту, не виконує автоматичну фільтрацію електронної пошти, не аналізує вкладення, не підключається до поштових скриньок і не проводить реальні фішингові кампанії. Її призначення полягає в демонстрації базових принципів: контролю доступу, журналювання, аналізу повідомлень за індикаторами ризику та збереження результатів у базі даних.

Основні рекомендації щодо застосування технологій протидії соціальній інженерії наведено в таблиці 3.5.

Таблиця 3.5

Рекомендації щодо застосування технологій протидії соціальній інженерії

Напрям	Рекомендація	Очікуваний ефект
Організаційний	Затвердити правила перевірки критичних запитів	Зменшення ризику виконання фальшивих інструкцій

Продовження таблиці 3.5

Організаційний	Розмежувати права доступу відповідно до службових обов’язків	Обмеження наслідків компрометації облікового запису
Навчальний	Проводити регулярне навчання персоналу з розпізнавання фішингу	Підвищення обізнаності користувачів
Навчальний	Використовувати приклади підозрілих повідомлень під час інструктажів	Формування практичних навичок перевірки повідомлень
Технічний	Використовувати багатофакторну автентифікацію	Зменшення ризику несанкціонованого входу після викрадення пароля
Технічний	Налаштувати фільтрацію електронної пошти та блокування небезпечних посилань	Зменшення кількості небезпечних повідомлень
Технічний	Вести журнал входів і подій безпеки	Можливість аналізу підозрілої активності
Програмний	Використовувати засоби аналізу повідомлень на індикатори ризику	Допомога користувачу у виявленні потенційних атак
Контрольний	Регулярно перевіряти журнали та результати інцидентів	Своєчасне виявлення повторюваних проблем
Навчально-демонстраційний	Використовувати розроблений застосунок для пояснення ознак соціальної інженерії	Краще розуміння користувачами типових сценаріїв атак

Застосування зазначених рекомендацій дозволяє підвищити стійкість організації до соціально-інженерних атак. Найбільш ефективним є не окреме використання одного засобу, а їх поєднання: організаційні правила визначають порядок дій, навчання формує обізнаність користувачів, технічні засоби зменшують кількість загроз, а програмні інструменти допомагають виявляти підозрілі ознаки в повідомленнях.

У практичному застосуванні розроблений консольний застосунок може використовуватися для демонстрації базових сценаріїв соціальної інженерії під час навчання користувачів. Наприклад, адміністратор або викладач може показати, як змінюється рівень ризику повідомлення залежно від наявності термінових формулювань, запитів на пароль, посилань або імітації авторитетного

джерела. Це дозволяє не лише пояснити теоретичні ознаки атак, а й показати їх на простому програмному прикладі.

У подальшому програмний засіб може бути розширений. Доцільно передбачити можливість імпорту повідомлень із текстових файлів, додати графічний інтерфейс, розширити базу індикаторів, реалізувати експорт результатів у PDF або Excel, додати статистичні звіти та фільтрацію журналів за датою, користувачем або типом події. Такі можливості дали б змогу зробити застосунок зручнішим для практичного використання та демонстрації результатів аналізу.

Отже, рекомендації із застосування технологій протидії соціальній інженерії передбачають поєднання організаційних, навчальних, технічних і програмних заходів. Розроблений консольний застосунок підтверджує можливість практичної реалізації окремих механізмів такого підходу: авторизації користувачів, захисту паролів, блокування після невдалих спроб входу, аналізу повідомлень на індикатори ризику, збереження результатів і журналювання подій безпеки.

Висновки до третього розділу

У третьому розділі розглянуто застосування технологій протидії соціальній інженерії на прикладі розроблення консольного програмного засобу. Практична частина роботи була спрямована на демонстрацію базових механізмів захисту користувачів, виявлення ознак соціально-інженерних атак у текстових повідомленнях, збереження результатів перевірок і журналювання подій безпеки.

У межах програмної реалізації розроблено консольний застосунок мовою Python для операційної системи Windows. Для зберігання даних використано базу Microsoft Access, у якій передбачено таблиці для облікових записів користувачів, перевірених повідомлень, знайдених індикаторів ризику, журналу входів і подій безпеки. Така структура дозволяє зберігати результати аналізу, пов’язувати їх із конкретним користувачем і фіксувати важливі дії в системі.

Реалізовано механізми реєстрації та авторизації користувачів, перевірку складності пароля, хешування паролів із використанням солі, блокування облікового запису після кількох невдалих спроб входу та розподіл ролей між звичайним користувачем і адміністратором. Це забезпечує базовий контроль доступу до функцій програми та демонструє застосування елементів безпеки в користувацькому програмному засобі.

Окрему увагу приділено модулю аналізу повідомлень. Програма перевіряє введений користувачем текст за групами індикаторів соціальної інженерії: терміновість, запит конфіденційних даних, фінансові дії, імітація авторитетного джерела, психологічний тиск і підозрілі посилання. Для кожної групи індикаторів нараховуються бали ризику, після чого повідомленню присвоюється низький, середній або високий рівень небезпеки.

Проведене тестування підтвердило працездатність основних функцій програмного засобу. Було перевірено реєстрацію користувачів, авторизацію, блокування після неправильних спроб входу, аналіз безпечних, підозрілих і небезпечних повідомлень, збереження результатів у базі даних, запис знайдених індикаторів, перегляд журналів і розблокування користувача адміністратором.

Сформовано рекомендації щодо застосування технологій протидії соціальній інженерії. Встановлено, що ефективний захист має поєднувати організаційні процедури, навчання персоналу, технічні засоби контролю та програмні інструменти аналізу повідомлень. Розроблений застосунок може використовуватися як навчально-демонстраційний засіб для пояснення користувачам типових ознак соціально-інженерних атак і принципів безпечної роботи з підозрілими повідомленнями.

