Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«17» травня 2024 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

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

спеціальність *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

освітньо-наукова програма *Кібербезпека*
(назва освітньої програми)

на тему: «Удосконалення методу протидії соціальному інжинірингу на об'єктах інформаційної діяльності з використанням штучного інтелекту»

Виконавець: студента ІІ курсу, групи КБм-21

_____ **Кирило ОЛІШЕВСЬКИЙ**
(підпис) (Ім'я, ПРІЗВИЩЕ)

	Ім'я, ПРІЗВИЩЕ	**Підпис**
Науковий керівник	Микола БРАІЛОВСЬКИЙ	
Нормоконтроль	Юрій БАБЕНКО	

Київ 2024

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«17» листопада 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

Здобувача(ки) КБм-21 (група) Олішевський Кирило Віталійович (прізвище ім'я по-батькові)

Тема кваліфікаційної роботи Удосконалення методу протидії соціальному інжинірингу на об'єктах інформаційної діяльності з використанням штучного інтелекту

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 5 від 15.11.2023 р.

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	Процеси ідентифікації та протидії атакам соціального інжинірингу на об'єктах інформаційної діяльності.
Предмет досліджень	Методи та алгоритми штучного інтелекту, що використовуються для ідентифікації та протидії атакам соціального інжинірингу, зокрема фішинговим атакам.
Мета	Удосконалення ефективного методу протидії соціальному інжинірингу, зокрема фішинговим атакам.

Вихідні дані для проведення роботи Методи та алгоритми штучного інтелекту, що використовуються для ідентифікації та протидії атакам соціального інжинірингу

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна Удосконалено комплексне рішення для прогнозування фішингових атак у реальному часі, що базується на застосуванні оптимізаційного алгоритму Бройдена-Флетчера-Гольдфарба-Шанно (БФГШ).

Практична цінність Інтеграція даної системи у сучасні інфраструктури кібербезпеки зможе значно підсилити їх захисні здібності, забезпечуючи вищий рівень захисту від широкого спектру кібератак.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Уточнення постановки задачі	17.11.2023 – 29.01.2024
Аналіз літературних джерел	30.01.2024 – 12.02.2024
Ознайомлення з сучасними трактуваннями ШІ технологій	13.02.2024 – 21.02.2024
Розгляд нормативно-правових актів, регулюючих функціонування та захист інформації	22.02.2024 – 26.02.2024
Дослідження загроз, вразливостей та атак, що спрямовуються за допомоги фішингових атак	27.02.2024 – 04.03.2024
Аналіз рекомендацій стосовно уникнення можливих загроз від фішингових атак	05.03.2024 – 10.03.2024
Дослідження існуючих заходів та засобів з забезпечення інформаційної безпеки	11.03.2024 – 17.03.2024
Розгляд відомих алгоритмів ШІ	18.03.2024 – 19.03.2024
Рорзобка ПЗ	20.03.2024 – 17.04.2024
Розробка рекомендацій щодо захисту інформаційних ресурсів від фішингових атак	18.04.2024 – 25.04.2024
Оформлення пояснювальної записки згідно методичних рекомендацій	26.04.2024 – 12.05.2024

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Подача пакету документів на розгляд ЕК	13.05.2024 – 17.05.2024

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект Зниження збитків через викрадення даних

Соціальний ефект Покращення методу забезпечення захисту інформації як особисто так і на підприємствах.

7. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ Микола БРАЇЛОВСЬКИЙ
(підпис) (Ім'я, ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ Кирило ОЛІШЕВСЬКИЙ
(підпис) (Ім'я, ПРІЗВИЩЕ)

Дата видачі завдання: 17.11.2023 р.
Термін подання кваліфікаційної роботи до ЕК 17.05.2024 р.

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи "Удосконалення методу протидії соціальному інжинірингу на об’єктах інформаційної діяльності з використанням штучного інтелекту" містить 105 сторінок основного тексту, 49 рисунків, 1 таблиця, 2 додатки та 50 літературних джерел.

Об’єкт дослідження – процеси ідентифікації та протидії атакам соціального інжинірингу на об’єктах інформаційної діяльності.

Мета даного дослідження полягає у розробці ефективного методу протидії соціальному інжинірингу, зокрема фішинговим атакам.

Методи дослідження: системний підхід, методи порівняння, структурний аналіз, кореляційно-регресійний аналіз.

У роботі досліджено вплив різноманітних атрибутів веб-сайтів на ідентифікацію фішингових ресурсів. Проведено аналіз ефективності та точності розробленої системи протидії на основі навчання моделі.

Запропоновано новий підхід до розробки алгоритмів роботи системи протидії фішинговим атакам. Побудовано математичну модель для виявлення фішингових сайтів. Розроблено трьохрівневу архітектуру класу та опис коду основних методів системи.

Практичне значення роботи полягає у підвищенні рівня захисту користувачів від фішингових атак через імплементацію розробленої системи в існуючі інформаційні інфраструктури.

Результати здійснених у дипломній роботі досліджень можуть бути використані в комерційних та державних організаціях для вдосконалення систем кібербезпеки.

Наукова новизна дослідження полягає у розробці ефективної методики виявлення фішингових сайтів з використанням алгоритму БФГШ (Бройдена-Флетчера-Гольдфарба-Шанно), що демонструє високу точність ідентифікації фішингових атак.

Напрямки подальших досліджень включають розробку та тестування нових алгоритмів машинного навчання для подальшого підвищення ефективності системи протидії, а також адаптація системи до нових видів фішингових атак.

Ключові слова: фішингові атаки, методи протидії, машинне навчання, алгоритм БФГШ, система кібербезпеки, ідентифікація фішингових сайтів, кореляційно-регресійний аналіз.

ЗМІСТ

ВСТУП

У сучасному світі, де інформаційні технології глибоко інтегровані у всі сфери людської діяльності, питання забезпечення інформаційної безпеки набуває особливої актуальності. Серед загроз інформаційній безпеці особливе місце займають атаки соціального інжинірингу, які використовують маніпуляції з людськими емоціями та психологією для отримання несанкціонованого доступу до конфіденційної інформації. Розвиток методів протидії таким атакам є важливим завданням для забезпечення цілісності, конфіденційності та доступності інформації на об’єктах інформаційної діяльності.

В останні роки зростає інтерес до використання штучного інтелекту (ШІ) як засобу підвищення ефективності систем інформаційної безпеки. Штучний інтелект має потенціал радикально трансформувати підходи до виявлення та нейтралізації спроб соціального інжинірингу завдяки своїй здатності аналізувати великі обсяги даних, вчасно ідентифікувати потенційні загрози та адаптуватися до нових методів атак.

Актуальність даної роботи визначається постійним зростанням кількості інцидентів, пов’язаних з соціальним інжинірингом, та необхідністю розробки ефективних методів їх протидії, які б враховували специфіку сучасних інформаційних просторів. Розробка методу протидії соціальному інжинірингу з використанням ШІ не тільки підвищить рівень інформаційної безпеки на об’єктах інформаційної діяльності, але й сприятиме подальшому розвитку наукових досліджень у галузі застосування штучного інтелекту для забезпечення інформаційної безпеки.

Об’єкти інформаційної діяльності в Україні, як і у всьому світі, стикаються з викликами, пов'язаними з забезпеченням інформаційної безпеки у контексті глобалізації цифрових технологій та зростаючої кіберзагрози. Посилення заходів протидії соціальному інжинірингу є критично важливим для захисту конфіденційності даних, інтелектуальної власності, а також для підтримання довіри споживачів і стабільності інформаційного простору. Використання штучного

інтелекту у цій сфері відкриває нові можливості для розробки більш ефективних і адаптивних систем захисту, які здатні прогнозувати та запобігати спробам несанкціонованого доступу до інформації через соціальний інжиніринг.

Однак, попри значний потенціал, використання ШІ у сфері протидії соціальному інжинірингу супроводжується рядом викликів, серед яких – необхідність забезпечення високої точності ідентифікації загроз, мінімізація помилкових спрацьовувань, а також забезпечення етичного використання персональних даних. Вирішення цих викликів вимагає глибокого розуміння особливостей соціального інжинірингу, постійного оновлення знань про сучасні методи атак, а також розробки комплексного підходу, що включає як технічні, так і організаційні заходи безпеки.

Мета даного дослідження полягає у розробці ефективного методу протидії соціальному інжинірингу, зокрема фішинговим атакам, на об’єктах інформаційної діяльності за допомогою використання алгоритмів штучного інтелекту, що забезпечує значне підвищення рівня захисту інформаційних ресурсів.

Для досягнення поставленої мети перед дослідженням ставляться наступні основні задачі:

– аналіз існуючих методів соціального інжинірингу та їх впливу на інформаційну безпеку, що передбачає детальне дослідження механізмів реалізації соціально-інженерних атак і визначення їх слабких місць, які можуть бути використані для розробки методу протидії;

– розробка методології ідентифікації фішингових атак з використанням технік машинного навчання, що включає вибір і обґрунтування використання конкретних алгоритмів машинного навчання для ефективного виявлення спроб соціального інжинірингу;

– розробка та верифікація алгоритму роботи методу протидії фішинговим атакам, зосереджуючись на створенні математичної моделі для розпізнавання фішингу, зборі та аналізі даних для тренування алгоритмів, а також на проєктуванні та реалізації відповідних програмних модулів;

– оцінка ефективності розробленого методу протидії, що передбачає аналіз точності, надійності системи детекції фішингу та рекомендації щодо її практичного впровадження.

Об’єкт дослідження – процеси ідентифікації та протидії атакам соціального інжинірингу на об’єктах інформаційної діяльності. Це включає в себе широкий спектр взаємодій, що зумовлені спробами несанкціонованого доступу до інформації через маніпуляції з людським фактором, а також зусилля, спрямовані на їх виявлення та нейтралізацію.

Предмет дослідження – методи та алгоритми штучного інтелекту, що використовуються для ідентифікації та протидії атакам соціального інжинірингу, зокрема фішинговим атакам. В межах цього предмету розглядаються техніки машинного навчання та аналізу даних, які дозволяють ефективно розпізнавати сайти, які використовують техніки фішингових атак.

Методи дослідження: системний підхід, методи порівняння, структурний аналіз, кореляційно-регресійний аналіз.

Наукова новизна одержаних в ході дослідження результатів полягає у наступному:

– вперше одержано комплексне рішення для прогнозування фішингових атак у реальному часі, що базується на застосуванні оптимізаційного алгоритму Бройдена-Флетчера-Гольдфарба-Шанно (БФГШ). Використання цього алгоритму в контексті протидії соціальному інжинірингу є новим підходом, який демонструє значні переваги у точності та швидкості ідентифікації потенційних загроз порівняно з існуючими методами;

– удосконалено процес аналізу та обробки великих обсягів даних за допомогою алгоритму БФГШ, що дозволяє забезпечити високу точність прогнозування фішингових атак. Це стало можливим завдяки оптимізації параметрів моделі машинного навчання, що забезпечує ефективне врахування різноманітних та динамічно змінюваних характеристик схем фішингу.

Таким чином, одержані результати дослідження відкривають нові перспективи у розробці систем протидії соціальному інжинірингу, підкреслюючи важливість інтеграції передових алгоритмів оптимізації технології штучного інтелекту.

Практичне значення одержаних результатів дослідження демонструє високу ефективність розробленої системи у превентивній відповіді на фішингові загрози в умовах реального часу. Інтеграція даної системи у сучасні інфраструктури кібербезпеки зможе значно підсилити їх захисні здібності, забезпечуючи вищий рівень захисту від широкого спектру кібератак. Впровадження цієї технології є критично важливим для зміцнення безпеки стратегічних галузей України, таких як фінансовий сектор, телекомунікації, енергетика та державне управління, надаючи надійний захист від постійно еволюціонуючих і все більш складних кіберзагроз.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРИ ТА ВИЗНАЧЕННЯ НАПРЯМІВ ДОСЛІДЖЕНЬ

1.1 Дослідження методів соціального інженірингу і їхнього впливу на захист інформації

Соціальний інжиніринг визначається як комплекс методів, що спираються на маніпулювання людьми для отримання несанкціонованого доступу до інформації, систем або мереж. Ця практика використовує психологічні хитрощі, спрямовані на викликання довіри або страху, щоб спонукати жертв до вчинення небезпечних дій, таких як розкриття конфіденційних даних, встановлення шкідливого програмного забезпечення або надання доступу до захищених систем. На відміну від традиційних кібератак, які зосереджені на експлуатації технічних вразливостей, соціальний інжиніринг звертається до «людського фактору» – найменш передбачуваної та часто найслабшої ланки в системі інформаційної безпеки.

Соціальний інжиніринг використовується зловмисниками з різноманітними цілями, від шахрайства та крадіжки ідентифікаційних даних до корпоративного шпигунства та саботажу. Ефективність таких атак заснована на здатності обходити традиційні технологічні захисти, використовуючи соціальні та психологічні механізми впливу. Враховуючи високу адаптивність людської поведінки, методи соціального інжинірингу постійно еволюціонують, ускладнюючи їх виявлення та протидію. Вплив соціального інжинірингу на захист інформації є значним, оскільки він ставить під загрозу конфіденційність, цілісність та доступність даних. Атаки соціального інжинірингу можуть призвести до значних фінансових втрат, пошкодження репутації, а також до юридичних наслідків для організацій, що стають їх жертвами. Тому розуміння та ідентифікація методів соціального інжинірингу є критично важливим для розробки ефективних стратегій захисту [1].

Методи соціального інжинірингу включають:

Фішинг

Фішинг є одним із найпоширеніших методів соціального інжинірингу, який зосереджений на виманюванні конфіденційної інформації від користувачів через масові інтернет-розсилки, що імітують легітимні повідомлення від відомих організацій або служб. У класичному варіанті фішингової атаки користувач отримує електронний лист, що містить здавалося б обґрунтовану вимогу перейти за посиланням для авторизації на сайті, який на вигляд не відрізняється від справжнього ресурсу відомої організації (рис. 1.1). Посилання веде на підставний сайт, дизайн та інтерфейс якого розроблені так, аби максимально точно копіювати оригінальний веб-сайт, вводячи користувача в оману [2].

Рисунок 1.1 – Класична схема реалізації фішингових атак

Повідомлення, відправлені шахраями, часто містять заклики до негайних дій, такі як необхідність змінити пароль або підтвердити банківські дані під загрозою блокування акаунта або втрати коштів. Створення почуття терміновості спонукає жертву до швидких і необдуманих дій. В результаті, не усвідомлюючи ризику, користувачі надають зловмисникам свої логіни, паролі, дані банківських карт або іншу особисту інформацію.

Фішингові атаки можуть мати різні форми та бути адаптовані під конкретні цільові групи, наприклад, через спеціалізовані повідомлення для співробітників певної компанії або користувачів певного сервісу, що значно підвищує ефективність таких атак. Сучасні фішингові кампанії можуть використовувати передові технології, включаючи штучний інтелект і машинне навчання, для автоматизації процесу створення і відправлення переконливих фішингових листів, а також для збору і аналізу відповідей від жертв [3].

Троян

Троян, або троянський кінь, являє собою тип шкідливого програмного забезпечення, призначеного для таємного проникнення в комп'ютерну систему жертви під виглядом легітимного програмного забезпечення. Ця техніка соціальної інженерії базується на обмані користувача, спонукаючи його до волонтерського, але неусвідомленого завантаження та встановлення шкідливого ПЗ на свій пристрій. Сценарії атаки з використанням троянів часто починаються з електронного листа, який містить спокусливі пропозиції, такі як можливість отримати додатковий дохід, виграш у лотереї, компромат на колегу, безкоштовні оновлення антивірусного програмного забезпечення або інші види «приманки» (рис. 1.2).

При завантаженні та встановленні програми, яка здається законною, користувач насправді інсталює троян на свій комп'ютер або мобільний пристрій. Ця шкідлива програма маскується під безпечні файли або програми, щоб уникнути виявлення користувачем або антивірусним програмним забезпеченням. Однак, на відміну від вірусів, трояни не розмножуються самостійно. Вони потребують дій користувача для їх розповсюдження або встановлення.

Після активації троян може виконувати широкий спектр шкідливих дій, від простого моніторингу активності користувача до крадіжки конфіденційної інформації, такої як паролі, банківські дані, особисті дані та інше. Деякі трояни здатні встановлювати додаткове шкідливе програмне забезпечення, змінювати або видаляти файли, перехоплювати контроль над пристроєм або навіть інтегрувати заражений пристрій у мережу зомбі-комп'ютерів для проведення розподілених атак відмови в обслуговуванні (DDoS) [4].

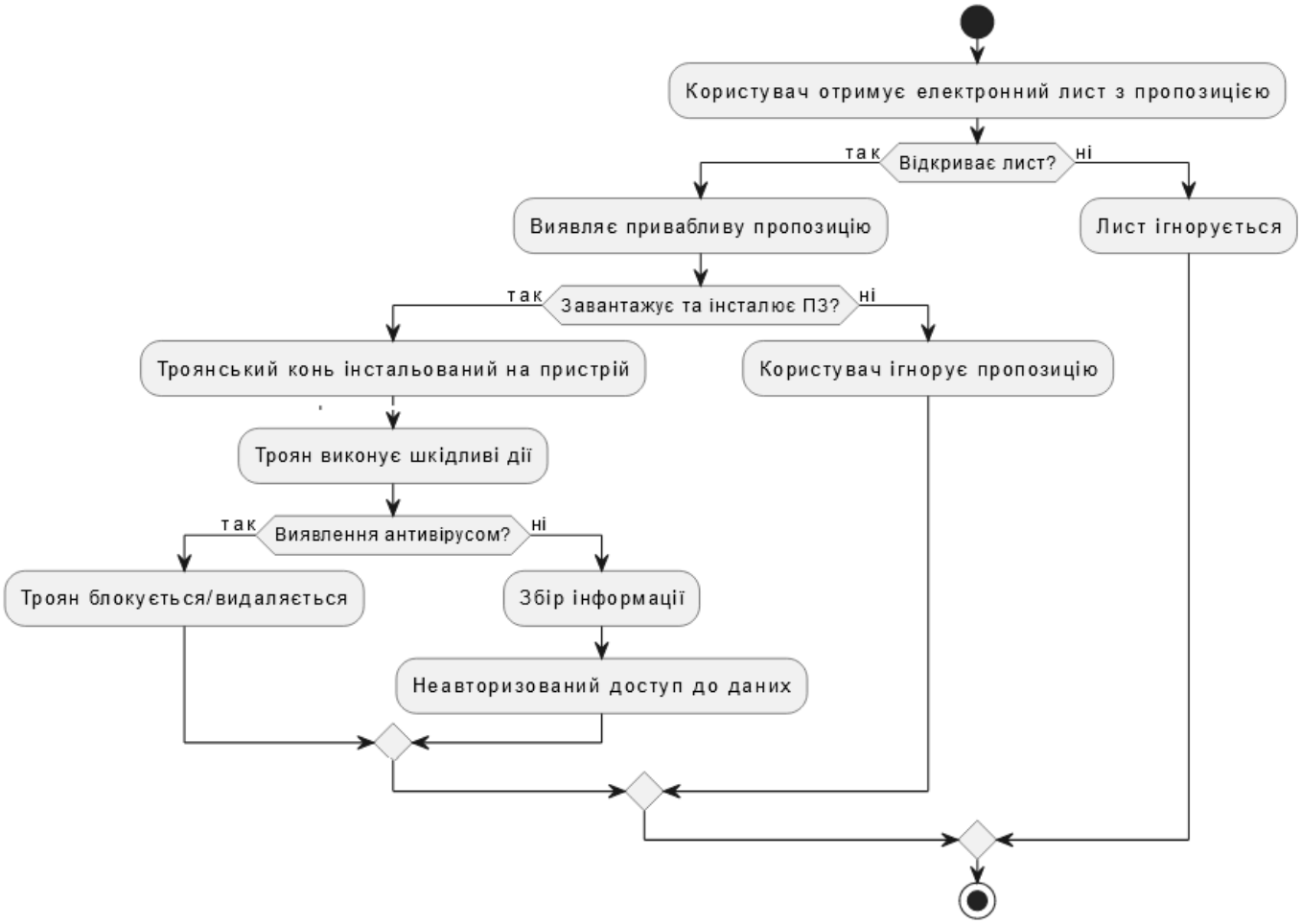

Рисунок 1.2 – Сценарії атаки з використанням троянів

Кві про кво (Послуга за послугу)

Метод «Кві про кво» у контексті соціальної інженерії ґрунтується на обміні послуг, де одна сторона надає щось цінне в обмін на виконання певної дії з боку іншої сторони. Назва цього методу походить від латинського виразу «quid pro quo», що дослівно перекладається як «послуга за послугу». В рамках кібербезпеки, цей термін описує ситуацію, коли зловмисник обіцяє жертві певну вигоду або послугу в обмін на виконання дії, яка відкриває доступ до конфіденційної інформації або систем.

Типовим сценарієм атаки «кві про кво» є ситуація, коли злочинець зв'язується з користувачем, часто за допомогою телефонного дзвінка, та представляється співробітником технічної підтримки відомої компанії або організації. Зловмисник повідомляє про існування нібито збоїв або проблем з програмним забезпеченням, яке використовує жертва, хоча насправді жодних неполадок немає. В обмін на допомогу

в усуненні цих неіснуючих проблем, зловмисник пропонує певні бонуси, такі як безкоштовні оновлення програмного забезпечення, додаткові послуги чи інші привабливі пропозиції (рис. 1.3).

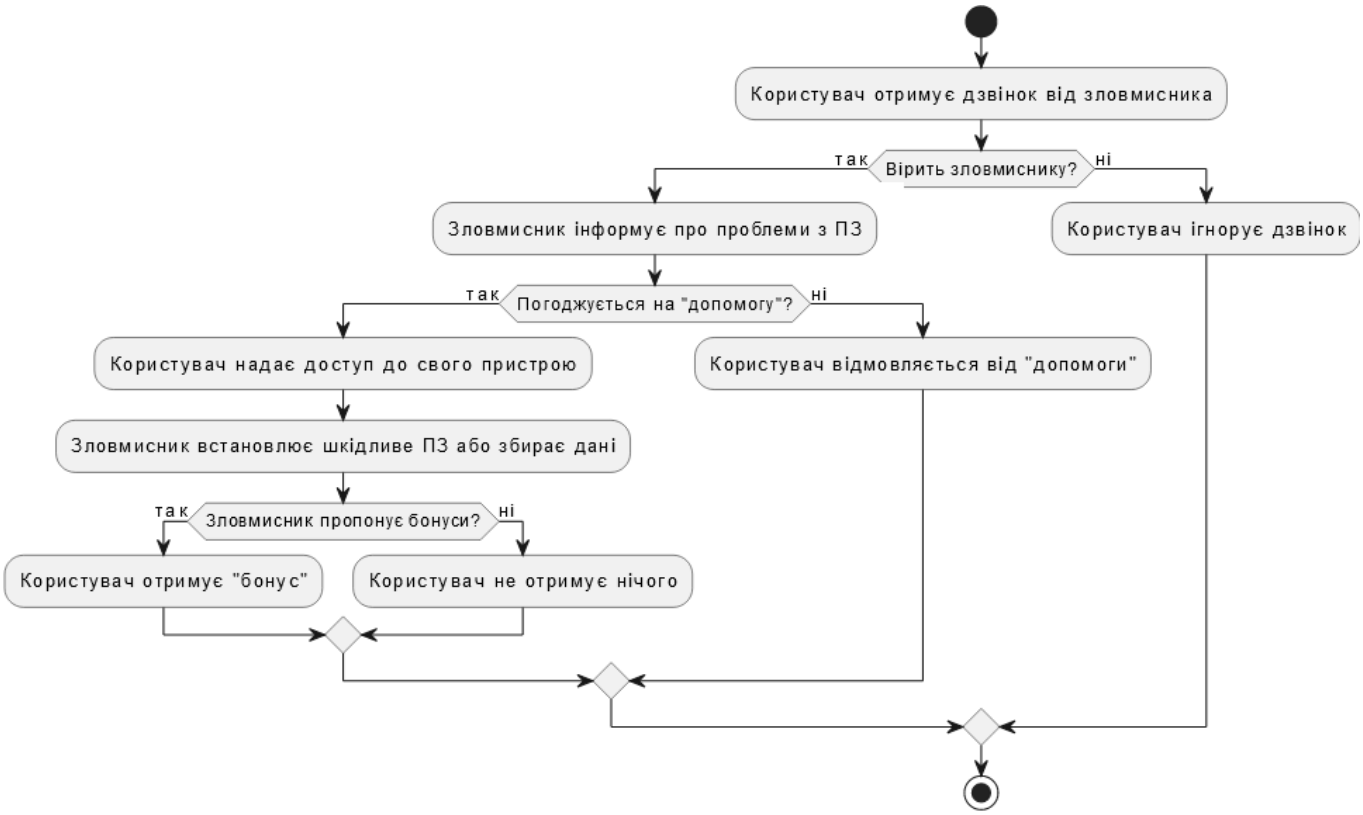

Рисунок 1.3 – Сценарії атаки з використанням «кві про кво»

Довірливі користувачі, сподіваючись отримати обіцяну вигоду, виконують вказівки зловмисника, що часто включають надання доступу до своїх систем чи конфіденційної інформації. Наприклад, їм може бути запропоновано встановити додаткове програмне забезпечення, яке насправді є шкідливим, або ввести свої логіни та паролі на підроблених веб-сайтах.

Ця тактика особливо ефективна, оскільки використовує природне бажання людей допомагати та отримувати щось взамін за свою допомогу. Це підкреслює важливість підвищення обізнаності серед користувачів щодо методів соціальної інженерії та необхідності критично ставитися до будь-яких пропозицій, які здаються занадто вигідними або вимагають виконання дій які можуть призвести до надання доступу до особистих або конфіденційних даних. Зловмисники часто використовують психологічний тиск, створюючи ілюзію негайної потреби або обмеженого часу для

пропозиції, щоб жертва під впливом емоцій прийняла небезпечне рішення. Важливо засвоїти, що легітимні компанії та організації рідко звертаються з подібними прямими запитами через незахищені канали, як-от електронна пошта або телефонні дзвінки без попередньої домовленості [5].

Претекстинг

Претекстинг є витонченою технікою соціальної інженерії, яка полягає у створенні вигаданої ситуації або контексту, в рамках якого зловмисник видає себе за іншу особу, зазвичай відому або авторитетну для жертви, з метою отримання конфіденційної інформації (рис. 1.4). Відмінною особливістю претекстингу є ретельна підготовка: зловмисник заздалегідь розробляє сценарій, в якому враховані потенційні відповіді та реакції жертви, щоб максимально переконливо втілити вигадану ідентичність.

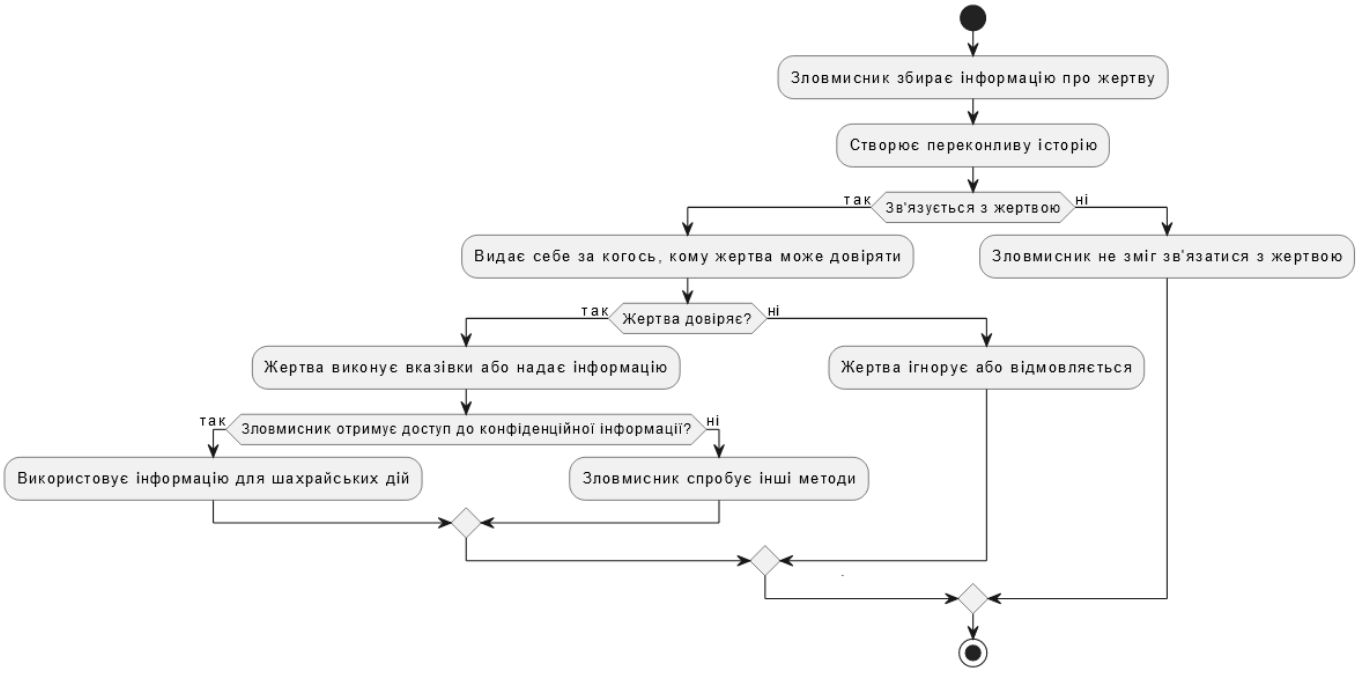

Рисунок 1.4 – Сценарії атаки з використанням претекстингу

Суть претекстингу полягає не лише в імітації особи, але й у створенні переконливої історії, що змушує жертву довіряти зловмиснику та виконувати його вказівки або надавати запитувану інформацію. Наприклад, соціальний хакер може видаавати себе за співробітника технічної підтримки, представника фінансової

установи, нового колегу або навіть за члена сім'ї жертви, використовуючи специфічні для цієї особи знання та деталі, зібрані заздалегідь.

Претекстинг часто вимагає від зловмисника глибокого дослідження задля збору інформації про жертву та її оточення, що може включати вивчення соціальних мереж, корпоративних вебсайтів, та інших відкритих джерел. Ця інформація використовується для підбору мови, термінології та інших елементів, що сприяють побудові довіри.

Однією з ключових характеристик претекстингу є його гнучкість: зловмисник готовий адаптувати свій сценарій у відповідь на реакцію жертви, використовуючи здобуту інформацію для подальшого маніпулювання або для коригування своєї стратегії з метою досягнення бажаного результату [6].

Реклама

Реклама як метод соціальної інженерії використовується злочинцями для приваблення потенційних жертв шляхом пропозиції послуг, які здаються легітимними та корисними. Цей підхід базується на створенні оголошень або рекламних кампаній, що звертаються до потреб аудиторії, наприклад, послуги з ремонту або відновлення операційної системи, як Windows (рис. 1.5). В процесі пошуку майстра або технічного спеціаліста, люди часто покладаються на власну інтуїцію та не завжди здатні об'єктивно оцінити якість пропонованих послуг. Зловмисники використовують цю вразливість, створюючи враження надійності та професіоналізму.

Після того, як потенційна жертва звертається до злочинця за допомогою, розпочинається другий етап атаки. Під виглядом виконання замовленої роботи, наприклад, ремонту або відновлення системи, шахрай може встановити на пристрій жертви шкідливе програмне забезпечення або отримати доступ до конфіденційної інформації. Таке програмне забезпечення може бути приховано під виглядом необхідних утиліт або оновлень, тим самим змушуючи жертву встановити його добровільно, не підозрюючи про реальну загрозу [7].

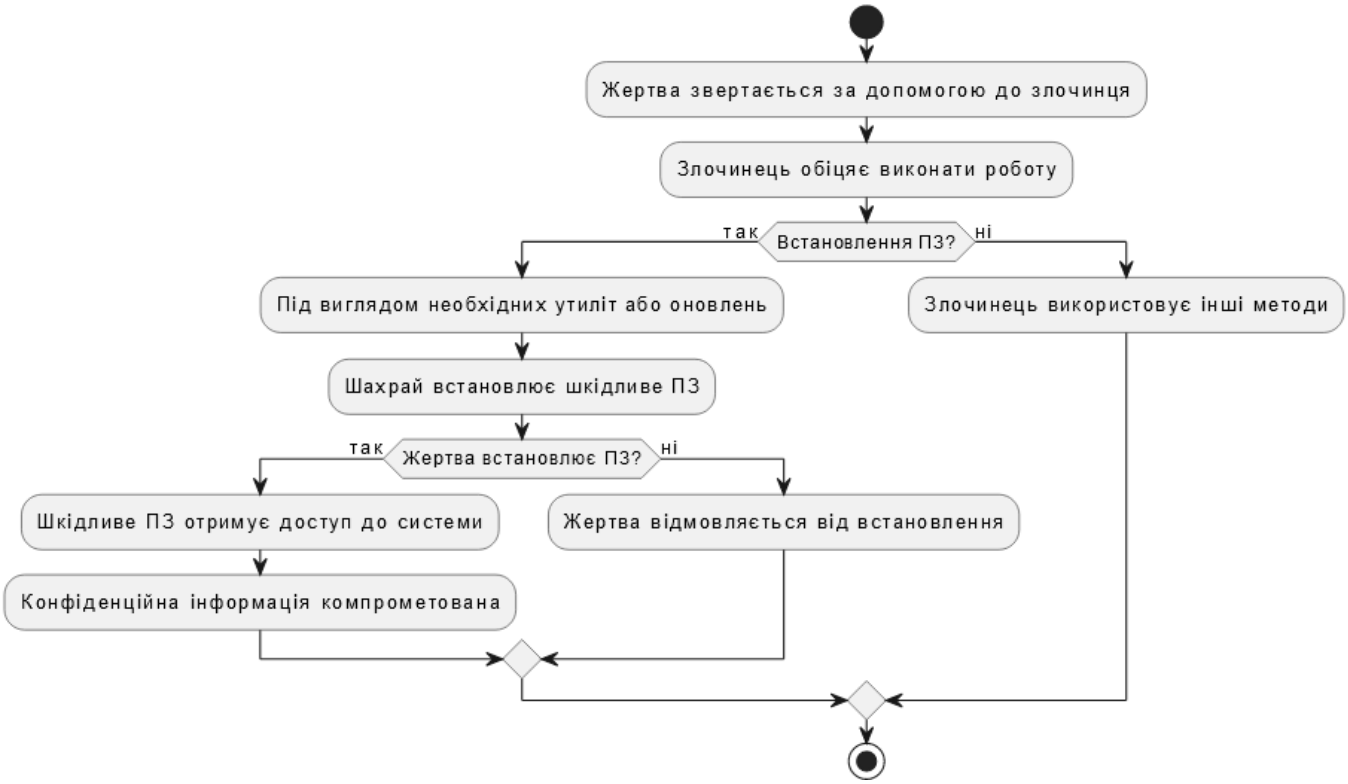

Рисунок 1.5 – Сценарії атаки з використанням реклами

Ключовим елементом успіху такої атаки є психологічна майстерність хакера, здатність побудувати довірливі відносини з жертвою та переконати її в необхідності певних дій, які насправді спрямовані на компрометацію її пристрою чи викрадення даних. Зловмисник може використовувати навіювання, тиск або маніпулювати інформацією для виведення необхідних даних у процесі комунікації.