Отже, у третьому розділі показано практичне застосування окремих технологій протидії соціальній інженерії. Розроблений програмний засіб не є повноцінною корпоративною системою кіберзахисту, однак він демонструє важливі принципи: авторизацію користувачів, захист паролів, розмежування ролей, журналювання подій, аналіз повідомлень за індикаторами ризику та збереження результатів у базі даних.

ВИСНОВКИ

За результатами дослідження технологій протидії соціальній інженерії сформовано такі висновки.

Проведено аналіз сучасного стану реалізації атак соціальної інженерії. Встановлено, що соціальна інженерія залишається небезпечною загрозою через орієнтацію на людський фактор, зокрема довіру, поспіх, страх, необізнаність або помилки користувачів під час роботи з інформацією. На відміну від суто технічних атак, соціально-інженерні методи часто не потребують складного злому системи, оскільки користувач може самостійно надати зловмиснику пароль, перейти за небезпечним посиланням або виконати фальшиву інструкцію.

Проаналізовано основні форми соціально-інженерних атак: фішинг, спір-фішинг, вішинг, смішинг, претекстинг, бейтинг, квід про кво та тейлгейтинг. Визначено, що такі атаки можуть реалізовуватися через електронну пошту, телефонні дзвінки, SMS, месенджери, фальшиві вебсторінки, фізичні носії та безпосередню взаємодію із жертвою.

Розглянуто основні підходи та методи протидії соціальній інженерії. Встановлено, що ефективний захист має бути комплексним і поєднувати організаційні заходи, навчання персоналу, технічні засоби контролю та програмні інструменти. Такий підхід дозволяє не лише зменшити ймовірність успішної атаки, а й обмежити її можливі наслідки.

Досліджено організаційні технології протидії соціальній інженерії. Визначено, що до них належать політика інформаційної безпеки, розмежування прав доступу, перевірка критичних запитів, навчання персоналу, моделювання атак, журналювання подій і порядок реагування на інциденти. Ці заходи формують основу безпечної поведінки користувачів і знижують ризик виконання небезпечних дій під впливом маніпуляції.

Проаналізовано технічні засоби й програмні рішення для захисту від соціально-інженерних атак. До таких засобів належать фільтрація електронної пошти, перевірка вкладень і посилань, протоколи SPF, DKIM і DMARC,

багатофакторна автентифікація, антивірусний захист, EDR-рішення, DLP-системи, моніторинг і журналювання подій. Встановлено, що ці засоби не усувають людський фактор повністю, але зменшують кількість небезпечних повідомлень, ускладнюють використання викрадених облікових даних і допомагають виявляти підозрілу активність.

Розглянуто інструменти моделювання та тестування атак соціальної інженерії, зокрема GoPhish, SET, King Phisher, Microsoft Attack Simulator, KnowBe4 і Proofpoint Security Awareness Training. Встановлено, що відкриті інструменти забезпечують гнучкість і можливість локального розгортання, але потребують технічної підготовки. Комерційні платформи є зручнішими для масштабного навчання персоналу, однак мають вищу вартість і залежать від зовнішньої інфраструктури.

У роботі обґрунтовано доцільність реалізації навчально-демонстраційного консольного застосунку, який поєднує механізми авторизації, журналювання, аналізу повідомлень за індикаторами ризику та збереження результатів у базі даних. Такий підхід дозволяє показати базові принципи протидії соціальній інженерії без використання складної серверної інфраструктури.

Розроблено консольний програмний засіб мовою Python для операційної системи Windows. У програмі реалізовано реєстрацію й авторизацію користувачів, перевірку складності пароля, хешування паролів із використанням солі, блокування облікового запису після кількох невдалих спроб входу, розподіл ролей admin і user, аналіз текстових повідомлень, визначення рівня ризику, збереження результатів у базі Microsoft Access і журналювання подій. Програма перевіряє повідомлення за шістьма групами індикаторів: терміновість, запит конфіденційних даних, фінансові дії, імітація авторитетного джерела, психологічний тиск і підозрілі посилання.

Проведене тестування підтвердило працездатність основних функцій програмного засобу: реєстрації, авторизації, блокування після невдалих спроб входу, аналізу повідомлень, збереження результатів у базі даних, перегляду журналів і розблокування користувача адміністратором. Практичне значення

одержаних результатів полягає в можливості використання розробленого застосунку як навчально-демонстраційного інструменту для пояснення типових ознак соціальної інженерії та принципів безпечної роботи з підозрілими повідомленнями.