1.2 Критичний аналіз джерел з проблематики виявлення і нейтралізації соціального інженірингу

У недавні роки дослідження у сфері боротьби з соціальним інжинірингом досягли значного прогресу, відповідаючи на еволюцію тактик, які застосовують кіберзлочинці. Основна увага приділяється трьом основним напрямкам: удосконаленню інструментів для ідентифікації загроз, розвитку стратегій психологічної протидії та збільшенню рівня освіченості серед користувачів.

В секторі розробки інструментів для виявлення зловмисних дій сталися значні прориви завдяки алгоритмам машинного навчання та штучного інтелекту. Ці

технології дозволяють аналізувати величезні масиви даних на предмет виявлення незвичайної поведінки чи комунікацій, включно з виявленням фішингових повідомлень або фальшивих інтернет-ресурсів за допомогою оцінки їхніх мовних особливостей та інших характеристик.

У площині психологічної стратегії велике значення надається аналізу мотивації та поведінкових паттернів злочинців. Розробляються методи, що допомагають людям краще ідентифікувати та уникати маніпулятивних впливів. Цей напрямок включає створення інтерактивних освітніх програм та тренінгів, мета яких - підвищити обізнаність щодо загроз соціального інжинірингу.

Освітні заходи та підвищення рівня інформованості серед співробітників компаній також відіграють ключову роль. Включення симуляцій фішингових атак, воркшопів з кібербезпеки та регулярних інформаційних оглядів про актуальні схеми шахрайства допомагають сформувати безпекову культуру, де кожен співробітник свідомий потенційних ризиків та володіє знаннями про способи їх запобігання.

Публікація [8] на порталі ScienceDirect, що представляє розроблену таксономію сценаріїв соціального інженірінгу, стала ключовим внеском у поглиблення розуміння цієї складної та вишуканої категорії кіберзлочинності. Таксономія пропонує систематизований підхід до класифікації та аналізу різних аспектів і методів, застосованих у соціальному інженірінгу. Це охоплює широкий спектр атак, включно з фішингом, вішингом, спуфінгом, та іншими техніками обману, дозволяючи детально вивчити механізми взаємодії зловмисників з жертвами, їхні методи маніпуляції інформацією та використання людських вразливостей для досягнення злочинних цілей.

Вивчення і класифікація сценаріїв соціального інженірінгу сприяють ідентифікації спільних стратегій і характеристик, що лежать в основі різноманітних атак. Це знання дозволяє розробити цілеспрямовані методи захисту, наприклад, акцентуючи увагу на ключових елементах безпеки або створюючи спеціалізовані програми тренінгів для навчання співробітників, які навчають їх ефективно розпізнавати і відповідати на потенційні загрози.

Дане дослідження вписується у широкий контекст наукових робіт, спрямованих на протистояння складним та невпинно розвиваючимся викликам у сфері кібербезпеки. Воно забезпечує цінний внесок у розробку комплексної стратегії кібербезпеки, яка буде корисною як для організацій, так і для індивідуальних користувачів, підвищуючи загальний рівень захисту в цифровому просторі.

На платформі MDPI опубліковане дослідження [9] зосереджує увагу на важливості розуміння психологічних аспектів соціального інженірінгу, акцентуючи на здатності злочинців використовувати слабкості людської психіки для обходу захисних механізмів організацій. В основі цього аналізу лежить ідея про те, що зловмисники ефективно застосовують психологічні маніпуляції, обіграючи такі людські емоції та реакції, як довіра, страх та авторитетність, щоб примусити осіб діяти на шкоду собі, надаючи важливу інформацію або доступ до захищених ресурсів.

Центральна частина дослідження присвячена детальному розгляду технік, що використовують зловмисники для створення фальшивого відчуття надійності або нагальної потреби, що мотивує жертви до передчасних дій. Автори дослідження наголошують на значущості інтеграції психологічного компоненту в загальні стратегії кібербезпеки, що дозволяє краще захиститися від маніпуляцій зловмисників.

Також у дослідженні визначено, що важливим елементом захисту організацій є освітні програми, які підвищують обізнаність і готовність співробітників ідентифікувати та ефективно реагувати на спроби соціального інженірінгу. Такі програми мають зосереджуватися на розвитку критичного мислення та психологічної стійкості, зміцнюючи захисні бар'єри організацій від психологічного впливу зловмисників. Дослідження підкреслює потребу у комплексному підході до кібербезпеки, який об'єднує як технічні, так і психологічні заходи протидії, враховуючи складність і різноманіття методів соціального інженірінгу.

На ресурсі Springer опубліковано дослідження [10], яке приділяє особливу увагу викликам соціального інженірінгу, особливо в контексті захисту мобільних пристроїв, що стає все більш актуальним у зв'язку з поширенням мобільних технологій. Автори дослідження акцентують на розробці інноваційних методик і інструментів для ефективного виявлення і запобігання таким атакам, включаючи

застосування алгоритмів машинного навчання та інших передових технологій аналізу даних, які допомагають виявляти аномалії в поведінці або комунікаційних патернах.

Центральна частина роботи зосереджена на розробці методів, що дозволяють вчасно ідентифікувати спроби соціального інженірингу, перед тим як вони зможуть завдати шкоди. Автори також підкреслюють важливість освітніх заходів, спрямованих на підвищення обізнаності серед співробітників організацій. Зокрема, рекомендуються тренінги, які містять інформаційні сесії про загрози соціального інженірингу та воркшопи з кібербезпеки, де учасники можуть навчитися розпізнавати та ефективно реагувати на маніпулятивні дії.

Додатково, в дослідженні акцентується на значенні кампаній з підвищення обізнаності, метою яких є інформування не лише співробітників, але й ширшої аудиторії про ризики, пов'язані з соціальним інженірингом. Такі кампанії можуть використовувати різноманітні медіаканали для ефективного розповсюдження інформації та зміцнення кібергігієни.

Отримані результати дослідження наголошують на необхідності інтегрованого підходу до кібербезпеки, що включає як технічні рішення, так і психологічну підготовку, дозволяючи формувати міцний фронт протистояння сучасним і все більш витонченим атакам соціального інженірингу.

У сучасному світі технології машинного навчання та нейронних мереж відіграють ключову роль у протистоянні загрозам соціального інженірингу, зокрема фішингу. Стаття [11], опублікована на визнаному ресурсі NCBI, представляє новаторський підхід до ідентифікації фішингових URL-адрес за допомогою рекурентних нейронних мереж (РНН), що є свідченням ефективного застосування передових технологій у боротьбі з кіберзагрозами. РНН, з їх унікальною здатністю обробляти послідовні дані, включаючи текст, дозволяють аналізувати URL на предмет шкідливого вмісту, використовуючи інформацію з попередніх вхідних даних для більш глибокого розуміння контексту.

Дослідження демонструє використання обширного набору даних, що містить записи про 7900 шкідливих та 5800 безпечних веб-сайтів, для перевірки та оцінки точності розробленої моделі машинного навчання у виявленні фішингу. Цей підхід

дозволяє не тільки перевірити здатність моделі розпізнавати шкідливі URL, але й оцінити її точність у класифікації веб-адрес.

Інша публікація [12] на платформі IEEE Xplore розкриває розробку моделі, що використовує алгоритми випадкового лісу (Random Forest) та дерева рішень (Decision Tree) для ідентифікації фішингових атак. Використання цих алгоритмів, які відзначаються високою ефективністю в класифікації та аналізі даних, є ідеальним для виявлення складних патернів, характерних для фішингових атак, демонструючи потенціал машинного навчання у забезпеченні кібербезпеки [13-14].

Дослідження [15], розміщене на ResearchGate, відкриває нові горизонти у використанні машинного навчання для боротьби з фішинговими атаками, акцентуючи на застосуванні класифікатора підтримки векторної машини (SVM) для розрізнення між безпечними сайтами та шкідливими ресурсами. Розроблене рішення, що підкреслює здатність SVM до високоточної класифікації, показало обнадійливі результати, точно визначаючи до 95.66% фішингових атак. Така ефективність відкриває перспективи для подальшого вдосконалення заходів кібербезпеки за допомогою SVM, який є ефективним інструментом для виявлення складних патернів великої кількості даних [16].

Серцевиною цієї техніки є здатність SVM створювати надійну модель, що заснована на аналізі представлених зразків, для відокремлення веб-сайтів за категоріями безпеки. В даному випадку, SVM ретельно аналізує атрибути сайтів, включно з їхніми URL-адресами та метаданими, щоб оцінити їх фішинговий потенціал. Важливість такої точності класифікації не можна недооцінити, враховуючи постійне ускладнення фішингових атак, які стають дедалі складнішими для виявлення .

Застосування SVM та інших методів машинного навчання представляє собою значний крок вперед у зміцненні кіберзахисту, надаючи організаціям та системам безпеки необхідні інструменти для вчасного розпізнавання та нейтралізації фішингових загроз. Такий підхід сприяє зменшенню ризику потенційних шкідливих наслідків і підвищує загальний рівень захищеності в цифровому просторі.

1.3 Сучасні стратегії та методології протидії маніпулятивним технікам соціального інженірингу

Боротьба з соціальним інженірингом вимагає інтегрованого підходу, який враховує технічні можливості, організаційну структуру та психологічний аспект взаємодії персоналу з інформаційними системами. Для ефективного протистояння такому типу атак необхідно впроваджувати комплексні рішення, які включають застосування передових технологій та активне залучення співробітників до процесу освіти та збільшення рівня кіберграмотності.

Серед найбільш ефективних методів протидії соціальному інжинірингу можна виділити:

Технічні засоби протидії

Забезпечення безпеки інформаційних систем та мереж є критично важливим завданням, особливо у контексті захисту проти соціального інженірингу. Інноваційні технічні рішення, які ілюструє рис. 1.6, є в авангарді протидії шкідливим атакам та спробам несанкціонованого доступу, впроваджуючи комплексні системи для ефективної ідентифікації, блокування та превенції таких загроз.

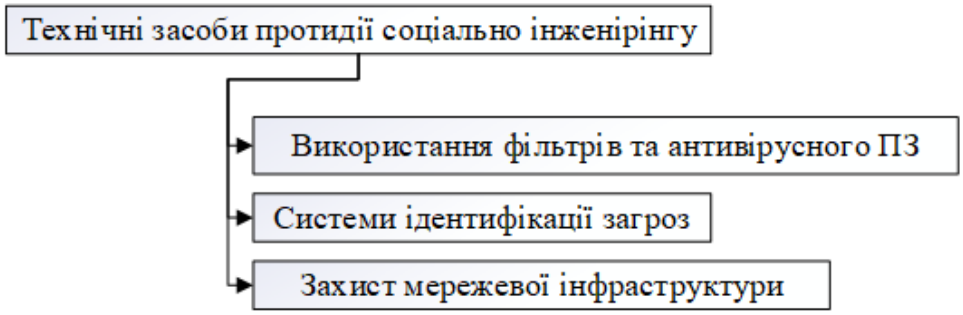

Рисунок 1.6 – Технічне забезпечення протидії методам соціального інженірингу

Ключовими компонентами в арсеналі цих технічних засобів є антивірусне програмне забезпечення та електронні фільтри, які відіграють вирішальну роль у виявленні та нейтралізації потенційно шкідливого вмісту. Антивіруси сканують програми та файли, щоб виявити наявність вірусів і троянів, запобігаючи їхньому розповсюдженню. Спам-фільтри ж аналізують електронні повідомлення, відсіюючи

листи з підозрілими посиланнями або контентом, тим самим зменшуючи ризик фішингових атак.

Системи виявлення вторгнень (IDS) та системи управління інформаційною безпекою (ISMS) стежать за мережевим трафіком, виявляючи незвичайну поведінку або дії, які можуть свідчити про спроби вторгнення. IDS працює у реальному часі, оповіщаючи про будь-яку підозрілу активність, в той час як ISMS надає загальне керівництво по безпеці інформації в організації [17-18].

Забезпечення безпеки мережевої інфраструктури є ще однією важливою задачею, яка включає заходи такі як шифрування даних, які передаються через мережу, і реалізація двофакторної аутентифікації для верифікації уповноважених користувачів. Застосування файрволів, VPN та інших мережевих технологій забезпечує додатковий рівень захисту, утруднюючи несанкціонований доступ до корпоративних ресурсів.

У цілому, впровадження цих технічних заходів створює надійну оборонну стійкість проти загроз соціального інженірінгу, включно з фішингом, вішингом, та іншими видами кібершахрайства. Вони відіграють критичну роль у захисті інформаційних активів організації, дозволяючи підтримувати цілісність та конфіденційність корпоративних даних [19].

Організаційні заходи

Зміцнення внутрішніх захисних механізмів проти атак соціального інженірінгу об'єднує в собі різноманітні організаційні стратегії та процедури, націлені на посилення безпекових процесів, підвищення рівня освіченості співробітників та контроль за доступом до важливих даних та ресурсів, що відображено на рис. 1.7.

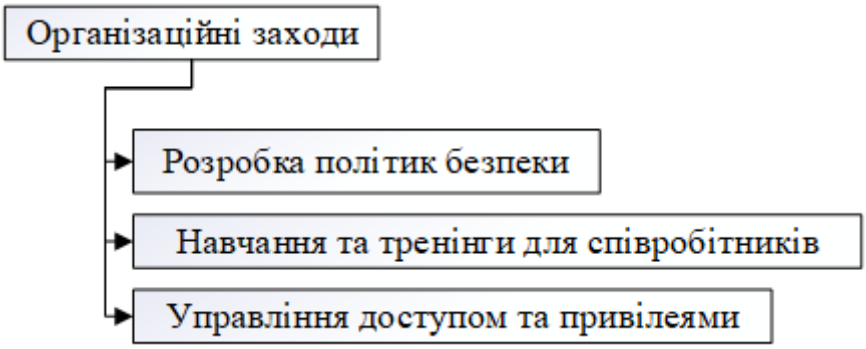

Рисунок 1.7 – Організаційні заходи протидії соціальному інженірінгу

Основою цієї стратегії є розробка та впровадження всебічних політик безпеки, які встановлюють єдині стандарти для управління інформацією, використання корпоративних активів та мережевих ресурсів. Ці політики визначають чіткі вказівки щодо обміну інформацією, доступу до систем та реакції на безпекові інциденти, включаючи атаки соціального інженірінгу, створюючи тим самим зрозумілу та послідовну основу для забезпечення безпеки.

Велике значення в організаційному захисті відіграють освітні програми для співробітників, які сприяють розвитку глибокого розуміння загроз соціального інженірінгу та навичок їх виявлення. Через тренінги, воркшопи та симуляції атак співробітники вчаться розпізнавати маніпулятивні техніки та правильно реагувати на спроби фішингу, зміцнюючи тим самим загальний рівень безпеки організації.

Контроль доступу та управління привілеями є ще однією важливою складовою, що передбачає встановлення обмежень на доступ до найважливіших систем та даних. Використання методів як двофакторна аутентифікація та моніторинг активності користувачів дозволяє ефективно запобігати неавторизованому доступу та мінімізувати ризик витоку інформації.

Усі ці організаційні ініціативи разом формують надійну основу для протистояння не тільки зовнішнім, але й внутрішнім загрозам, забезпечуючи захист конфіденційності, цілісності та доступності важливих інформаційних активів.

Психологічні аспекти

Зміцнення психологічної стійкості проти атак соціального інженірінгу лежить у фундаменті розуміння та адаптації людської поведінки для ефективного протистояння психологічним маніпуляціям, які широко використовуються зловмисниками (див. рис. 1.8).

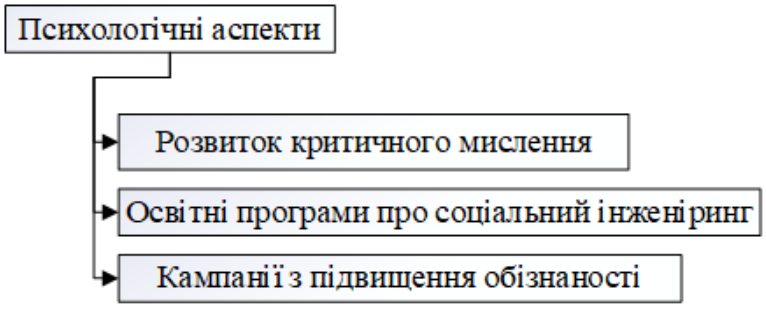

Рисунок 1.8 – Психологічні аспекти протидії соціальному інженірінгу

Підходи до психологічного захисту охоплюють:

– виховання культури критичного аналізу. Важливість вироблення уміння критично оцінювати отриману інформацію не може бути переоцінена. Це передбачає навчання людей ставити під сумнів достовірність інформації та активно аналізувати повідомлення, електронні листи чи запити, що дозволяє виявляти та уникати шахрайських дій;

– організація освітніх заходів. Реалізація освітніх програм, які ознайомлюють співробітників з тактиками та методами соціального інженірінгу, є критичною для формування психологічної оборони. Важливо надавати приклади реальних сценаріїв маніпуляцій, проводити практичні заняття з ідентифікації та відповіді на обман, розвивати усвідомлення найефективніших стратегій захисту;

– ініціювання інформаційних кампаній. Запуск кампаній, що підвищують обізнаність про загрози соціального інженірінгу та методи їхнього запобігання, стає додатковим бар'єром на шляху зловмисників. Використання інформаційних бюлетенів, організація зустрічей з обговоренням теми безпеки, поширення порад через корпоративні медіа допомагають підтримувати високий рівень усвідомлення серед співробітників.

Загалом, акцент на психологічних аспектах протидії соціальному інженірінгу дозволяє не лише підвищити рівень обізнаності та стійкість індивідів і колективів до маніпулятивних впливів, але й створити ефективний захист на рівні організації, мінімізуючи ризики від потенційних атак.

Ефективна боротьба з соціальним інженірингом вимагає інтеграції заходів на технічному, організаційному та психологічному рівнях. Застосування інтегрованого підходу, що охоплює широкий спектр методик та стратегій, є ключовим для формування надійної захисної стратегії проти різноманітних форм та методів соціального інженірінгу.