Сформовано рекомендації щодо застосування технологій протидії соціальній інженерії. Доцільним є поєднання організаційних правил, навчання персоналу, багатофакторної автентифікації, фільтрації електронної пошти, журналювання подій і програмних засобів аналізу повідомлень. Подальший розвиток програмного засобу може передбачати створення графічного інтерфейсу, імпорт повідомлень із файлів, розширення бази індикаторів, аналіз вкладень, формування статистичних звітів та експорт результатів перевірок.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про основні засади забезпечення кібербезпеки України : Закон України від 05.10.2017 № 2163-VIII // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/2163-19 (дата звернення: 31.05.2026).
2. Про інформацію : Закон України від 02.10.1992 № 2657-XII // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/2657-12 (дата звернення: 31.05.2026).
3. Грищук Р. В., Даник Ю. Г. Основи кібернетичної безпеки : монографія / за заг. ред. Ю. Г. Даника. Житомир : ЖНАЕУ, 2016. 636 с.
4. Соціальна інженерія (системний аналіз) : навч. посіб. / за заг. ред. В. І. Курганевича та В. М. Петрика. Київ, 2019. 200 с. URL: https://www.researchgate.net/publication/377557864_Socialna_inzeneria_sistemnij_analiz_Navcalnij_posibnik (дата звернення: 31.05.2026).
5. Бурячок В. Л., Толубко В. Б., Хорошко В. О., Толюпа С. В. Інформаційна та кібербезпека: соціотехнічний аспект : підручник. Львів : Магнолія, 2018. 320 с. URL: https://spadok.org.ua/books/Buryachok-Osnovy-info-ta-ciberbezpeky.pdf (дата звернення: 31.05.2026).
6. Ozkaya E. Learn Social Engineering: Learn the Art of Human Hacking with an Internationally Renowned Expert. Birmingham : Packt Publishing, 2018. 557 p. URL: https://www.researchgate.net/publication/333402902_Learn_Social_Engineering (дата звернення: 31.05.2026).
7. Hadnagy C. Social Engineering: The Science of Human Hacking. 2nd ed. Hoboken : Wiley, 2018. 320 p. URL: https://repo.darmajaya.ac.id/4637/1/Social%20Engineering_%20The%20Science%20of%20Human%20Hacking%20%28%20PDFDrive%20%29.pdf (дата звернення: 31.05.2026).
8. Hadnagy C., Fincher M. Phishing Dark Waters: The Offensive and Defensive Sides of Malicious Emails. Indianapolis : Wiley, 2015. 224 p. URL:

https://lira.epac.to/DOCS-TECH/Hacking/Phishing/Phishing%20Dark%20Waters.pdf (дата звернення: 31.05.2026).

9. Cialdini R. B. Influence: The Psychology of Persuasion. Revised ed. New York : Harper Business, 2006. 336 p. URL: https://dn710600.ca.archive.org/0/items/ThePsychologyOfPersuasion/The%20Psychology%20of%20Persuasion.pdf (дата звернення: 31.05.2026).
10. Mitnick K. D., Simon W. L. The Art of Deception: Controlling the Human Element of Security. Indianapolis : Wiley Publishing, 2002. 352 p. URL: https://www.researchgate.net/publication/234806566_The_Art_of_Deception_Controlling_the_Human_Element_of_Security (дата звернення: 31.05.2026).
11. Laptiev S. Удосконалений метод захисту персональних даних від атак за допомогою алгоритмів соціальної інженерії. Кібербезпека: освіта, наука, техніка. 2022. Вип. 4(16). С. 45–62. URL: https://www.csecurity.kubg.edu.ua/index.php/journal/article/view/363/301 (дата звернення: 31.05.2026).
12. Воловенко Л. П., Мерінова С. В. Виявлення ознак соціальної інженерії та технологія протидії соціальним хакерам на підприємстві. Підприємництво та інновації. 2019. Вип. 10. С. 183–187. (дата звернення: 31.05.2026).
13. Соколов В. Ю., Курбанмурадов Д. М. Методика протидії соціальному інжинірингу на об’єктах інформаційної діяльності. Кібербезпека: освіта, наука і техніка. 2018. № 1(1). С. 6–16. URL: https://csecurity.kubg.edu.ua/index.php/journal/article/view/15 (дата звернення: 31.05.2026).
14. Цуркан О. В., Герасімов Р. П., Крук О. М. Методи протидії використанню соціальної інженерії. Інформаційні технології та спеціальна безпека. 2019. URL: https://ela.kpi.ua/items/58e63fd4-62b9-4adb-8980-80a93f7a702e (дата звернення: 31.05.2026).
15. Жмурко О. Соціальна інженерія як загроза кібербезпеці: методи запобігання та захисту. Педагогіка безпеки. 2024. Т. 9, № 1. С. 37–42. URL:

https://pedbezpeka.vntu.edu.ua/index.php/pb/article/view/151/132 (дата звернення: 31.05.2026).

16. Юдін О. К., Матвійчук-Юдіна О. В., Супрун О. М. Інформаційно-психологічна війна та технології соціального інжинірингу. Science-Based Technologies. 2021. Т. 50, № 2. URL: https://www.researchgate.net/publication/362766470 (дата звернення: 31.05.2026).
17. National Institute of Standards and Technology. Building an Information Technology Security Awareness and Training Program. NIST Special Publication 800-50. Gaithersburg : NIST, 2003. URL: https://csrc.nist.gov/publications/detail/sp/800-50/final (дата звернення: 31.05.2026).
18. National Institute of Standards and Technology. Digital Identity Guidelines: Authentication and Lifecycle Management. NIST Special Publication 800-63B. Gaithersburg : NIST, 2017. URL: https://pages.nist.gov/800-63-3/sp800-63b.html (дата звернення: 31.05.2026).
19. ENISA. Phishing: Threat Landscape and Good Practice Guide. Heraklion : European Union Agency for Cybersecurity, 2021. URL: https://www.enisa.europa.eu/publications/phishing (дата звернення: 31.05.2026).
20. Verizon. 2023 Data Breach Investigations Report. Basking Ridge : Verizon Communications, 2023. URL: https://www.verizon.com/business/resources/reports/2023-data-breach-investigations-report-dbir.pdf (дата звернення: 31.05.2026).
21. Proofpoint. State of the Phish 2023: An In-Depth Exploration of User Awareness, Vulnerability and Resilience. Sunnyvale : Proofpoint, 2023. URL: https://www.ciphertech.com.tw/wp-content/uploads/2023/09/2023-Proofpoint-State-of-the-Phish.pdf (дата звернення: 31.05.2026).
22. Anti-Phishing Working Group. Phishing Activity Trends Reports. URL: https://apwg.org/trendsreports/ (дата звернення: 31.05.2026).
23. GoPhish. GoPhish User Guide. URL: https://docs.getgophish.com/user-guide/ (дата звернення: 31.05.2026).