Цей комплексний підхід передбачає синергію технічних заходів безпеки, що захищають інформаційні системи від несанкціонованого доступу та шкідливих програм, організаційних принципів, що встановлюють правила поведінки та

комунікації в рамках організації, а також психологічних ініціатив, спрямованих на підвищення обізнаності та критичного мислення серед співробітників [20].

Зокрема, в рамках технічного захисту використовуються передові антивірусні програми та системи виявлення вторгнень, які забезпечують моніторинг та захист інформаційних потоків. На організаційному рівні, ключову роль відіграють розробка та дотримання детально пропрацьованих політик безпеки, що регламентують доступ до інформації та використання корпоративних ресурсів. Психологічні ініціативи, у свою чергу, включають організацію тренінгів та інформаційних кампаній, які спрямовані на зміцнення вміння співробітників розпізнавати спроби маніпуляцій та реагувати на них адекватно.

1.4 Обґрунтування розробки методу протидії фішингом атакам як пріоритетному напрямку дослідження

В епоху цифровізації, коли потоки даних невпинно збільшуються та набувають стратегічного значення, питання безпеки інформації перетворюється на критично важливий компонент управління та захисту корпоративних та особистих даних. Зокрема, фішингові атаки, як одна з найбільш розповсюджених форм кіберзагроз, вимагають розробки новітніх і ефективних методів протидії. В цьому контексті, застосування штучного інтелекту відкриває нові горизонти для захисту інформаційних систем від соціального інженірингу. Штучний інтелект, зі своєю здатністю аналізувати великі обсяги даних та виявляти складні шаблони поведінки, стає ключовим елементом в розробці ефективних захисних механізмів.

Обґрунтування розробки методу протидії фішинговим атакам з використанням штучного інтелекту як пріоритетного напрямку дослідження стає актуальним і необхідним кроком для забезпечення безпеки в інформаційному просторі, зокрема:

– зростаюча загроза фішингових атак. У сучасному цифровому світі, де обсяги онлайн-транзакцій та обміну даними невпинно зростають, фішинг став одним з найбільш поширених та ефективних методів соціального інженірингу,

використовуваних кіберзлочинцями. Це створює гостру потребу в розробці новітніх методів протидії цьому виду атак;

– комплексність та адаптивність фішингових схем. Фішингові атаки постійно еволюціонують, стаючи дедалі складнішими та витонченішими, що ускладнює їх виявлення та блокування за допомогою традиційних методів кібербезпеки. Використання штучного інтелекту дозволяє розробити більш гнучкі та адаптивні системи, здатні аналізувати поведінкові патерни та контекстуальні ознаки для ефективного виявлення фішингових атак;

– необхідність автоматизації процесів виявлення та реагування. Обсяги даних, що обробляються сучасними організаціями, вимагають автоматизації процесів виявлення та нейтралізації загроз. Штучний інтелект та машинне навчання надають таку можливість, забезпечуючи швидке та точне ідентифікування фішингових атак, що значно підвищує ефективність захисних механізмів;

– забезпечення захисту великих обсягів інформації. В епоху великих даних захист інформації на об’єктах інформаційної діяльності вимагає використання передових технологій, здатних ефективно обробляти та аналізувати великі обсяги даних. Штучний інтелект демонструє високу ефективність у виявленні та нейтралізації фішингових атак, що робить його важливим інструментом у розробці методів протидії соціальному інженірингу. Враховуючи зростаючу кількість фішингових атак та їхню все більшу витонченість, використання штучного інтелекту стає пріоритетним напрямком дослідження з метою розробки ефективних засобів захисту інформаційних систем.

Висновки за розділом 1

У рамках даного розділу було здійснено аналіз існуючих методів соціального інженірингу, включно з фішингом, використанням троянів, методами «кві про кво», претекстингом та рекламними стратегіями. Особлива увага приділена схемам реалізації фішингових атак, які були детально розкриті та аналізовані з метою зрозуміти їхню структуру та вплив на захист інформації. Критичний огляд літератури

дозволив виявити ключові аспекти виявлення та нейтралізації спроб соціального інженірингу, визначити слабкі місця існуючих підходів та відкрити нові напрямки для подальших досліджень.

Було розглянуто та оцінено сучасні стратегії та методології протидії маніпулятивним технікам, об'єднуючи технічні засоби, організаційні заходи та психологічні аспекти захисту. Такий комплексний підхід демонструє важливість інтегрованого захисту, що включає не тільки технологічні рішення, але й зміцнення внутрішніх процесів та підвищення рівня обізнаності співробітників.

Обґрунтування розробки методу протидії фішинговим атакам виступає як відповідь на зростаючу загрозу цього виду кібератак, підкреслюючи комплексність сучасних фішингових схем та необхідність впровадження автоматизованих систем виявлення та реагування на них. Висвітлено, що штучний інтелект може стати ключовим елементом у створенні ефективних рішень для захисту великих обсягів інформації та запобіганні спробам соціального інженірингу.

Результати дослідження у цьому розділі закладають фундамент для подальшого вибору методу машинного навчання його розробки та тестуванню для протидії фішингу.

РОЗДІЛ 2
МЕТОДОЛОГІЧНІ АСПЕКТИ ДОСЛІДЖЕННЯ ТА ВАЛІДАЦІЯ ПІДХОДІВ

2.1 Оцінювання технік виявлення фішингових втручань

Атаки, спрямовані на отримання конфіденційної інформації шляхом застосування інженерних прийомів маніпуляції та обману, вимагають глибокого розуміння для ефективного захисту. Розробка методів ідентифікації та протидії таким атакам передбачає знання їхніх механізмів та вміння адаптуватися до змінних тактик зловмисників, забезпечуючи надійний захист в динамічному інформаційному середовищі. Аналіз та вдосконалення стратегій виявлення фішингу є вирішальним для створення ефективних систем захисту.

Що стосується традиційних методів протидії фішингу, то вони зосереджуються на ідентифікації та блокуванні раніше відомих шахрайських дій. Базуються ці методи на певних правилах та критеріях, встановлених заздалегідь, та використовують історичну інформацію про атаки для ефективного реагування на відомі загрози. Незважаючи на ефективність проти відомих атак, ці методи можуть бути обмежені у виявленні нових та невідомих загроз через необхідність оновлення баз даних та знань. Серед традиційних інструментів виявлення виділяються:

Метод чорного списку URL

Використання чорного списку URL-адрес є однією з найбазовіших і водночас надійних стратегій для ідентифікації фішингових веб-сайтів. Цей метод включає розробку та утримання спеціалізованих баз даних, які оновлюються за рахунок додавання URL-адрес, ідентифікованих як асоційовані з фішинговими діями. Основна ідея методу полягає у безперервному скануванні трафіку в інтернеті для виявлення спроб користувачів здійснити доступ до сайтів, що внесені до чорного списку, і відповідно блокування таких спроб. Це дозволяє ефективно запобігати ризику потрапляння конфіденційної інформації користувачів у руки шахраїв. В алгоритмі роботи, представленому на рис. 2.1, детально висвітлено процес ідентифікації

фішингових атак за допомогою методу чорного списку, що включає аналіз веб-адрес, які відвідують користувачі, із подальшим їх порівнянням із зазначеними у базі даних небезпечними URL [21].

Рисунок 2.1 – Алгоритм методу чорного списку

На початковому етапі формується база даних з URL-адресами, які належать до відомих фішингових ресурсів. До цього переліку періодично додаються нові URL, виявлені завдяки різноманітним каналам, включаючи звіти від користувачів, моніторинг мережевого трафіку, дослідження в області кібербезпеки тощо. Під час спроби користувача перейти за певною веб-адресою або натиснути на лінк, інтегровані системи безпеки — такі як веб-браузери, файрволи чи антивірусне ПЗ — здійснюють перевірку запитуваної адреси з даними, що містяться у чорному списку.

У випадку збігу URL з переліком в чорному списку, доступ до потенційно шахрайського сайту блокується, а користувач отримує сповіщення про ризик фішингової атаки. Системи безпеки можуть запропонувати опцію ігнорування попередження та продовження відвідування сайту, однак такі дії вважаються ризикованими та зазвичай не рекомендуються [22].

Фіксація кожного інциденту блокування сайтів забезпечує збір важливої інформації для аналізу спроб доступу до фішингових ресурсів. Це сприяє

удосконаленню механізмів кібербезпеки та актуалізації даних чорного списку, підвищуючи ефективність захисту.

Переваги методу чорного списку URL:

– простота реалізації та використання. Метод чорного списку URL характеризується відносною простотою в розробці та інтеграції з існуючими системами безпеки, що робить його доступним рішенням для багатьох організацій та індивідуальних користувачів;

– швидке блокування відомих загроз. Завдяки використанню актуалізованого переліку шахрайських сайтів, система може миттєво реагувати на спроби доступу до небезпечних URL, ефективно запобігаючи можливі фішингові атаки;

– мінімізація помилкових спрацьовувань. Оскільки чорний список містить лише перевірені фішингові адреси, ризик блокування легітимних сайтів через помилку зводиться до мінімуму;

– підтримка з боку спільноти. Створення та оновлення бази даних фішингових сайтів часто відбувається за участі спільноти користувачів та організацій, що забезпечує широкий огляд та актуалізацію інформації.

Недоліки методу чорного списку URL:

– обмеження щодо нових та невідомих загроз. Метод не ефективний проти новостворених фішингових сайтів, які ще не були додані до чорного списку, залишаючи вікно вразливості, поки інформація не буде оновлена;

– необхідність регулярного оновлення. Ефективність методу напряму залежить від частоти оновлень чорного списку, що вимагає постійних ресурсів та уваги;

– відсутність захисту від поліморфних атак. Фішингові сайти можуть швидко змінювати свої URL-адреси, уникнувши виявлення за допомогою статичного чорного списку;

– потенційна втрата конфіденційності. Для перевірки відвідуваних сайтів проти чорного списку необхідно аналізувати інтернет-трафік користувачів, що може порушувати приватність в деяких випадках [23].

Отже, метод чорного списку URL являє собою важливий інструмент у арсеналі стратегій забезпечення кібербезпеки, спрямованих на боротьбу з фішинговими атаками. Він заснований на створенні та підтримці баз даних, що містять URL-адреси, визначені як шахрайські, та надає можливість швидко реагувати на спроби доступу до таких сайтів, блокуючи їх та попереджаючи користувачів про потенційну загрозу. Простота та відносна ефективність цього методу роблять його цінним рішенням для організацій різного масштабу та індивідуальних користувачів, які прагнуть захистити свої дані від фішингу. Водночас, обмеженість методу в умовах динамічно змінювального кіберпростору, де зловмисники невпинно розробляють нові техніки атак, вимагає його доповнення іншими методами кібербезпеки для забезпечення комплексного захисту.

Незважаючи на певні недоліки, такі як нездатність ефективно протидіяти новим або невідомим загрозам без регулярного оновлення бази даних, метод чорного списку залишається надійним засобом захисту від відомих фішингових атак. Регулярне оновлення чорного списку та використання його у поєднанні з іншими технологіями, такими як системи аналізу поведінки користувачів, машинне навчання для ідентифікації нових загроз та засоби шифрування для захисту даних, можуть значно підвищити рівень кібербезпеки.

Метод аналізу змісту веб-сторінок

Аналіз вмісту веб-сторінок відіграє ключову роль у виявленні фішингових спроб, базуючись на детальному розгляді елементів та атрибутів сторінки для виявлення ознак шахрайства. Ця стратегія включає оцінку тексту, HTML-структури, дизайну та інших технічних параметрів веб-сторінки на предмет наявності сигналів, типових для фішингу. Застосування цього методу дозволяє ідентифікувати аномалії та шаблони поведінки, що можуть свідчити про маніпулятивні наміри.

Ключовим аспектом є пошук незвичайних елементів або повторюваних патернів, які могли б вказувати на фішинг, включно з аналізом використання ключових слів або фраз, що часто асоціюються з шахрайськими повідомленнями, або перевіркою на імітацію легітимних сайтів через візуальну подібність.

Для ефективності цього методу необхідне застосування передових алгоритмів та методів обробки інформації, таких як аналіз тексту, застосування регулярних виразів, обробка зображень та використання машинного навчання для виявлення шаблонів. На рис. 2.2 представлено схему такого алгоритму, яка включає в себе застосування різних методів і технологій для точного виявлення фішингу, що вимагає регулярних оновлень та адаптацій для відстеження постійно змінюваних тактик шахраїв.

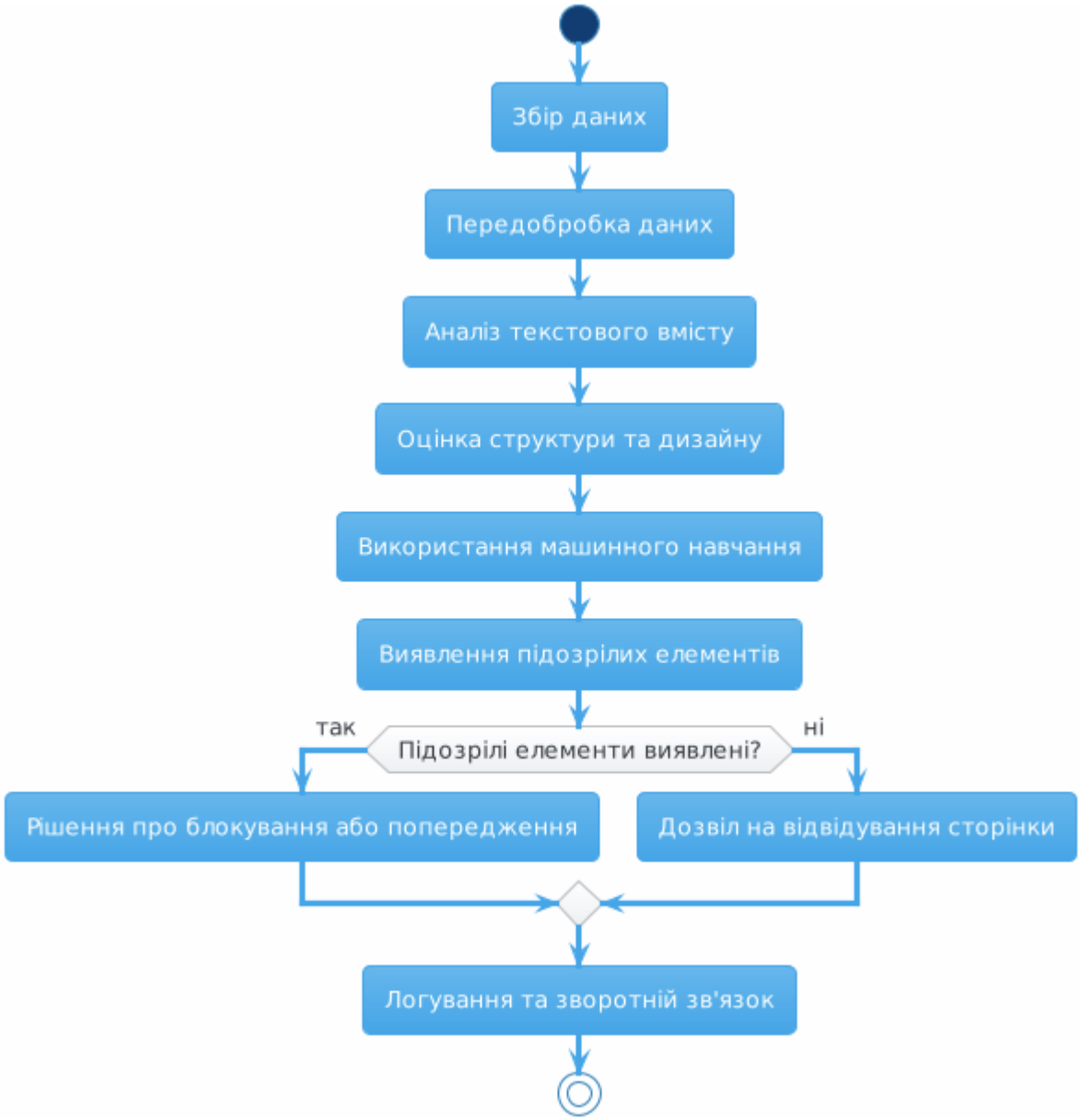

Рисунок 2.2 – Алгоритм методу аналізу змісту веб-сторінок

Процес виявлення фішингових атак за допомогою аналізу змісту веб-сторінок розпочинається з етапу збору інформації, на якому система акумулює весь доступний

контент веб-сторінки, включаючи текст, код HTML, стилі CSS, зображення та інші компоненти. Наступним кроком є предобробка зібраних даних, що може включати очищення від непотрібних елементів і стандартизацію тексту для подальшого аналізу.

Система продовжує аналізувати зібраний текст на предмет наявності специфічних слів або виразів, які зазвичай використовуються у фішингових схемах. Одночасно відбувається перевірка структури HTML і дизайну веб-сторінки на наявність характеристик, притаманних шахрайським сайтам, наприклад, невластиве застосування форм введення даних або копіювання дизайну законних сайтів [24].

У цьому контексті використовуються алгоритми машинного навчання, натреновані на великих датасетах для ідентифікації фішингових патернів, що сприяє виявленню підозрілих аспектів, які порівнюються з базою даних відомих прикладів фішингу. У кінці система може заблокувати доступ до підозрілої сторінки або ж надіслати користувачеві попередження про можливу фішингову атаку. Всі виявлені інциденти фіксуються для наступного розгляду, що допомагає поліпшити точність алгоритмів виявлення фішингу в майбутньому.