24. TrustedSec. Social-Engineer Toolkit. GitHub Repository. URL: https://github.com/trustedsec/social-engineer-toolkit (дата звернення: 31.05.2026).
25. Microsoft. Get started using Attack simulation training. Microsoft Learn. URL: https://learn.microsoft.com/en-us/defender-office-365/attack-simulation-training-get-started (дата звернення: 31.05.2026).
26. KnowBe4. Security Awareness Training. URL: https://www.knowbe4.com/security-awareness-training (дата звернення: 31.05.2026).

ДОДАТКИ

ДОДАТОК А

Лістинг основних модулів консольного застосунку

Лістинг А.1 - Файл main.py

```
from __future__ import annotations

import sys
from getpass import getpass

from admin_menu import admin_menu
from auth import authenticate, create_user
from database import DatabaseError
from db_init import ensure_default_users, ensure_schema
from user_menu import user_menu


def configure_console() -> None:
    if hasattr(sys.stdout, "reconfigure"):
        sys.stdout.reconfigure(encoding="utf-8")
    if hasattr(sys.stderr, "reconfigure"):
        sys.stderr.reconfigure(encoding="utf-8")


def register_user() -> None:
    print("\nРеєстрація користувача")
    username = input("Логін: ").strip()
    password = getpass("Пароль: ")
    confirm = getpass("Підтвердження пароля: ")
    if password != confirm:
        print("Паролі не збігаються.")
        return
    ok, message = create_user(username, password)
    print(message)


def login_user() -> None:
    print("\nВхід")
```

```
    username = input("Логін: ").strip()
    password = getpass("Пароль: ")
    ok, message, user = authenticate(username, password)
    print(message)
    if not ok or not user:
        return
    if user["role"] == "admin":
        admin_menu(user)
    else:
        user_menu(user)


def main_menu() -> None:
    while True:
        print("\nГоловне меню")
        print("1. Реєстрація")
        print("2. Вхід")
        print("3. Вихід")
        choice = input("Ваш вибір: ").strip()
        if choice == "1":
            register_user()
        elif choice == "2":
            login_user()
        elif choice == "3":
            print("Завершення роботи.")
            break
        else:
            print("Невірний вибір.")


def main() -> None:
    configure_console()
    try:
        ensure_schema()
        ensure_default_users()
        main_menu()
    except DatabaseError as exc:
        print(str(exc))
    except KeyboardInterrupt:
        print("\nРоботу перервано користувачем.")
```

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

Лістинг А.2 - Файл user_menu.py

```
from __future__ import annotations

from analyzer import analyze_message, save_analysis
from auth import change_password
from database import fetch_all_dicts, get_connection

try:
    from tabulate import tabulate
except ImportError:  # pragma: no cover
    tabulate = None


def print_result(result: dict) -> None:
    print("\nРезультат аналізу повідомлення:")
    print(f"Рівень ризику: {result['risk_level']}")
    print(f"Кількість балів: {result['risk_score']}")
    print("\nЗнайдені індикатори:")
    if not result["indicators"]:
        print("Індикатори не знайдено.")
    else:
        for index, indicator in enumerate(result["indicators"], start=1):
            print(f"{index}. {indicator['indicator_type']}:
{indicator['description']} (+{indicator['score']})")
    print("\nРекомендація:")
    print(result["recommendation"])


def analyze_new_message(user: dict) -> None:
    print("Введіть текст повідомлення для аналізу.")
    message_text = input("Повідомлення: ").strip()
    if not message_text:
        print("Повідомлення не може бути порожнім.")
        return
    result = analyze_message(message_text)
    save_analysis(user, result)
    print_result(result)
```

```
def view_history(user: dict) -> None:
    with get_connection() as conn:
        rows = fetch_all_dicts(
            conn.cursor(),
            """
            SELECT id, risk_score, risk_level, created_at, message_text
            FROM Messages
            WHERE user_id = ?
            ORDER BY id DESC
            """,
            (user["id"],),
        )
    if not rows:
        print("Історія перевірок порожня.")
        return
    if tabulate:
        print(tabulate(rows, headers="keys", tablefmt="grid"))
    else:
        for row in rows:
            print(row)


def user_menu(user: dict) -> None:
    while True:
        print("\nМеню користувача")
        print("1. Аналіз повідомлення")
        print("2. Моя історія перевірок")
        print("3. Змінити пароль")
        print("4. Вийти")
        choice = input("Ваш вибір: ").strip()
        if choice == "1":
            analyze_new_message(user)
        elif choice == "2":
            view_history(user)
        elif choice == "3":
            change_password(user)
        elif choice == "4":
            break
        else:
            print("Невірний вибір.")
```

Лістинг А.3 - Файл admin_menu.py

```
from __future__ import annotations

from database import fetch_all_dicts, get_connection
from security_logger import log_security_event

try:
    from tabulate import tabulate
except ImportError:  # pragma: no cover
    tabulate = None


def print_table(rows: list[dict], empty_message: str = "Дані відсутні.") -> None:
    if not rows:
        print(empty_message)
        return
    if tabulate:
        print(tabulate(rows, headers="keys", tablefmt="grid"))
        return
    headers = list(rows[0].keys())
    print(" | ".join(headers))
    for row in rows:
        print(" | ".join(str(row.get(header, "")) for header in headers))


def view_users() -> None:
    with get_connection() as conn:
        rows = fetch_all_dicts(
            conn.cursor(),
            """
            SELECT id, username, role, failed_attempts, is_blocked, created_at
            FROM Users
            ORDER BY id
            """,
        )
    print_table(rows)


def view_login_log() -> None:
```

```
    with get_connection() as conn:
        rows = fetch_all_dicts(
            conn.cursor(),
            """
            SELECT id, user_id, username_attempt, login_time, [status], [note]
            FROM LoginLog
            ORDER BY id DESC
            """,
        )
    print_table(rows)


def view_security_events() -> None:
    with get_connection() as conn:
        rows = fetch_all_dicts(
            conn.cursor(),
            """
            SELECT id, user_id, event_type, description, created_at
            FROM SecurityEvents
            ORDER BY id DESC
            """,
        )
    print_table(rows)


def view_all_messages() -> None:
    with get_connection() as conn:
        rows = fetch_all_dicts(
            conn.cursor(),
            """
            SELECT Messages.id, Users.username, Messages.risk_score,
Messages.risk_level,
                   Messages.created_at, Messages.message_text
            FROM Messages LEFT JOIN Users ON Messages.user_id = Users.id
            ORDER BY Messages.id DESC
            """,
        )
    print_table(rows)

def unblock_user(admin_user: dict) -> None:
    username = input("Введіть логін користувача для розблокування: ").strip()
    if not username:
```

```
        print("Логін не може бути порожнім.")
        return
    with get_connection() as conn:
        cursor = conn.cursor()
        cursor.execute(
            "UPDATE Users SET failed_attempts = 0, is_blocked = False WHERE
username = ?",
            username,
        )
        changed = cursor.rowcount
    if changed:
        log_security_event("user_unblocked", f"Адміністратор розблокував
користувача {username}", admin_user["id"])
        print("Користувача розблоковано.")
    else:
        print("Користувача не знайдено.")


def admin_menu(user: dict) -> None:
    while True:
        print("\nМеню адміністратора")
        print("1. Переглянути користувачів")
        print("2. Переглянути журнал входів")
        print("3. Переглянути події безпеки")
        print("4. Переглянути всі перевірені повідомлення")
        print("5. Розблокувати користувача")
        print("6. Вийти")
        choice = input("Ваш вибір: ").strip()
        if choice == "1":
            view_users()
        elif choice == "2":
            view_login_log()
        elif choice == "3":
            view_security_events()
        elif choice == "4":
            view_all_messages()
        elif choice == "5":
            unblock_user(user)
        elif choice == "6":
            break
        else:
            print("Невірний вибір.")
```

ДОДАТОК Б

Лістинг модулів авторизації, аналізу повідомлень і роботи з базою даних

Лістинг Б.1 - Файл auth.py

```
from __future__ import annotations


import base64

import hashlib

import hmac

import os

import re

from getpass import getpass

from typing import Any


from config import MAX_FAILED_ATTEMPTS, PBKDF2_ITERATIONS

from database import fetch_one_dict, get_connection, now

from security_logger import log_login, log_security_event




def validate_password(password: str) -> list[str]:

    errors: list[str] = []

    if len(password) < 8:

        errors.append("мінімум 8 символів")

    if not re.search(r"[A-ZA-ЯІЇЄҐ]", password):

        errors.append("хоча б одна велика літера")

    if not re.search(r"[a-za-яіїєґ]", password):
```

```
        errors.append("хоча б одна мала літера")
    if not re.search(r"\d", password):
        errors.append("хоча б одна цифра")
    if not re.search(r"[^A-Za-zA-Яа-яІіЇїЄєҐґ0-9]", password):
        errors.append("хоча б один спеціальний символ")
    return errors


def make_salt() -> str:
    return base64.b64encode(os.urandom(16)).decode("ascii")


def hash_password(password: str, salt: str) -> str:
    digest = hashlib.pbkdf2_hmac(
        "sha256",
        password.encode("utf-8"),
        base64.b64decode(salt.encode("ascii")),
        PBKDF2_ITERATIONS,
    )
    return base64.b64encode(digest).decode("ascii")


def verify_password(password: str, salt: str, stored_hash: str) -> bool:
    candidate = hash_password(password, salt)
    return hmac.compare_digest(candidate, stored_hash)


def get_user_by_username(username: str) -> dict[str, Any] | None:
```

```
    with get_connection() as conn:
        return fetch_one_dict(
            conn.cursor(),
            "SELECT * FROM Users WHERE username = ?",
            (username,),
        )



def create_user(username: str, password: str, role: str = "user") -> tuple[bool,
str]:
    username = username.strip()
    if not username:
        return False, "Логін не може бути порожнім."
    if get_user_by_username(username):
        return False, "Користувач із таким логіном уже існує."


    errors = validate_password(password)
    if errors:
        return False, "Пароль не відповідає вимогам: " + ", ".join(errors) + "."


    salt = make_salt()
    password_hash = hash_password(password, salt)
    with get_connection() as conn:
        conn.cursor().execute(
            """
            INSERT INTO Users (username, password_hash, salt, role,
failed_attempts, is_blocked, created_at)
            VALUES (?, ?, ?, ?, ?, ?, ?)
            """,
```

```
            username,
            password_hash,
            salt,
            role,
            0,
            False,
            now(),
        )
    user = get_user_by_username(username)
    log_security_event("registration", f"Зареєстровано користувача {username}",
user["id"] if user else None)
    return True, "Користувача успішно зареєстровано."



def authenticate(username: str, password: str) -> tuple[bool, str, dict[str, Any]
| None]:
    user = get_user_by_username(username)
    if not user:
        log_login(username, "unknown_login", "Невідомий логін")
        log_security_event("failed_login", f"Спроба входу з невідомим логіном
{username}")
        return False, "Неправильний логін або пароль.", None


    if bool(user["is_blocked"]):
        log_login(username, "blocked_user", "Спроба входу заблокованого
користувача", user["id"])
        return False, "Користувач заблокований. Зверніться до адміністратора.",
None


    if verify_password(password, user["salt"], user["password_hash"]):
        with get_connection() as conn:
```

```
            conn.cursor().execute("UPDATE Users SET failed_attempts = 0 WHERE id =
?", user["id"])

        log_login(username, "success", "Успішна авторизація", user["id"])

        log_security_event("login_success", f"Успішний вхід користувача
{username}", user["id"])

        return True, "Вхід виконано успішно.", get_user_by_username(username)



    failed_attempts = int(user["failed_attempts"]) + 1

    should_block = failed_attempts >= MAX_FAILED_ATTEMPTS

    with get_connection() as conn:

        conn.cursor().execute(

            "UPDATE Users SET failed_attempts = ?, is_blocked = ? WHERE id = ?",

            failed_attempts,

            should_block,

            user["id"],

        )

    status = "blocked_after_failed_attempts" if should_block else "wrong_password"

    note = f"Невдала спроба входу. Кількість помилок: {failed_attempts}"

    log_login(username, status, note, user["id"])

    log_security_event("failed_login", note, user["id"])

    if should_block:

        log_security_event("user_blocked", f"Користувача {username} заблоковано
після 3 невдалих спроб", user["id"])

        return False, "Користувача заблоковано після 3 невдалих спроб.", None

    return False, "Неправильний логін або пароль.", None




def change_password(user: dict[str, Any]) -> None:

    old_password = getpass("Поточний пароль: ")

    fresh_user = get_user_by_username(user["username"])
```

```
    if not fresh_user or not verify_password(old_password, fresh_user["salt"],
fresh_user["password_hash"]):
        print("Поточний пароль введено неправильно.")
        return

    new_password = getpass("Новий пароль: ")
    confirm = getpass("Підтвердження нового пароля: ")
    if new_password != confirm:
        print("Паролі не збігаються.")
        return
    errors = validate_password(new_password)
    if errors:
        print("Пароль не відповідає вимогам: " + ", ".join(errors) + ".")
        return

    salt = make_salt()
    password_hash = hash_password(new_password, salt)
    with get_connection() as conn:
        conn.cursor().execute(
            "UPDATE Users SET password_hash = ?, salt = ? WHERE id = ?",
            password_hash,
            salt,
            user["id"],
        )
    log_security_event("password_change", f"Користувач {user['username']} змінив
пароль", user["id"])
    print("Пароль успішно змінено.")
```