Переваги методу аналізу змісту веб-сторінок:

– висока точність виявлення. Завдяки детальному аналізу елементів веб-сторінки, включаючи текст, HTML-структуру, CSS-стилі, та зображення, цей метод може точно ідентифікувати фішингові сайти, засновуючись на характерних ознаках та шаблонах;

– здатність виявляти нові фішингові сайти. Метод аналізу змісту веб-сторінок не обмежується відомими шахрайськими URL і може ефективно ідентифікувати нові фішингові атаки на основі аналізу контенту;

– гнучкість у виявленні складних шахрайських технік. Через залучення різноманітних методів аналізу, включаючи машинне навчання, метод здатен адаптуватися та виявляти складні та поліморфні форми фішингу;

– покращення з часом. Завдяки накопиченню даних та навчанню на великих наборах інформації, системи, що використовують цей метод, можуть з часом покращувати свою точність та ефективність виявлення фішингу.

Недоліки методу аналізу змісту веб-сторінок:

– великі вимоги до обчислювальних ресурсів. Детальний аналіз контенту веб-сторінок вимагає значних обчислювальних ресурсів, особливо при використанні алгоритмів машинного навчання;

– помилкові позитивні та негативні результати. Незважаючи на високу точність, система може іноді неправильно ідентифікувати легітимні веб-сторінки як фішингові (помилкові позитивні) або пропускати деякі фішингові сайти (помилкові негативні);

– складність у аналізі динамічного контенту. Веб-сторінки, що активно використовують JavaScript або інші технології для генерації динамічного контенту, можуть ускладнити аналіз та виявлення фішингу;

– необхідність постійного оновлення та адаптації. Фішингові сайти неперервно еволюціонують, використовуючи нові методи та техніки для обходу систем безпеки, що вимагає регулярного оновлення та адаптації алгоритмів аналізу [25].

Отже, метод аналізу змісту веб-сторінок виявляється вкрай ефективним інструментом у боротьбі з фішинговими атаками, оскільки він дозволяє ідентифікувати шахрайські сайти на основі їхнього контенту, включаючи текст, HTML-структуру, CSS-стилі, та зображення. Цей підхід відрізняється високою точністю та гнучкістю, дозволяючи виявляти як відомі, так і нові форми фішингу. Завдяки застосуванню складних алгоритмів та методів обробки даних, включно з машинним навчанням, системи, які використовують цей метод, можуть адаптуватися до змін у тактиках шахраїв, покращуючи свою ефективність з часом.

Втім, цей метод має свої обмеження, включаючи великі вимоги до обчислювальних ресурсів, потенційні помилкові сигнали, складності в аналізі динамічного контенту та необхідність постійного оновлення для ефективного виявлення нових фішингових технік.

Сучасні підходи до виявлення фішингу

Сучасні стратегії боротьби з фішингом залучають новітні технології та алгоритми для розпізнавання еволюціонуючих та складних шахрайських методик. Ці підходи дозволяють детально аналізувати тактики фішерів, пропонуючи можливість

адаптації до них і надаючи змогу реалізувати більш надійний захист. Важливою є здатність систем швидко ідентифікувати нові загрози та обробляти масивні обсяги інформації. Одним із ключових напрямків є:

Методи машинного навчання

Метод машинного навчання для ідентифікації фішингових атак включає використання алгоритмів для аналізу об'ємних датасетів і виявлення зложених патернів та поведінкових індикаторів фішингу. Цей підхід базується на тренуванні моделей впізнавати характеристики фішингу через широкий спектр даних, які включають правдиві та фальшиві веб-сайти, електронні листи та інші релевантні інформаційні матеріали.

Застосування різноманітних технік машинного навчання, зокрема нейронних мереж, навчання з підкріпленням, та методів розпізнавання образів, дозволяє аналізувати і виявляти специфічні особливості даних, як-от текстовий контент електронних листів, URL-структури, метадані та користувацьку взаємодію з сайтами. Використання машинного навчання у виявленні фішингу сприяє розвитку адаптивних та гнучких систем, здатних ефективно відстежувати та реагувати на непередбачувані зміни в методах шахраїв. Алгоритм, що демонструється на рис. 2.3, ілюструє процес роботи такого методу.

Процес машинного навчання для ідентифікації фішингових атак розпочинається із збору обширного масиву даних, що включає інформацію про як автентичні, так і шахрайські ресурси, включно з веб-сайтами та електронними листами. Після очищення та стандартизації цих даних, відбувається визначення ключових атрибутів для навчання моделі, серед яких можуть бути текстові особливості, структура веб-сторінок, метадані та моделі поведінки.

У наступній фазі, модель машинного навчання навчається на вибраному датасеті, опановуючи визначення відмінностей між фішинговими запитами та легітимними взаємодіями. Після завершення навчання модель проходить етап валідації на окремому наборі даних, що дозволяє оцінити її точність та здатність до розрізнення між безпечними та шахрайськими даними.

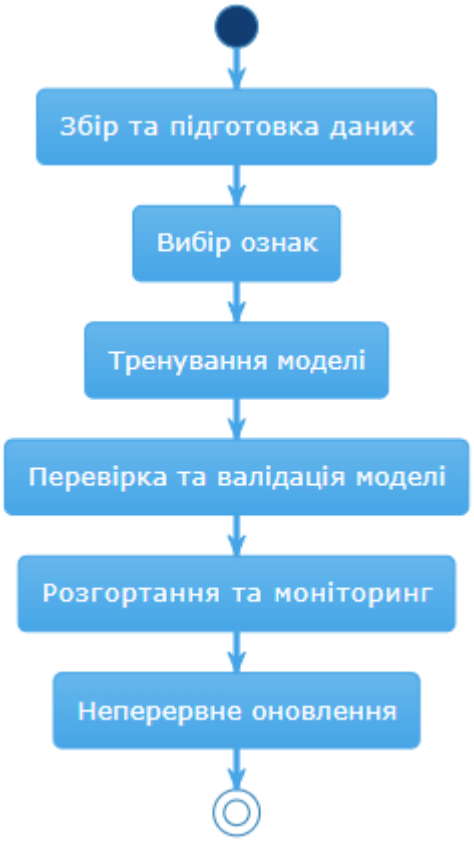

Рисунок 2.3 – Алгоритм методу машинного навчання

Оптимізована модель потім імплементується в оперативне середовище, де вона аналізує надходження інформації, як-от веб-трафік чи електронні листи, вишукування індикаторів фішингу. Останнім кроком є постійне доповнення моделі новими даними для забезпечення її здатності адаптуватися до постійно змінюваних шахрайських стратегій, що уможливлює надійне виявлення фішингових атак.

Переваги методу машинного навчання у виявленні фішингових атак:

– здатність до виявлення нових шахрайських схем. Методи машинного навчання можуть розпізнавати та адаптуватися до постійно змінюваних фішингових тактик, завдяки здатності аналізувати широкий спектр ознак та шаблонів поведінки;

– автоматизація процесу виявлення. Використання машинного навчання дозволяє автоматизувати процес ідентифікації фішингових спроб, знижуючи необхідність у постійному людському моніторингу;

– висока точність та швидкість обробки даних. Моделі машинного навчання можуть швидко обробляти великі обсяги даних і з високою точністю розрізняти між легітимними та шахрайськими активностями;

– гнучкість та масштабованість. Методи машинного навчання можуть бути застосовані до різних джерел даних та легко масштабуються залежно від потреби у виявленні фішингу.

Недоліки методу машинного навчання у виявленні фішингових атак:

– помилкові позитиви та негативи. Незважаючи на високу точність, існує ризик помилкового ідентифікування легітимних веб-сайтів як шахрайських (помилкові позитиви) або пропуску фішингових сайтів (помилкові негативи);

– необхідність великих наборів даних для тренування. Ефективність моделей машинного навчання залежить від обсягу та якості даних для тренування, що може вимагати значних ресурсів для збору та підготовки цих даних;

– складність налаштування та тюнінгу моделей. Розробка та оптимізація моделей машинного навчання вимагають високої кваліфікації та розуміння з боку спеціалістів, що може ускладнити їх впровадження та підтримку;

– вразливість до маніпуляцій. Атакуючі можуть навмисно маніпулювати даними або поведінкою, щоб обійти системи виявлення на основі машинного навчання, що вимагає постійного оновлення моделей для забезпечення їх актуальності [26].

Отже, методи машинного навчання у контексті виявлення фішингових атак виступає як могутній інструмент, який забезпечує високу точність і ефективність у боротьбі проти постійно змінюваних та складних шахрайських схем. Завдяки здатності аналізувати об'ємні набори даних і адаптуватися до нових загроз, цей метод значно підвищує можливості автоматизації та масштабування систем безпеки. Однак, ефективність методу машинного навчання залежить від якості та обсягу даних для тренування, потребує вдосконалених навичок для налаштування моделей та постійного оновлення для протидії адаптації шахраїв. Враховуючи ці особливості, метод машинного навчання є ключовим елементом у створенні більш гнучких та

адаптивних систем виявлення фішингу, хоча і вимагає комплексного підходу до реалізації та управління.

Поведінкові методи аналізу

Методика поведінкового аналізу при ідентифікації фішингових нападів фокусується на вивченні дій користувачів та системних реакцій. Основна ідея полягає в пошуку незвичних або аномальних дій, що можуть сигналізувати про спроби фішингу. Наприклад, раптове введення чутливої інформації на сайті або незвичайні запити серверу можуть бути ознаками шахрайства.

Для реалізації цього методу застосовуються різноманітні технічні засоби та аналітичні інструменти, зокрема системи детектування інтрузій, що сканують мережевий трафік, та інструменти для моніторингу користувацької активності на сайтах. Ці інструменти допомагають розпізнавати дії, які виходять за межі нормальної поведінки, спричиняючи спрацьовування алармів щодо потенційних фішингових атак.

Алгоритми поведінкового аналізу забезпечують як реактивний, так і проактивний моніторинг, дозволяючи виявляти складні або опосередковані загрози, які відхиляються від стандартної поведінки. Такий підхід виявляється надзвичайно корисним для ідентифікації нових чи адаптованих форм фішингових нападів, як це зображено на рис. 2.4, і є ключовим у протидії сучасним шахрайським схемам.

Процедура поведінкового аналізу для ідентифікації фішингових атак ініціюється зі стадії моніторингу, протягом якої здійснюється збір даних про дії користувачів та системні операції. Це охоплює аналіз мережевого трафіку, обробку серверних запитів, відстеження користувацької активності на сайтах та інтеракцій з інтерфейсами. На основі цієї інформації формується базовий профіль типової поведінки, що слугує еталоном для порівняння.

В подальшому етапі ведеться неперервний аналіз поточної активності з метою виявлення відхилень від стандартної поведінки, які можуть свідчити про спроби фішингу. Аномалії, такі як незвична активність або атипові запити, розглядаються як можливі ознаки шахрайства, що спонукає систему до видачі попереджень для подальшого розслідування чи вжиття заходів безпеки.

Рисунок 2.4 – Алгоритм методу поведінкового аналізу

Завершальний етап обумовлюється циклічним удосконаленням і адаптацією системи через інтеграцію новозібраної інформації, що дозволяє підвищувати точність розпізнавання легітимних дій та мінімізувати помилкові тривоги. Такий підхід сприяє еволюції ефективності методу в часі, забезпечуючи його високу надійність у розпізнаванні фішингових атак.

Переваги методу поведінкового аналізу:

– динамічне виявлення атак. Метод дозволяє ідентифікувати нові та невідомі фішингові атаки в реальному часі, завдяки аналізу аномальної поведінки замість статичних ознак;

– зниження помилкових спрацьовувань. Оскільки метод базується на аналізі поведінки, а не лише на змісті, він може зменшити кількість помилкових спрацьовувань порівняно з традиційними методами виявлення, що використовують сигнатури або чорні списки;

– гнучкість та адаптивність. Поведінковий аналіз може адаптуватися до змін у поведінці користувачів та шахраїв, що робить його ефективним проти постійно еволюціонуючих загроз;

– виявлення складних атак. Метод ефективний у виявленні складних фішингових атак, які можуть не мати явних шкідливих ознак або маскуватися під легітимні поведінкові патерни.

Недоліки методу поведінкового аналізу:

– великі обсяги даних. Для ефективного аналізу поведінки потрібно збирати та обробляти великі обсяги даних, що може бути ресурсомістким;

– потреба в постійному оновленні. Щоб залишатися ефективними, системи поведінкового аналізу вимагають постійного оновлення та налаштування для адаптації до нових поведінкових патернів та загроз;

– складність реалізації. Розробка та впровадження систем поведінкового аналізу може бути складною задачею, що вимагає глибоких знань у сфері безпеки та аналітики даних;

– приватність та етичні питання. Моніторинг та аналіз поведінки користувачів може порушувати приватність і викликати етичні занепокоєння, особливо якщо користувачі не повідомлені про такий моніторинг або не дали на нього згоду [27].

Отже, метод поведінкового аналізу у виявленні фішингових атак пропонує динамічний та адаптивний підхід до ідентифікації шахрайських дій, базуючись на моніторингу та аналізі аномальної поведінки користувачів і систем. Цей метод демонструє високу ефективність у виявленні як традиційних, так і новітніх, складних фішингових загроз, забезпечуючи зниження помилкових спрацьовувань та гнучке реагування на еволюцію шахрайських тактик. Однак, виклики, пов'язані з великими обсягами необхідних даних, складністю впровадження, потребою в постійному оновленні системи та забезпеченням приватності користувачів, вимагають ретельного планування та управління. Незважаючи на ці виклики, метод поведінкового аналізу є ключовим елементом сучасних стратегій кібербезпеки, забезпечуючи комплексний захист від фішингу.

Глибинний аналіз доменних імен

Метод детального дослідження доменних назв в рамках ідентифікації фішингових нападів орієнтований на глибоке вивчення доменів та їх характеристик для виявлення сайтів з потенційною загрозою. Він охоплює не лише аналіз самого домену, а й вивчення додаткової інформації, наприклад, історію реєстрації, вік домену, дані про власника, та асоційовані поведінкові моделі.

Ключовим елементом цієї методики є виявлення фішингових сайтів, які намагаються імітувати законні ресурси за допомогою дрібних змін у назві домену, використання варіацій доменних рівнів, або перестановки букв. Особлива увага приділяється перевірці SSL-сертифікатів, оскільки легітимні сайти зазвичай використовують зашифровані з'єднання, в той час як на фішингових сайтах можуть бути встановлені фальшиві або неактивні сертифікати.

Також, в аналіз включається оцінка «репутації» домену, яка може брати до уваги зміни власників, розташування серверів, динаміку DNS-записів та інші важливі аспекти. Нові або часто змінювані домени, наприклад, можуть вважатися підозрілими.

Цей підхід стає ключовим елементом стратегії протидії фішингу, надаючи змогу вчасно розпізнавати та нейтралізувати потенційні загрози шляхом інформування користувачів або обмеження доступу до сумнівних доменів. Алгоритм зосереджений на ретельному вивченні доменних назв та атрибутів дозволяє системі ефективно виявляти та превентивно реагувати на можливі фішингові атаки (рис. 2.5).

Процес використання методу вивчення доменних імен для виявлення потенційних фішингових сайтів розпочинається зі збору розширеної інформації про кожен домен. Це охоплює деталі реєстрації, час існування домену, історію його змін, дані про власника, серед інших ключових атрибутів. Далі відбувається деталізований аналіз на предмет схожості з легітимними доменами, з метою виявлення можливих орфографічних відхилень, схожості символів або альтернацій на рівні доменних зон.

Особлива увага приділяється перевірці SSL-сертифікатів на їх наявність та автентичність, оскільки фальшиві або відсутні сертифікати можуть бути ознаками шахрайства. В рамках аналізу також оцінюється репутація домену, включаючи зміни

власників, місцезнаходження серверів, та інші сигнали, які можуть вказувати на ненадійність ресурсу.

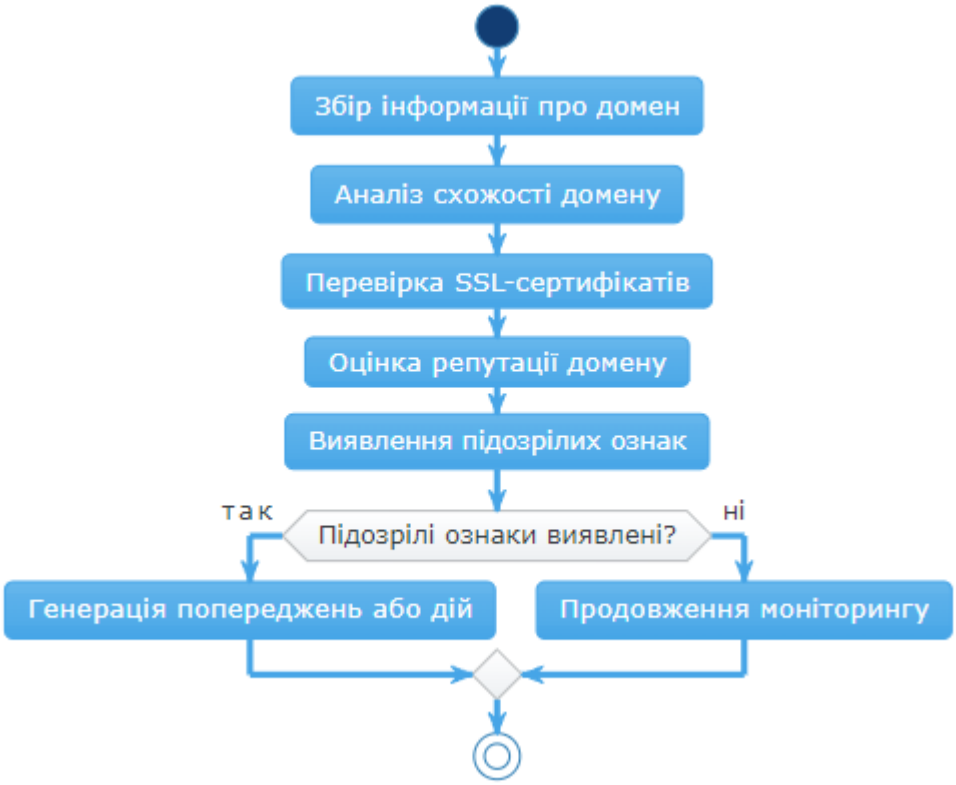

Рисунок 2.5 – Алгоритм методу глибинного аналізу

Завершальний етап полягає у синтезі всієї зібраної інформації для ідентифікації доменів з ознаками фішингу. У випадку підтвердження підозр, система ініціює створення попереджень або виконує блокування доступу до підозрілого домену, надаючи захист від можливих фішингових атак.