Лістинг Б.2 - Файл analyzer.py

```
from __future__ import annotations


import re

from dataclasses import dataclass

from typing import Any


from database import get_connection, now

from security_logger import log_security_event




@dataclass(frozen=True)

class IndicatorRule:

    indicator_type: str

    score: int

    phrases: tuple[str, ...]

    explanation: str




RULES = (

    IndicatorRule(

        "Терміновість",

        2,

        ("терміново", "негайно", "сьогодні", "протягом 24 годин", "останній шанс",
"акаунт буде заблоковано"),

        "Повідомлення створює відчуття поспіху або обмеженого часу.",

    ),

    IndicatorRule(
```

```
        "Запит конфіденційних даних",

        3,

        ("пароль", "логін", "код підтвердження", "cvv", "номер картки",
"персональні дані", "облікові дані", "код із sms"),

        "У тексті є прохання надати чутливу інформацію.",

    ),

    IndicatorRule(

        "Фінансові дії",

        2,

        ("переказ", "оплата", "рахунок", "платіж", "компенсація", "виграш",
"штраф"),

        "Повідомлення спонукає до фінансової дії або згадує гроші.",

    ),

    IndicatorRule(

        "Імітація авторитету",

        2,

        ("служба безпеки", "адміністратор", "керівник", "банк", "державна
установа", "технічна підтримка", "відділ безпеки"),

        "Відправник намагається виглядати як авторитетна особа або організація.",

    ),

    IndicatorRule(

        "Психологічний тиск",

        2,

        ("інакше", "буде заблоковано", "ви втратите доступ", "порушення",
"санкції", "негайне підтвердження"),

        "Текст містить тиск, погрози або страх втрати доступу.",

    ),

)



URL_PATTERN = re.compile(r"https?://[^\s)>\"]+", re.IGNORECASE)
```

```
IP_URL_PATTERN = re.compile(r"https?://(?:\d{1,3}\.){3}\d{1,3}(?::\d+)?(?:/|\b)",
re.IGNORECASE)

SHORTENERS = ("bit.ly", "tinyurl", "cutt.ly")

URL_KEYWORDS = ("login", "verify", "account", "secure", "update")




def _find_phrases(text_lower: str, phrases: tuple[str, ...]) -> list[str]:

    return [phrase for phrase in phrases if phrase.lower() in text_lower]




def _find_link_signals(text: str) -> list[str]:

    urls = URL_PATTERN.findall(text)

    signals: list[str] = []

    if urls:

        signals.append("наявність http:// або https://")

    for url in urls:

        lower_url = url.lower()

        if any(shortener in lower_url for shortener in SHORTENERS):

            signals.append("коротке посилання")

        if IP_URL_PATTERN.search(url):

            signals.append("посилання з IP-адресою")

        if any(keyword in lower_url for keyword in URL_KEYWORDS):

            signals.append("слова login/verify/account/secure/update у URL")

        domain_match = re.search(r"https?://([^/]+)", lower_url)

        if domain_match and len(domain_match.group(1)) > 25:

            signals.append("домен із великою кількістю символів")

    return sorted(set(signals))
```

```
def risk_level(score: int) -> str:

    if score <= 2:

        return "низький"

    if score <= 6:

        return "середній"

    return "високий"




def recommendation(level: str) -> str:

    if level == "низький":

        return "Явних ознак соціальної інженерії не виявлено, але варто
дотримуватися стандартних правил безпеки."

    if level == "середній":

        return "Повідомлення містить підозрілі ознаки. Не переходьте за
посиланнями та перевірте відправника через незалежний канал."

    return "Висока ймовірність соціально-інженерної атаки. Не виконуйте вимоги
повідомлення і повідомте адміністратора безпеки."




def analyze_message(message_text: str) -> dict[str, Any]:

    text_lower = message_text.lower()

    indicators: list[dict[str, Any]] = []



    for rule in RULES:

        matches = _find_phrases(text_lower, rule.phrases)

        if matches:

            indicators.append(

                {

                    "indicator_type": rule.indicator_type,

                    "description": f"{rule.explanation} Знайдено: {',
'.join(matches)}.",
```

```
                    "score": rule.score,
                }
            )

    link_signals = _find_link_signals(message_text)
    if link_signals:
        indicators.append(
            {
                "indicator_type": "Підозрілі посилання",
                "description": "У повідомленні виявлено підозрілі URL-ознаки: " +
", ".join(link_signals) + ".",
                "score": 3,
            }
        )

    score = sum(item["score"] for item in indicators)
    level = risk_level(score)
    return {
        "message_text": message_text,
        "risk_score": score,
        "risk_level": level,
        "indicators": indicators,
        "recommendation": recommendation(level),
    }


def save_analysis(user: dict[str, Any], result: dict[str, Any]) -> int:
    with get_connection() as conn:
        cursor = conn.cursor()
```

```
        cursor.execute(
            """
            INSERT INTO Messages (user_id, message_text, risk_score, risk_level,
created_at)
            VALUES (?, ?, ?, ?, ?)
            """,
            user["id"],
            result["message_text"],
            result["risk_score"],
            result["risk_level"],
            now(),
        )
        message_id = int(cursor.execute("SELECT @@IDENTITY").fetchone()[0])
        for indicator in result["indicators"]:
            cursor.execute(
                """
                INSERT INTO Indicators (message_id, indicator_type, description,
score)
                VALUES (?, ?, ?, ?)
                """,
                message_id,
                indicator["indicator_type"],
                indicator["description"],
                indicator["score"],
            )


    log_security_event("message_analysis", f"Користувач {user['username']} виконав
аналіз повідомлення", user["id"])
    if result["risk_level"] == "високий":
        log_security_event("high_risk_message", "Виявлено повідомлення з високим
рівнем ризику", user["id"])
```

```
    return message_id
```

Лістинг Б.3 - Файл database.py

```
from __future__ import annotations


from contextlib import contextmanager

from datetime import datetime

from pathlib import Path

from typing import Any, Iterable


try:

    import pyodbc

except ImportError:  # pragma: no cover

    pyodbc = None


from config import ACCESS_DRIVER, DATA_DIR, DB_PATH




class DatabaseError(RuntimeError):

    pass




def connection_string(db_path: Path = DB_PATH) -> str:

    return f"DRIVER={{{ACCESS_DRIVER}}};DBQ={db_path};"



def ensure_data_dir() -> None:
```

```
    DATA_DIR.mkdir(parents=True, exist_ok=True)



def create_accdb_if_missing(db_path: Path = DB_PATH) -> bool:

    ensure_data_dir()

    if db_path.exists():

        return False



    try:

        import win32com.client  # type: ignore

    except ImportError as exc:

        raise DatabaseError(

            "Файл Microsoft Access не знайдено. Встановіть Microsoft Access
Database Engine "

            "і пакет pywin32 або створіть порожній файл
data/social_engineering_defense.accdb вручну, "

            "після чого запустіть python db_init.py."

        ) from exc



    catalog = win32com.client.Dispatch("ADOX.Catalog")

    provider = f"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={db_path};"

    catalog.Create(provider)

    return True




@contextmanager

def get_connection():

    if not DB_PATH.exists():

        create_accdb_if_missing()
```

```
    if pyodbc is None:
        raise DatabaseError(
            "Пакет pyodbc не встановлено. Виконайте команду: pip install -r
requirements.txt"
        )
    try:
        conn = pyodbc.connect(connection_string(), autocommit=True)
    except pyodbc.Error as exc:
        raise DatabaseError(
            "Не вдалося підключитися до Microsoft Access. Перевірте, що
встановлено "
            "Microsoft Access Database Engine та ODBC-драйвер для .accdb."
        ) from exc
    try:
        yield conn
    finally:
        conn.close()



def table_exists(cursor: pyodbc.Cursor, table_name: str) -> bool:
    rows = cursor.tables(table=table_name, tableType="TABLE").fetchall()
    return bool(rows)



def execute(cursor: pyodbc.Cursor, sql: str, params: Iterable[Any] = ()) -> None:
    cursor.execute(sql, tuple(params))



def fetch_all_dicts(cursor: pyodbc.Cursor, sql: str, params: Iterable[Any] = ()) -
> list[dict[str, Any]]:
```

```
    cursor.execute(sql, tuple(params))

    columns = [column[0] for column in cursor.description]

    return [dict(zip(columns, row)) for row in cursor.fetchall()]



def fetch_one_dict(cursor: pyodbc.Cursor, sql: str, params: Iterable[Any] = ()) ->
dict[str, Any] | None:

    rows = fetch_all_dicts(cursor, sql, params)

    return rows[0] if rows else None



def now() -> datetime:

    return datetime.now().replace(microsecond=0)
```