Переваги методу глибинного аналізу доменних імен:

– раннє виявлення фішингу. Метод дозволяє ідентифікувати потенційно шкідливі домени на ранніх стадіях, навіть до того, як вони будуть активно використані для фішингових атак, завдяки детальному вивченню реєстраційних даних і історії домену;

– виявлення складних шахрайських схем. Аналіз дозволяє розпізнати складні шахрайські схеми, які можуть бути неочевидні при поверхневому огляді, наприклад, через орфографічні помилки або легкі зміни в доменних іменах;

– покращення безпеки користувачів. Використання методу може допомогти запобігти доступу користувачів до фішингових сайтів, підвищуючи загальний рівень кібербезпеки та захисту даних;

– аналіз безпеки з'єднань. Перевірка SSL-сертифікатів в рамках аналізу допомагає виявити використання неавтентичних або підроблених сертифікатів, що є додатковою ознакою фішингу.

Недоліки методу глибинного аналізу доменних імен:

– великий обсяг даних для обробки. Аналіз великої кількості доменних імен та їх атрибутів може вимагати значних обчислювальних ресурсів і часу для обробки.

– потенційні помилкові спрацьовування. Метод може призвести до помилкових позитивних результатів, коли легітимні домени помилково ідентифікуються як шкідливі через схожість імен або інші критерії;

– потреба у постійному оновленні. Шахраї постійно адаптують свої тактики, тому метод вимагає регулярних оновлень та адаптацій для забезпечення ефективності виявлення;

– залежність від якості даних. Ефективність аналізу значною мірою залежить від якості та актуальності інформації про доменні імена, що може бути обмеженою через неповні або застарілі дані [28].

Отже, метод глибинного аналізу доменних імен є потужним інструментом у виявленні фішингових атак, який базується на детальному вивченні доменних імен та їх атрибутів. Цей підхід дозволяє вчасно ідентифікувати потенційні загрози, аналізуючи реєстраційні дані, вік домену, SSL-сертифікати та інші пов’язані характеристики, що надає змогу розпізнавати навіть складні шахрайські схеми. Втім, цей метод супроводжується викликами, такими як необхідність обробки великих обсягів даних, ризик помилкових спрацьовувань, потреба у постійному оновленні системи та залежність від якості доступної інформації. Незважаючи на ці виклики, метод глибинного аналізу доменних імен залишається важливою складовою комплексної стратегії кібербезпеки, спрямованої на захист від фішингових атак..

З усіх обговорених підходів метод машинного навчання вирізняється завдяки своїй гнучкості в адаптації до еволюційних методик фішингу, здатності автоматично

обробляти значні масиви даних та розпізнавати складні поведінкові моделі, що є індикаторами шахрайства. Цей метод показує високий рівень точності та ефективності, пропонуючи всебічний підхід до ідентифікації фішингових атак, що робить його особливо цінним серед інших стратегій, з огляду на невпинне ускладнення фішингових схем

2.2 Обґрунтування використання технік машинного навчання для ідентифікації фішингових атак

Використання машинного навчання надає значні переваги для створення дієвих засобів протидії фішингу. Зокрема, важливо відзначити застосування спеціалізованих методів оптимізації та аналізу, що допомагають підвищити ефективність та точність виявлення шахрайських дій. Серед використовуваних підходів, алгоритми Бройдена-Флетчера-Гольдфарба-Шанно (БФГШ), стохастичний градієнтний спуск (СГС) та метод випадкового пошуку (МВП) відіграють ключову роль.

Метод Бройдена-Флетчера-Гольдфарба-Шанно

БФГШ, що належить до квазі-ньютонівських методів оптимізації, широко застосовується для вирішення задач нелінійної оптимізації в машинному навчанні. Цей метод фокусується на мінімізації функції втрат без необхідності обчислення других похідних, використовуючи апроксимацію гессіана на основі інформації про градієнти.

Алгоритм актуалізує параметри моделі, опираючись на дані про градієнт та апроксимоване наближення до оберненого гессіана, що забезпечує ефективне просування до оптимального рішення. БФГШ відзначається ефективністю в задачах, де функція втрат має гладку структуру та обмежену кількість локальних мінімумів, сприяючи точному налаштуванню моделей для досягнення найкращих результатів прогнозування.

Цей алгоритм знаходить ідеальний баланс між швидкістю збіжності та акуратністю оновлень параметрів, що робить його цінним інструментом для

оптимізації складних функцій втрат в рамках машинного навчання, забезпечуючи високу якість прогнозування.

На рис. 2.6 представлено схему алгоритму роботи методу БФГШ.

Рисунок 2.6 – Схема алгоритму БФГШ

Процес виконання методу БФГШ ініціюється з вибору стартової позиції у просторі параметрів та заданням первісного припущення оберненого гессіана, зазвичай представленого як ідентична матриця. На кожному етапі алгоритму визначається градієнт функції втрат у поточній точці, що вказує на напрям збільшення функції.

Використовуючи апроксимацію оберненого гессіана, метод встановлює оптимальний напрямок для зменшення функції втрат, після чого відбувається оновлення параметрів моделі, переміщуючи поточну точку у напрямку, протилежному градієнту. З кожним кроком БФГШ адаптує своє наближення оберненого гессіана, виходячи з нових даних про градієнт та зміни параметрів.

Метод продовжує процес ітерацій, поки не буде досягнуто умов зупинки, які можуть включати незначні зміни в значенні функції втрат або вичерпання ліміту

ітерацій. Цей підхід дозволяє БФГШ ефективно просуватися до мінімуму функції втрат, враховуючи не лише напрям зменшення значення функції, але й її кривизну.

Переваги методу БФГШ:

– ефективність у великих просторах параметрів. BFGS демонструє високу ефективність у задачах з великою кількістю параметрів, завдяки своїй здатності апроксимувати обернений гессіан і ефективно визначати напрямок для оптимізації;

– не потребує обчислення других похідних. На відміну від деяких інших методів оптимізації, БФГШ не вимагає обчислення других похідних (гессіана), що робить його менш обчислювально складним і швидшим;

– самоналаштування. Метод може автоматично налаштовувати крок оновлення параметрів на основі інформації про градієнт і наближення оберненого гессіана, що забезпечує гнучкість і підвищує шанси на досягнення оптимуму;

– широке застосування. БФГШ успішно застосовується в різноманітних областях машинного навчання, включаючи регресію, класифікацію та інші оптимізаційні задачі, де потрібно мінімізувати функцію втрат.

Недоліки методу БФГШ:

– обмежена ефективність для дуже великих задач. Попри гарні результати в багатьох ситуаціях, для дуже великих задач (де кількість параметрів йде на мільйони) БФГШ може бути менш ефективним через великі вимоги до пам'яті;

– необхідність зберігання оберненого гессіана. Метод вимагає зберігання матриці, що апроксимує обернений гессіан, що може бути ресурсно витратно у термінах пам'яті, особливо для великих моделей;

– потенційні проблеми зі збіжністю. У деяких випадках БФГШ може не знайти оптимальне рішення або збіжність може бути повільною, особливо якщо функція втрат має складну топологію;

– початкова точка. Ефективність методу може значно залежати від вибору початкової точки. Невдало обрана початкова точка може призвести до повільної збіжності або збіжності до локального мінімуму [29].

Отже, метод БФГШ представляє собою потужний інструмент для оптимізації в машинному навчанні, який ефективно застосовується для вирішення широкого

спектру задач, зокрема мінімізації функцій втрат. Він вирізняється здатністю апроксимувати обернений гессіан без необхідності обчислення других похідних, що забезпечує ефективність і швидкість обчислень, особливо у великомасштабних задачах з значною кількістю параметрів. БФГШ автоматично адаптує кроки оновлення, спираючись на інформацію про градієнти, що робить його гнучким і дозволяє досягати високої точності оптимізації. Однак, метод має обмеження, зокрема високі вимоги до обчислювальних ресурсів і пам'яті для зберігання апроксимованого оберненого гессіана, що може обмежувати його застосування для дуже великих датасетів. Також, ефективність BFGS може залежати від вибору початкової точки та існує ризик збіжності до локальних мінімумів у випадку складних топологій функції втрат. Незважаючи на ці недоліки, BFGS залишається важливим і широко використовуваним інструментом в оптимізації машинного навчання, завдяки своїй здатності до самоналаштування та широкому спектру застосування.

Метод стохастичного градієнтного спуску

СГС займає ключове місце серед алгоритмів оптимізації в сфері машинного та глибокого навчання. Він заснований на ітеративному підході до оновлення параметрів моделі з метою зниження значення функції втрат. На відміну від традиційного градієнтного спуску, що розраховує градієнт за весь датасет перед оновленням параметрів, СГС вносить корективи, використовуючи лише один елемент даних або міні-пакет, що робить його більш гнучким і швидким.

Важливим аспектом СГС є регулювання швидкості навчання, яка впливає на розмір кроків оновлення параметрів на основі отриманої помилки за кожен крок. Правильний вибір швидкості навчання критично важливий, адже великий крок може призвести до проблем зі збіжністю, тоді як малий крок сповільнює процес навчання.

СГС демонструє високу ефективність у тренуванні різноманітних моделей, особливо у ситуаціях з об'ємними даними. Його перевага полягає у можливості швидкого оновлення параметрів на основі обмеженої кількості даних, що робить процес навчання менш ресурсозатратним і більш оперативним.

Використання СГС особливо актуальне для великих датасетів, оскільки алгоритм дозволяє вносити корективи в модель, базуючись на обробці лише частини

даних за раз. Це забезпечує ефективне та швидке навчання моделей, як ілюструється на рис. 2.7.

Рисунок 2.7 – Схема алгоритму СГС

Процедура застосування СГС ініціюється з початкового встановлення параметрів моделі, що часто здійснюється на основі випадкового вибору або згідно з певними заздалегідь заданими правилами. На кожному етапі процесу обирається окремий елемент або невелика група (міні-пакет) тренувальних даних на випадковій основі, що дозволяє зробити процес оновлення параметрів ефективнішим і зменшити розкид у оновленнях.

Потім, для вибраного елемента або міні-пакету розраховується градієнт функції втрат на основі поточних значень параметрів моделі, де градієнт вказує на напрям найшвидшого збільшення функції втрат. Використовуючи цей градієнт, параметри моделі оновлюються у протилежному напрямку, шляхом віднімання добутку швидкості навчання на градієнт від існуючих параметрів.

Оновлення параметрів відбувається ітеративно на протязі всього процесу навчання до досягнення визначених умов зупинки, як-от певна кількість ітерацій, мінімальна зміна в значенні функції втрат, або досягнення задовільної точності моделі. Це дозволяє моделі ефективно адаптуватися та зменшувати втрати, покращуючи свою продуктивність на основі аналізу тренувальних даних.

Переваги методу СГС:

– швидкість оновлень. Завдяки оновленню параметрів моделі після кожного тренувального прикладу або міні-пакету, СГС може швидко адаптуватися до даних, забезпечуючи значне прискорення навчання порівняно з повним градієнтним спуском;

– ефективність з великими датасетами. Метод є особливо ефективним при роботі з великими обсягами даних, оскільки не вимагає обчислення градієнтів за весь датасет для кожного оновлення, зменшуючи обчислювальні витрати;

– здатність до виявлення глобального мінімуму. СГС має потенціал вийти з локальних мінімумів завдяки своїй стохастичності, що може допомогти знайти більш оптимальне рішення;

– гнучкість налаштування. Швидкість навчання може бути адаптована протягом процесу навчання, дозволяючи методу ефективніше збігатися до оптимуму.

Недоліки методу стохастичного градієнтного спуску:

– варіативність оновлень. Стохастичний характер методу може призвести до великої варіативності в оновленнях параметрів, що іноді ускладнює збіжність до оптимуму;

– потреба в тонкому налаштуванні. Вибір оптимальної швидкості навчання є критичним і може вимагати значних зусиль для тонкого налаштування, щоб забезпечити ефективне навчання;

– ризик нестабільності. Занадто велика швидкість навчання може призвести до нестабільності і навіть розбіжності процесу навчання, тоді як занадто мала швидкість сповільнює процес;

– залежність від ініціалізації. Початкова ініціалізація параметрів може значно вплинути на ефективність та швидкість збіжності методу, роблячи важливим правильний вибір початкових значень [30].

Отже, метод СГС представляє собою фундаментальний інструмент в оптимізації для машинного та глибокого навчання, що відрізняється своєю спроможністю швидко адаптуватися та оновлювати параметри моделі в процесі навчання. Цей метод демонструє особливу ефективність у роботі з великими датасетами, забезпечуючи можливість швидшого навчання за рахунок ітеративного

оновлення параметрів на основі окремих тренувальних прикладів або міні-пакетів. СГС здатен ефективно вийти з локальних мінімумів та сприяє знаходженню глобального мінімуму, хоча й вимагає уважного налаштування параметрів, зокрема швидкості навчання, для забезпечення стабільності та оптимальної збіжності. Незважаючи на свою варіативність та залежність від ініціалізації, СГС залишається ключовим вибором для багатьох задач машинного навчання, завдяки своїй гнучкості та широкому спектру застосування.

Метод випадкового пошуку

МВП служить ефективним засобом для настроювання гіперпараметрів у машинному навчанні, пропонуючи альтернативний підхід до більш класичних методів оптимізації завдяки своїй стохастичній природі. Основна ідея полягає у випадковому виборі різних наборів гіперпараметрів із визначеного діапазону пошуку та їх подальшій оцінці згідно із визначеною функцією втрат або показником продуктивності.

Наприклад, при налаштуванні гіперпараметрів для нейронної мережі, метод випадково обирає різноманітні конфігурації, такі як швидкість навчання, число шарів, кількість нейронів у кожному шарі, та інше, а потім оцінює, як такі налаштування впливають на ефективність мережі.

Ця стратегія відома своєю спроможністю швидко покривати велику частину простору пошуку, що дозволяє виявити робочі рішення зі значно меншими обчислювальними зусиллями порівняно з більш методичними підходами, як-от сітковий пошук. Ефективність методу випадкового пошуку ілюстрована на рис. 2.8, демонструючи його здатність до відшукання оптимальних рішень у великому просторі гіперпараметрів.

Процедура методу випадкового пошуку, який застосовується для фініксування гіперпараметрів у машинному навчанні, ініціюється з визначення діапазону гіперпараметрів, які підлягають оптимізації. Це може включати налаштування, як-от швидкість навчання, розмір шарів нейронної мережі, типи оптимізаторів тощо. Далі, алгоритм вибирає на кожному етапі випадкові значення для кожного з гіперпараметрів із їх заданих діапазонів.

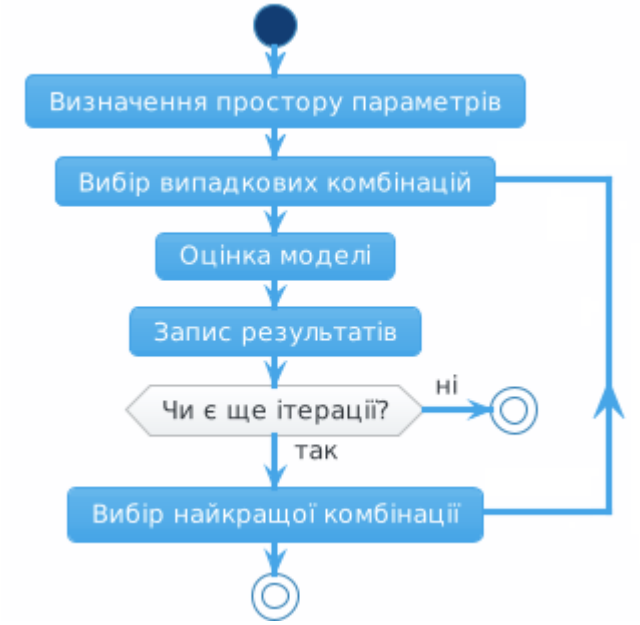

Рисунок 2.8 – Схема алгоритму МВП

Обрані комбінації гіперпараметрів використовуються для навчання моделі, а її ефективність визначається на основі заданої метрики або функції втрат, як, наприклад, точність на тестових даних. Результати кожної оцінки реєструються для подальшого аналізу. Цей цикл випадкового вибору та оцінки продовжується певну кількість ітерацій або доки не буде досягнуто ліміту обчислювальних можливостей.

На завершення, найкращий набір гіперпараметрів, що продемонстрував оптимальну продуктивність, вибирається з усіх протестованих комбінацій. Такий підхід дозволяє економно та ефективно проходити через широкий простір потенційних налаштувань моделі, забезпечуючи високі шанси на виявлення ефективних конфігурацій з порівняно низькими витратами ресурсів, на відміну від методів, які застосовують більш систематичний підхід до пошуку [31].

Переваги МВП:

– широкий огляд простору гіперпараметрів. Випадковий пошук дозволяє охопити широкий діапазон простору гіперпараметрів, забезпечуючи можливість виявлення ефективних конфігурацій, які могли б залишитися непоміченими при використанні більш систематичних методів;

– ефективність у великих просторах. Завдяки своїй стохастичній природі, метод випадкового пошуку часто знаходить задовільні рішення швидше, ніж методи, які вимагають систематичного перебору всіх можливих комбінацій;

– простота реалізації. Випадковий пошук не вимагає складної логіки або великих обчислювальних ресурсів для реалізації, роблячи його доступним навіть для початківців у машинному навчанні;

– гнучкість. Метод легко адаптується під різноманітні задачі оптимізації, дозволяючи досліджувати гіперпараметри різних типів моделей, від нейронних мереж до ансамблів дерев рішень.

Недоліки методу випадкового пошуку:

– відсутність гарантії оптимальності. Оскільки метод базується на випадковості, він не гарантує знаходження абсолютно найкращого набору гіперпараметрів;

– неефективність у деяких сценаріях. Для дуже малих або дуже специфічних просторів пошуку метод може бути менш ефективним порівняно з цілеспрямованими методами оптимізації;

– потенційна велика кількість ітерацій. У деяких випадках, для досягнення задовільного рішення може знадобитися велика кількість ітерацій, особливо якщо простір гіперпараметрів є великим;

– випадковість у виборі. Стохастичний характер вибору може призвести до пропуску ефективних рішень, особливо в тих випадках, коли оптимальні гіперпараметри знаходяться у вузьких областях простору пошуку [32].

Отже, метод випадкового пошуку в оптимізації гіперпараметрів машинного навчання пропонує простий і гнучкий підхід до пошуку ефективних конфігурацій моделей. Завдяки своїй спроможності швидко оглядати широкий простір гіперпараметрів і знаходити задовільні рішення з порівняно низькими обчислювальними витратами, метод є цінним інструментом, особливо при роботі з великими даними та складними моделями. Хоча випадковий пошук не гарантує знаходження абсолютного оптимуму і може вимагати значної кількості ітерацій для досягнення задовільного результату, його простота в реалізації та здатність

ефективно охопити різноманітність простору гіперпараметрів роблять цей метод корисним доповненням до інструментарію спеціалістів у галузі машинного навчання.

У табл. 2.1 проведено порівняльний аналіз розглянутих методів машинного навчання.

Таблиця 2.1 – Порівняльний аналіз методів машинного навчання

Характеристика	БФГШ	СГС	МВП
Точність оптимізації	Висока (завдяки точному обчисленню градієнтів та оберненого гессіана)	Середня (може варіюватися в залежності від швидкості навчання)	Залежить від випадковості (може бути менш точним)
Збіжність у складних оптимізаційних ландшафтах	Добра (ефективний при позитивно визначеному гессіані)	Може бути обмежена (через стохастичність оновлень)	Може бути обмежена (не завжди знаходить оптимальне рішення)
Потреба у обчисленні градієнтів	Так (важливий для оновлення параметрів)	Так (але обчислюється лише для підмножини даних)	Не потрібні (фокус на випадковому виборі гіперпараметрів)
Застосування до середніх і малих задач	Підходить (ефективний у випадках з меншою кількістю параметрів)	Менш підходить (більше підходить для великих датасетів)	Менш підходить (оптимізує лише гіперпараметри)

Вибір методу БФГШ для задач протидії соціальному інженірингу може бути виправданий на основі його ключових переваг, виділених у аналітичному огляді. Перш за все, висока точність БФГШ у визначенні оптимальних параметрів моделі через точне врахування градієнтів та апроксимацію оберненого гессіана робить цей метод цінним для точного прогнозування потенційних фішингових атак.

Додатково, БФГШ відзначається хорошою збіжністю у складних задачах оптимізації, що є особливо важливим у випадках виявлення фішингу, де моделі можуть зіткнутися зі складними патернами даних. Ефективність методу у випадках з позитивно визначеним гессіаном робить його надійним варіантом для моделей, що потребують детального налаштування гіперпараметрів.

Незважаючи на потребу в обчисленні градієнтів, що є критичним для адекватного оновлення параметрів моделі, цей аспект сприяє досягненню високої точності у прогнозуванні. БФГШ оптимально підходить для застосування в задачах середньої та малої масштабності, що може бути корисним у ситуаціях з обмеженими обчислювальними ресурсами або коли необхідна модель не вимагає великої кількості параметрів для налаштування.

Отже, з огляду на наведені аргументи, БФГШ представляє собою вагомий вибір для оптимізації моделей в рамках виявлення фішингових атак, забезпечуючи ефективне поєднання точності, виконавчої ефективності та практичності.

2.3 Опис методики проведення досліджень

Аналіз та розробка заходів протидії соціальному інженірингу в контексті інформаційних систем вимагає інтегрованого підходу, який охоплює як технічні, так і соціальні фактори безпеки. Це включає не лише впровадження передових технічних рішень для виявлення та блокування спроб шахрайства, але й залучення знань про поведінку та мотивацію людей, щоб ефективно протистояти маніпулятивним тактикам атакуючих. Ключовою задачею в цій сфері є створення комплексу засобів, які включають алгоритми ідентифікації потенційних загроз, методи попередження користувачів про ризики та стратегії швидкого реагування на інциденти соціального інженірингу. Такий підхід передбачає розробку набору інструментів, що забезпечують комплексну захист від різноманітних форм соціально-інженерних атак, враховуючи сучасні виклики та загрози.

Формулювання проблеми дослідження

Проблематика дослідження зосереджується на створенні ефективної і надійної системи, здатної ідентифікувати фішингові атаки в режимі реального часу. Необхідно розробити алгоритм, який може відокремлювати автентичні сайти від фішингових, з урахуванням різноманітності та складності сучасних методик соціального інженірингу. Дослідження має на меті:

– ідентифікацію ознак фішингу. Встановлення ключових параметрів, що дозволяють розпізнавати фішингові веб-сайти, включаючи аналіз доменних імен, присутність IP-адрес у URL, специфічні символи в URL, структуру URL, SSL-сертифікати, DNS-записи, а також ознаки веб-трафіку та інші технічні характеристики, які можуть свідчити про небезпеку;

– розробку дієвої моделі. Створення моделі машинного навчання, здатної ефективно визначати фішингові атаки на основі виявлених ознак;

– оптимізацію моделі. Застосування методу Бройдена-Флетчера-Гольдфарба-Шанно для точної та ефективної настройки моделі, що забезпечує її оптимальну працездатність;

– реалізацію системи в реальному часі. Гарантування, що розроблена система виявлення фішингу може бути застосована безпосередньо в умовах реального часу.

Завдання дослідження полягає у створенні системи виявлення фішингу, яка була б здатна швидко та точно реагувати на атаки, адаптуючись до непередбачуваних змін у тактиках, які застосовуються кіберзлочинцями у сфері соціального інжинірингу.

Методологія

Для аналізу технологій боротьби з соціальним інжинірингом обрана методологія ґрунтується на аналізі даних з датасету, отриманого з Kaggle, що включає інформацію про автентичні та фішингові веб-сайти. Ці дані слугуватимуть фундаментом для навчання та вдосконалення моделей машинного навчання, спрямованих на виявлення дискримінувальних характеристик між законними та шахрайськими сайтами.

Методологія включає застосування оптимізаційного методу БФГШ, що допоможе ефективно адаптувати моделі до особливостей набору даних, гарантуючи їх високу відповідність та продуктивність у задачах ідентифікації фішингових атак. Акцент робиться на аналізі властивостей URL, таких як присутність IP-адрес у назвах, структура URL, використання перенаправлень, HTTPS та інші технічні характеристики, що можуть свідчити про маніпулятивні наміри.

Процедура охоплює збір інформації з вказаного датасету, її попередню обробку, тренування моделей за допомогою БФГШ та оцінку їхньої продуктивності. Цей підхід не лише сприятиме створенню надійної системи для розпізнавання фішингових сайтів, але й допоможе глибше зрозуміти, які саме характеристики сайтів часто використовуються шахраями для обману користувачів. Така методологія забезпечить всебічний підхід до дослідження феномену фішингу та розробки технічно ефективних заходів протидії.

Обробка даних

Процес аналізу та трактування даних з набору даних фішингових та автентичних веб-сайтів охоплює декілька важливих кроків. Спочатку здійснюється підготовка даних за допомогою вдосконалених інструментів програмування, таких як ML.NET і C#, що дозволяє виявляти закономірності та взаємозв'язки між характеристиками сайтів та їх класифікацією як фішингових.

На стадії попередньої обробки застосовуються статистичні методи для оцінки важливості характеристик, як-от URL довжина, присутність IP-адреси в URL, застосування перенаправлень тощо, дозволяючи визначити їхній вплив на вірогідність фішингу. Проводиться оцінка розподілу цих ознак між фішинговими та легітимними сайтами для виявлення ключових відмінностей та загальних закономірностей. Далі, використовуючи алгоритми машинного навчання, в тому числі метод БФГШ для оптимізації, розробляються прогностичні моделі, які оцінюються за їхньою здатністю акуратно розділяти сайти на фішингові та легітимні на основі аналізованих ознак.

Завершальний крок включає глибоке трактування отриманих результатів, з акцентом на визначення ознак, що найбільше впливають на класифікацію, та їхнє потенційне використання для підвищення рівня безпеки. Цей аналіз допоможе не лише оцінити ефективність створеної моделі, але й визначити напрямки для розвитку майбутніх заходів протидії фішинговим атакам.

Оцінка результативності застосованих методів та стратегій протидії фішингу буде проводитися за допомогою визначених метрик ефективності:

– площа під ROC-кривою. Ця метрика демонструє спроможність моделі диференціювати класи (фішингові проти легітимних сайтів), де більша площа під кривою свідчить про вищу ефективність розрізнення;

– F1-оцінка. Вона вимірює баланс між точністю (процент правильно ідентифікованих фішингових сайтів) та повнотою (процент усіх фішингових сайтів, що були виявлені моделлю), виражаючи це як гармонійне середнє. Висока F1-оцінка підкреслює ефективний баланс між цими двома показниками;

– точність. Вказує на частку сайтів, правильно класифікованих як фішингові з усіх, що модель визначила як такі, демонструючи здатність моделі уникати помилково позитивних висновків;

– повнота. Ця метрика відображає відсоток фішингових сайтів, які модель змогла виявити з загальної кількості фішингових сайтів у наборі даних, вказуючи на ефективність моделі в ідентифікації загроз.

Ці показники надають детальну інформацію про здібності моделі точно виявляти фішингові сайти, при цьому мінімізуючи кількість помилок і забезпечуючи високий захист в інформаційному середовищі. Вони також допоможуть у визначенні потенційних ризиків і користі від застосування розробленої моделі в практиці захисту інформації.

На підставі аналізу отриманих даних будуть сформульовані висновки та рекомендації для вдосконалення заходів боротьби з соціальним інженірингом. Це може включати розробку нових стратегій безпеки, технологій, поліпшення навчальних програм або впровадження нових процедур для зміцнення інформаційної безпеки.

Висновки за розділом 2

У рамках даного розділу було проведено глибокий аналіз різноманітних методів і технік для ідентифікації та протидії фішинговим атакам, що включало алгоритми методу чорного списку, аналізу змісту веб-сторінок, машинного навчання, поведінкового аналізу та глибинного аналізу доменних імен. Особлива увага була

приділена вивченню та обґрунтуванню застосування алгоритмів машинного навчання, зокрема методів Бройдена-Флетчера-Гольдфарба-Шанно, стохастичного градієнтного спуску та методу випадкового пошуку. В результаті аналізу було обрано метод БФГШ як найбільш перспективний для розробки ефективної стратегії протидії соціальному інжинірингу.

Ретельно було сформульовано проблему дослідження, визначено ключові методологічні підходи та описано процедуру обробки даних, яка включала оцінку за допомогою критично важливих метрик, таких як площа під ROC-кривою, F1-оцінка, точність та повнота. Ці метрики були використані для оцінювання ефективності моделей у розрізненні між фішинговими та легітимними веб-сайтами, забезпечуючи грунтовну базу для подальшої валідації обраних підходів.

Отримані результати вказують на значний потенціал методу БФГШ у точній ідентифікації фішингових атак, а також на важливість комплексного врахування різних ознак веб-сайтів. Ефективність застосованих моделей та їхній вплив на рівень інформаційної безпеки становлять фундамент для розробки нових стратегій, політик та навчальних програм, спрямованих на підвищення обізнаності та захист від соціального інжинірингу. Результати дослідження закладають фундамент для розробки програмного рішення для протидії фішингу, що відповідають сучасним викликам інформаційної безпеки.

РОЗДІЛ 3
РОЗРОБКА ТА ВЕРИФІКАЦІЯ МЕТОДИКИ ПРОТИДІЇ

3.1 Формування математичної бази для розпізнавання фішингу

Процес створення моделі охоплює кілька етапів: збір і аналіз даних, вибір характеристик, що найточніше описують фішингові атаки, та застосування методу БФГШ для оптимального налаштування класифікатора. Завдяки цьому алгоритму можна ефективно обробляти великі обсяги даних і складні моделі поведінки, типові для сучасних фішингових атак.

Проблема, яку треба вирішити, полягає у бінарній класифікації даних (X,Y), де $X \in R^{(n \times m)}$ представляє собою матрицю характеристик, а $Y \in \{0,1\}^n$ — вектор міток. У даному контексті, n вказує на кількість прикладів у датасеті, тоді як m=18 означає кількість характеристик, які включають наявність IP, символ «@», довжина URL, глибина URL, переадресація, HTTPS у домені, скорочений URL, префікс/суфікс, запис DNS, веб-трафік, вік домену, кінець домену, iFrame, Mouse Over, правий клік, веб-переадресації.

На основі цих вхідних даних структурується клас PhishingData, що містить поля, як-от назва домену веб-сайту, наявність IP-адреси в URL замість доменного імені, наявність символу '@' в URL, довжина URL, глибина URL, вказана кількістю слешів, наявність переадресацій, використання HTTPS у доменній назві, використання скороченого URL, наявність незвичайних префіксів або суфіксів, наявність запису DNS, рейтинг веб-трафіку, вік домену, час до закінчення реєстрації домену, використання iFrame тегів, використання скриптів для зміни поведінки курсору миші, дозвіл на використання правого кліку миші, та наявність автоматичних переадресацій на сайті, а також мітка, що вказує, чи вважається сайт фішинговим (1) або легітимним (0).

Функція втрат логістичної регресії може бути виражена наступним чином:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right], \quad (3.1)$$

де $h_\theta(x)$ представляє собою сигмоїдну функцію, а θ є вектором параметрів моделі.

Матриця X складається з екземплярів об'єкту PhishingData, де кожен рядок представляє окремий екземпляр «PhishingData», а стовпці кореспондують до атрибутів, таких як Have_IP, Have_At, URL_Length, URL_Depth та інші. Таким чином, елемент x_{ij} в матриці X означає значення j-го атрибуту i-го екземпляру «PhishingData»:

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{pmatrix} \quad (3.2)$$

Аналогічно, вектор Y створюється з міток *Label* кожного екземпляру *PhishingData*:

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix}, \quad (3.3)$$

де y_i може бути 1 (атака) або 0 (не атака).

Після тренування моделі, прогнози $\hat{Y}$ отримуються за допомогою сигмоїдної функції $h_\theta(x)$:

$$\hat{Y} = h_\theta(x) = \frac{1}{1+\exp(-X\theta)} \quad (3.4)$$

Ці прогнозовані значення записуються в атрибут «*Prediction*» об'єктів класу *CoapPrediction*, де прогноз буде розглядатися як:

$$CoapPrediction.Prediction = \begin{cases} 1, \text{якщо } \hat{y}_i \geq 0{,}5 \\ 0, \text{якщо } \hat{y}_i < 0{,}5 \end{cases} \quad (3.5)$$

Цей підхід дозволяє не лише організовано викладати вхідні та вихідні дані, але також ефективно застосовувати їх для навчання та визначення прогнозів у моделі, що істотно збільшує точність і надійність системи для ідентифікації фішингових атак.

3.2 Збір та аналіз даних для тренування алгоритмів детектування фішингу

Ключовим кроком у створенні системи виявлення фішингу є адекватна підготовка і аналітичний огляд набору даних. Вибраний для цієї мети набір даних з Kaggle включає інформацію про легітимні та фішингові сайти. Перед застосуванням

цього набору для тренування алгоритму, необхідно здійснити декілька важливих процедур обробки:

– перевірка на відсутні дані є критичною, щоб виявити записи без деяких даних і вирішити, як із цим краще впоратися: видалити ці записи або заповнити відсутні значення, наприклад, середніми;

– аналіз балансу класів допоможе визначити, чи є розподіл між легітимними та фішинговими сайтами в датасеті рівномірним, що важливо для ефективного навчання моделі;

– нормалізація числових атрибутів необхідна для забезпечення того, щоб модель обробляла всі ознаки однаково, не віддаючи перевагу тим, що мають вищі значення.

Набір даних, завантажений з Kaggle, охоплює різноманітні атрибути веб-сайтів, що є вирішальними для ідентифікації моделей, які допомагають відрізняти легітимні ресурси від фішингових. Аналіз цих даних дозволяє висунути кілька ключових спостережень:

– загальні характеристики. У наборі даних переважають бінарні числові атрибути (значення 0 або 1), хоча атрибут, пов'язаний з глибиною URL (URL_Depth), виявляється з максимумом у 20 та середнім значенням приблизно 3;

– не виявлено відсутні дані. В аналізованому наборі даних не знайдено жодних прогалин у даних, що підкреслює його високу якість та придатність для машинного навчання без додаткової підготовки;баланс класів. Набір даних характеризується ідеальним балансом між мітками класів, з рівним розподілом 50% на 50% між легітимними та фішинговими сайтами, що забезпечує оптимальні умови для тренування моделей;

– розподіл атрибутів. Атрибути, такі як наявність IP або символу '@', переадресація, наявність HTTPS, використання скорочених URL і інші, переважно бінарні з переважанням значення 0, що може вказувати на їх нерівномірний вплив у моделі;

– особливості деяких атрибутів. Атрибути, як рейтинг веб-трафіку, вік домену та термін його дії, демонструють більшу варіативність, де, наприклад, рейтинг

веб-трафіку має значення, що можуть бути критичними для виявлення фішингових сайтів.

Набір даних виглядає добре структурованим і збалансованим для використання в задачах машинного навчання, зокрема у виявленні фішингових сайтів.