Лістинг Б.4 - Файл db_init.py

```
from __future__ import annotations


from config import DEFAULT_ADMIN, DEFAULT_USER

from auth import create_user, get_user_by_username

from database import create_accdb_if_missing, get_connection, table_exists



CREATE_TABLES = {

    "Users": """

        CREATE TABLE Users (

            id AUTOINCREMENT PRIMARY KEY,

            username TEXT(100) NOT NULL,
```

```
        password_hash LONGTEXT NOT NULL,
        salt TEXT(255) NOT NULL,
        role TEXT(20) NOT NULL,
        failed_attempts INTEGER NOT NULL,
        is_blocked YESNO NOT NULL,
        created_at DATETIME NOT NULL
    )
""",
"Messages": """
    CREATE TABLE Messages (
        id AUTOINCREMENT PRIMARY KEY,
        user_id INTEGER NOT NULL,
        message_text LONGTEXT NOT NULL,
        risk_score INTEGER NOT NULL,
        risk_level TEXT(20) NOT NULL,
        created_at DATETIME NOT NULL
    )
""",
"Indicators": """
    CREATE TABLE Indicators (
        id AUTOINCREMENT PRIMARY KEY,
        message_id INTEGER NOT NULL,
        indicator_type TEXT(100) NOT NULL,
        description LONGTEXT NOT NULL,
        score INTEGER NOT NULL
    )
""",
"LoginLog": """
```

```
        CREATE TABLE LoginLog (
            id AUTOINCREMENT PRIMARY KEY,
            user_id INTEGER,
            username_attempt TEXT(100) NOT NULL,
            login_time DATETIME NOT NULL,
            [status] TEXT(100) NOT NULL,
            [note] LONGTEXT
        )
    """,
    "SecurityEvents": """
        CREATE TABLE SecurityEvents (
            id AUTOINCREMENT PRIMARY KEY,
            user_id INTEGER,
            event_type TEXT(100) NOT NULL,
            description LONGTEXT NOT NULL,
            created_at DATETIME NOT NULL
        )
    """,
}



def ensure_schema() -> None:
    create_accdb_if_missing()
    with get_connection() as conn:
        cursor = conn.cursor()
        for table_name, sql in CREATE_TABLES.items():
            if not table_exists(cursor, table_name):
                cursor.execute(sql)
```

```
        try:
            cursor.execute("CREATE UNIQUE INDEX idx_users_username ON Users
(username)")
        except Exception:
            pass



def ensure_default_users() -> None:
    for account in (DEFAULT_ADMIN, DEFAULT_USER):
        if not get_user_by_username(account["username"]):
            ok, message = create_user(account["username"], account["password"],
account["role"])
            print(message if ok else f"Не вдалося створити {account['username']}:
{message}")



def main() -> None:
    ensure_schema()
    ensure_default_users()
    print("Базу даних підготовлено.")
    print("Тестові облікові записи: admin / Admin123!, user / User123!")



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

Лістинг Б.5 - Файл security_logger.py

```
from __future__ import annotations
```

```
from typing import Optional


from database import get_connection, now



def log_login(
    username_attempt: str,
    status: str,
    note: str = "",
    user_id: Optional[int] = None,
) -> None:
    with get_connection() as conn:
        conn.cursor().execute(
            """
            INSERT INTO LoginLog (user_id, username_attempt, login_time, [status],
[note])
            VALUES (?, ?, ?, ?, ?)
            """,
            user_id,
            username_attempt,
            now(),
            status,
            note,
        )



def log_security_event(
    event_type: str,
```

```
    description: str,
    user_id: Optional[int] = None,
) -> None:
    with get_connection() as conn:
        conn.cursor().execute(
            """
            INSERT INTO SecurityEvents (user_id, event_type, description,
created_at)
            VALUES (?, ?, ?, ?)
            """,
            user_id,
            event_type,
            description,
            now(),
        )
```

ДОДАТОК В

Лістинг допоміжних файлів запуску та налаштування

Лістинг В.1 - Файл config.py

```
from pathlib import Path



BASE_DIR = Path(__file__).resolve().parent
DATA_DIR = BASE_DIR / "data"
DB_PATH = DATA_DIR / "social_engineering_defense.accdb"

ACCESS_DRIVER = "Microsoft Access Driver (*.mdb, *.accdb)"
PBKDF2_ITERATIONS = 200_000
MAX_FAILED_ATTEMPTS = 3

DEFAULT_ADMIN = {
    "username": "admin",
    "password": "Admin123!",
    "role": "admin",
}

DEFAULT_USER = {
    "username": "user",
    "password": "User123!",
    "role": "user",
}
```

Лістинг В.2 - Файл requirements.txt

```
pyodbc

tabulate

colorama

pywin32; platform_system == "Windows"
```

Лістинг В.3 - Файл start.bat

```
@echo off

chcp 65001 > nul

where py > nul 2> nul

if %errorlevel%==0 (

    py -3 main.py

    pause

    exit /b

)


where python > nul 2> nul

if %errorlevel%==0 (

    python main.py

    pause

    exit /b

)


echo Python не знайдено.
```

```
echo Встановіть Python 3.x або вимкніть alias python.exe у Windows:
echo Settings ^> Apps ^> Advanced app settings ^> App execution aliases.
pause
```