3.3 Розробка алгоритмів роботи методу протидії фішингом атакам

Розробка алгоритмічної структури для методу протидії соціальному інженірингу розглядається як складний і багатогранний науково-дослідницький процес. На рис. 3.1 показано блок-схему, яка демонструє процес навчання та зберігання моделі у системі.

Процес навчання моделі розпочинається з вибору відповідного файлу, який містить дані для тренування. Одразу ж здійснюється перевірка файлу на точність і повноту даних, після чого відбувається їх завантаження в систему. На наступному етапі відбувається сам процес тренування моделі, в ході якого машина «вчиться» на підготовленому датасеті. Результати тренування представляються для аналізу ефективності навченої моделі. Далі в систему вноситься інформація про параметри та конфігурацію моделі, яка підлягає ретельній перевірці на коректність. Після підтвердження правильності даних вони додаються до бази даних. Це дозволяє вести облік всіх навчених моделей і використовувати ці дані для подальшого поліпшення процесів тренування та аналізу. У кінці система надає зведену інформацію про всі існуючі моделі, що включає дані про їх ефективність та параметри.

Рисунок 3.1 – Алгоритм навчання та зберігання даних моделі

Рис. 3.2 ілюструє алгоритм процес взаємодії користувача із формою для виявлення фішингових сайтів на основі введеної користувачем інформації за допомогою моделі машинного навчання.

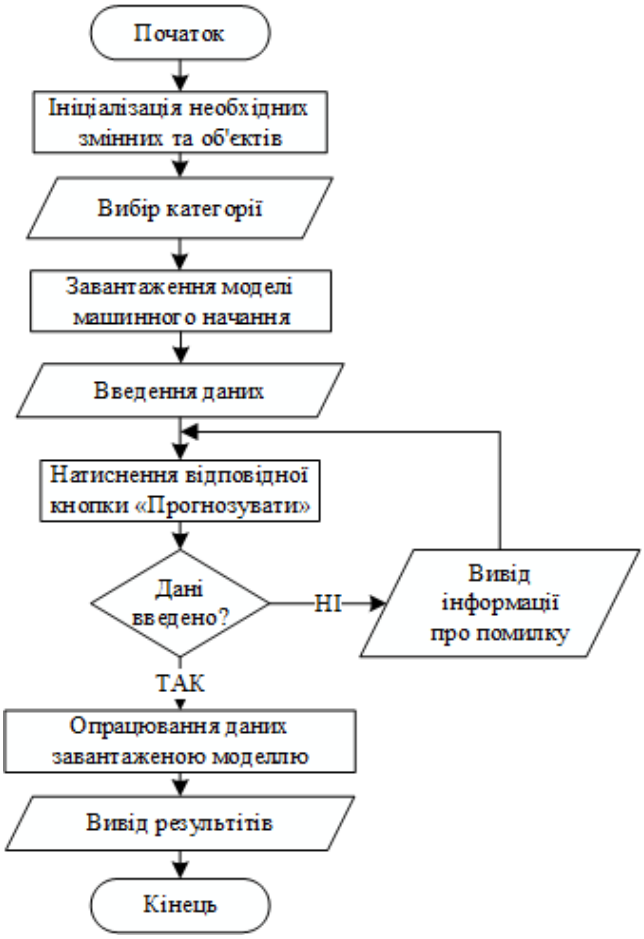

Рисунок 3.2 – Алгоритм процесу процес взаємодії користувача з моделлю

Процес прогнозування чи є веб-сайт фішинговим починається з ініціалізації необхідних змін та об'єктів. Після цього користувач вибирає категорії, які будуть використовуватися для аналізу, і вводить дані, що характеризують сайт. На основі введених даних машина виконує навчання моделі.

Коли модель навчена, користувач натискає візуальну кнопку «Прогнозувати», щоб ініціювати процес прогнозування. Якщо в процесі з'являється запитання «Дані введено коректно?», відбувається перевірка на вірність введення даних. Якщо дані були введені некоректно, користувач отримує відповідне повідомлення про помилку і має можливість ввести дані знову.

У випадку правильного введення даних відбувається організація даних та проведення аналізу за допомогою заздалегідь навченої моделі. По завершенні процесу аналізу на екран виводяться результати прогнозу, які вказують на те, чи є

сайт фішинговим. На заключному етапі процес прогнозування завершується і система готова до нового циклу аналізу.

3.4 Проєктування бази даних системи протидії фішингом атакам

Створення архітектури бази даних є вирішальним для успіху будь-якої інформаційної системи. Цей процес охоплює ретельне вироблення структури, де дані упорядковуються в таблиці, розроблені для оптимальної взаємодії. В ході дизайну бази даних визначаються категорії даних, які будуть представлені у таблицях, а також обираються відповідні типи даних для стовпців. Життєво важливим є встановлення ключів та інших обмежень, які гарантують цілісність і правильне співвідношення даних. Налагодження зв'язків між таблицями є ключовим для відображення комплексних даних і забезпечення ефективного керування та доступу до інформації.

Головна ціль проектування бази даних — створення міцної та продуктивної системи, яка забезпечує швидкий і зручний доступ до даних, а також їх обробку. Від акуратності проектування бази даних залежить якість виконання запитів і загальна продуктивність системи.

У базі даних присутні такі ключові сутності:

– сутність Users слугує для зберігання інформації про користувачів системи. Вона містить такі поля: унікальний ідентифікатор користувача (UsersId), ім'я (FirstName), прізвище (LastName), користувацьке ім'я (UsersName), пароль (UsersPassword), ідентифікатор ролі (RoleId), опис користувача (Description) та електронну пошту (Email). Первинний ключ таблиці — це UsersId;

– сутність Logs призначена для збору логів або записів подій, що відбуваються в системі. Включає поля: ідентифікатор запису (LogsId), ідентифікатор користувача, який пов'язаний з подією (UsersId), назва події (EventNameShow), дата події (EventDate). LogsId виступає як первинний ключ;

– сутність PhishingData використовується для зберігання даних, що характеризують веб-сайти з точки зору фішингу. Вона містить поля: унікальний ідентифікатор даних (PhishingDataId), доменне ім'я (Domain) та ряд характеристик

сайту, які включають наявність IP у URL (Have_IP), наявність символу '@' (Have_At), довжину URL (URL_Length), глибину URL (URL_Depth), наявність переадресацій (Redirection), використання HTTPS в домені (https_Domain), наявність скорочених URL (TinyURL), префікси та суфікси (Prefix_Suffix), наявність DNS-записів (DNS_Record), веб-трафік (Web_Traffic), вік домену (Domain_Age), час до закінчення дії домену (Domain_End), використання iFrame (iFrame), взаємодію з курсором миші (Mouse_Over), можливість використання правого кліку (Right_Click), автоматичні переадресації (Web_Forwards), мітку класифікації (Label) та ідентифікатор користувача, який додав дані (UsersId);

– сутність Neural містить дані про моделі, що використовуються у системі для машинного навчання. Вона включає поля: ідентифікатор моделі мережі (NeuralId), назву моделі (NeuralNames), ідентифікатор сценарію (ScenariosId), до якого вона відноситься, та файл моделі (NeuralFileModel);

– сутність Scenarios призначена для опису сценаріїв, які можуть використовувати моделі машинного навчання. Вона має поля: ідентифікатор сценарію (ScenariosId), назву сценарію (ScenariosName) та детальний опис (Description).

Завершивши процес визначення структури і взаємозв'язків інформаційної моделі, наступним етапом розробки стало її імплементація в реальну базу даних. У цьому контексті було обрано Microsoft SQL Server як платформу для реалізації, завдяки його ефективності, стабільності та зручності управління, що є критично важливим для складних та об'ємних систем. Щоб глибше зануритись в деталі створення бази даних, в Додатку А розміщено SQL-скрипти, які використовувались для її створення. Для наочності, в розділі, що містить рис. 3.3, представлена візуалізація фізичної структури бази даних.

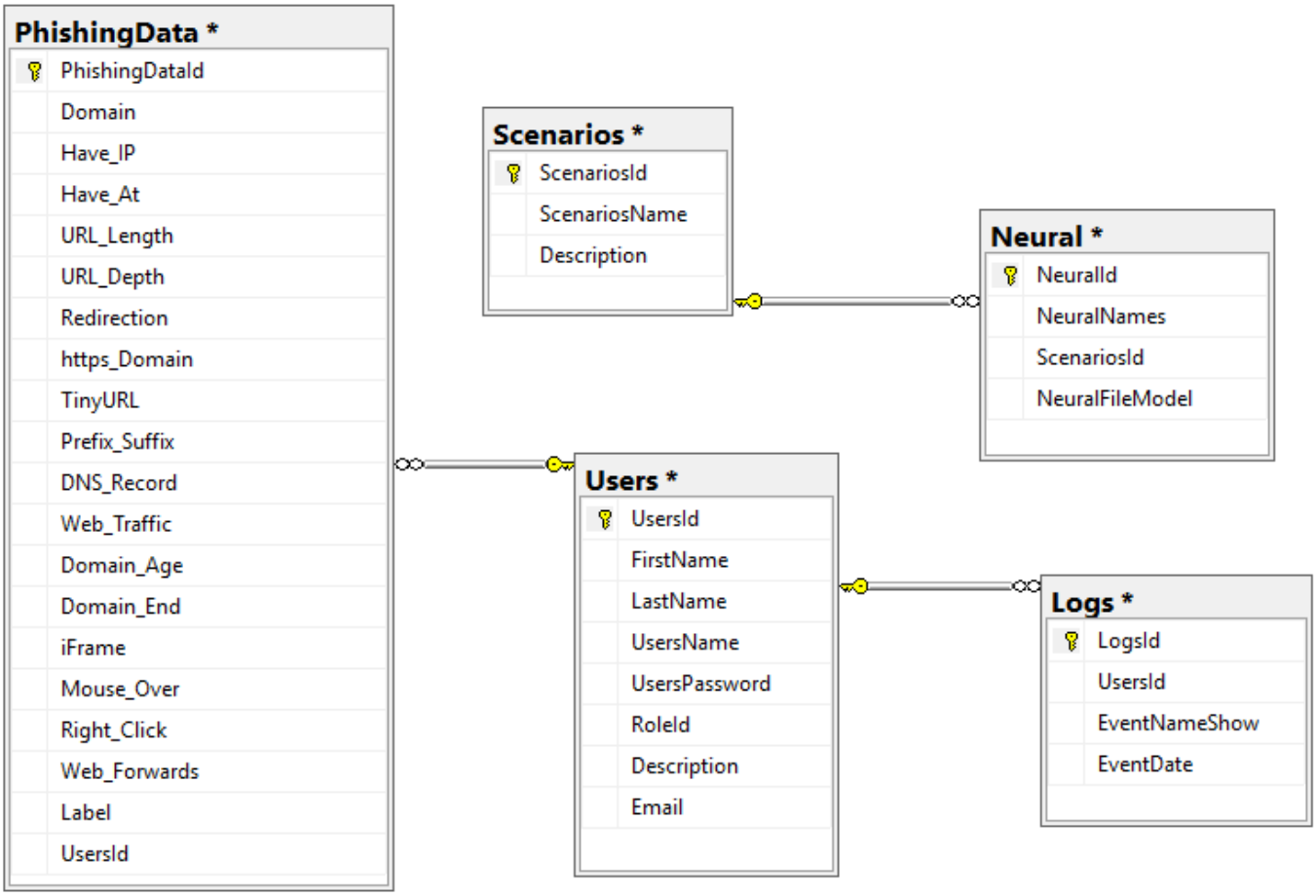

Рисунок 3.3 – Фізична модель бази даних

Візуалізація фізичної моделі дозволяє оцінити структуру та взаємозв'язки між різними сутностями в базі даних, підкреслюючи складність і організаційну структуру розробленої системи.

3.5 Розробка основних модулів системи протидії фішингом атакам

В процесі створення технологій для протистояння соціальній інженерії та ідентифікації фішингових веб-сайтів використовувалась трирівнева системна архітектура. Така багаторівнева структура сприяє оптимізації обробки інформації та інтерактивності з кінцевим користувачем, забезпечуючи при цьому еластичність та масштабованість. Система організована на наступні основні шари:

– шар доступу до даних, що займається управлінням інформації, необхідної для аналізу фішингу, включаючи класи для зв'язку з базою даних і операції CRUD, що підвищує модульність та спрощує технічне обслуговування;

– бізнес-логічний шар, який втілює ключові механізми для визначення фішингу, працюючи з даними для їх аналізу та класифікації за допомогою різних алгоритмів, що гарантує точність та надійність функціонування системи;

– шар представлення, що фокусується на візуалізації інформації для користувачів, відповідаючи за користувацькі інтерфейси та інші способи візуалізації результатів роботи системи, що дозволяє користувачам легко інтерпретувати результати аналізу та реагувати на них.

На початковому етапі було сформовано шар доступу до даних, що став основоположним елементом для роботи системи, надаючи інструменти для комунікації з базою даних, як показано на рис. 3.4. У рамках цього шару були створені спеціалізовані класи для кожної категорії даних, що використовуються в системі. Ці класи є важливими для взаємодії з таблицями бази даних і надають можливості виконувати стандартні операції CRUD для відповідних сегментів даних, як-от управління даними про метрики, журнали змін або результати аналітики.

Кожен з цих класів має інтерфейс, що окреслює набір методів для маніпулювання даними, що дозволяє, за потреби, гнучко змінювати або оновлювати логіку доступу до даних без впливу на інші частини системи. Це сприяє підтримці високого рівня ефективності системи та її здатності до масштабування.

На діаграмі рівня даних представлено основні компоненти, серед яких:

– клас LogsProvider, який відіграє роль центру збору та обробки логів діяльності системи. Завдяки методам запису, читання та аналізу логів, цей клас збирає критично важливі дані про взаємодії користувачів, системні події, збої та інше, що допомагає відслідковувати та досліджувати потенційні загрози;

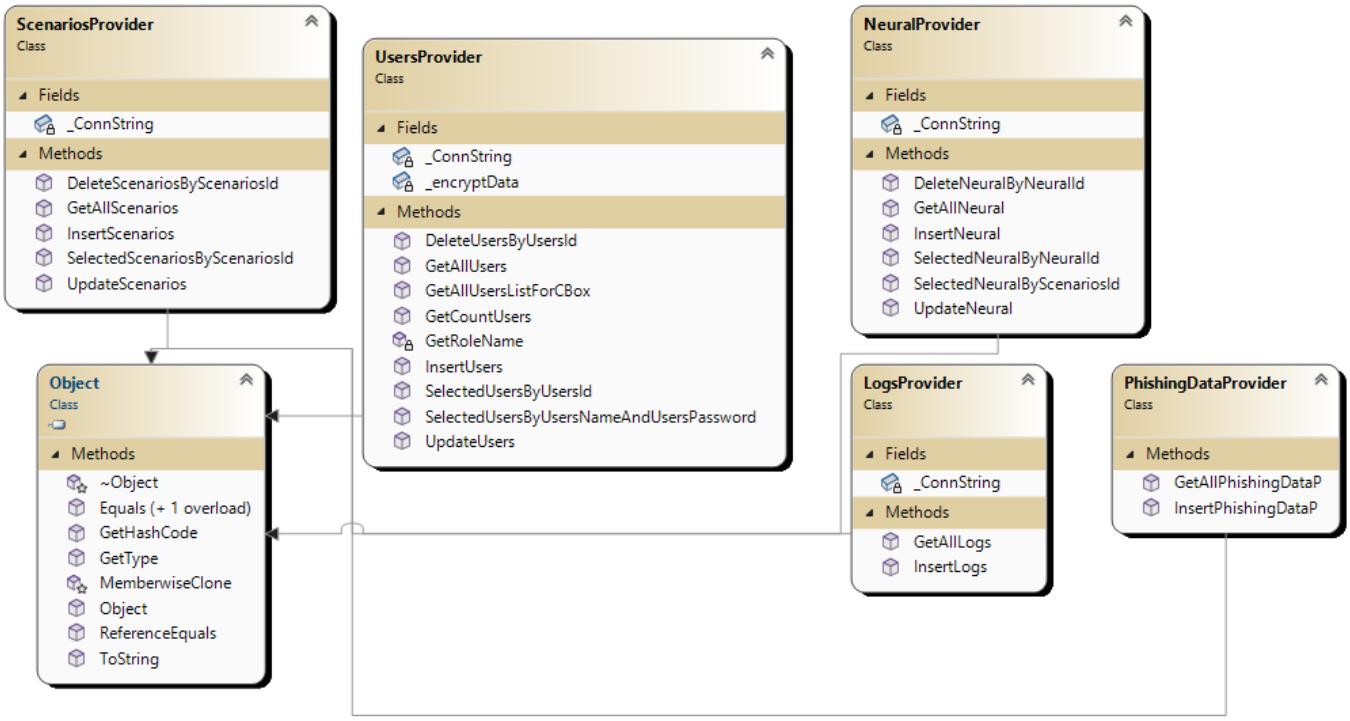

Рисунок 3.4 – Діаграма класів рівня даних

– клас NeuralProvider служить для втілення нейронних мереж та інших алгоритмів машинного навчання у систему. Він управляє зберіганням, опрацюванням та тренуванням моделей на дата сетах, орієнтованих на виявлення фішингу, і включає інструменти для оновлення та застосування цих моделей до прогнозування;

– клас ScenariosProvider контролює сценарії виявлення фішингу та інших форм соціального інжинірингу, включаючи аналітику поведінки та виявлення аномалій, з можливістю їх актуалізації для відповідності новітнім фішинговим методикам;

– клас UsersProvider забезпечує управління інформацією користувачів системи, охоплюючи збереження та обробку даних про їхні профілі, ролі та рівні доступу, що допомагає в підтримці безпеки та управлінні авторизацією та аутентифікацією.

Подальший етап розробки системи включав в себе створення бізнес-логіки, з якої було сформовано відповідну діаграму класів, представлену на рис. 3.5.

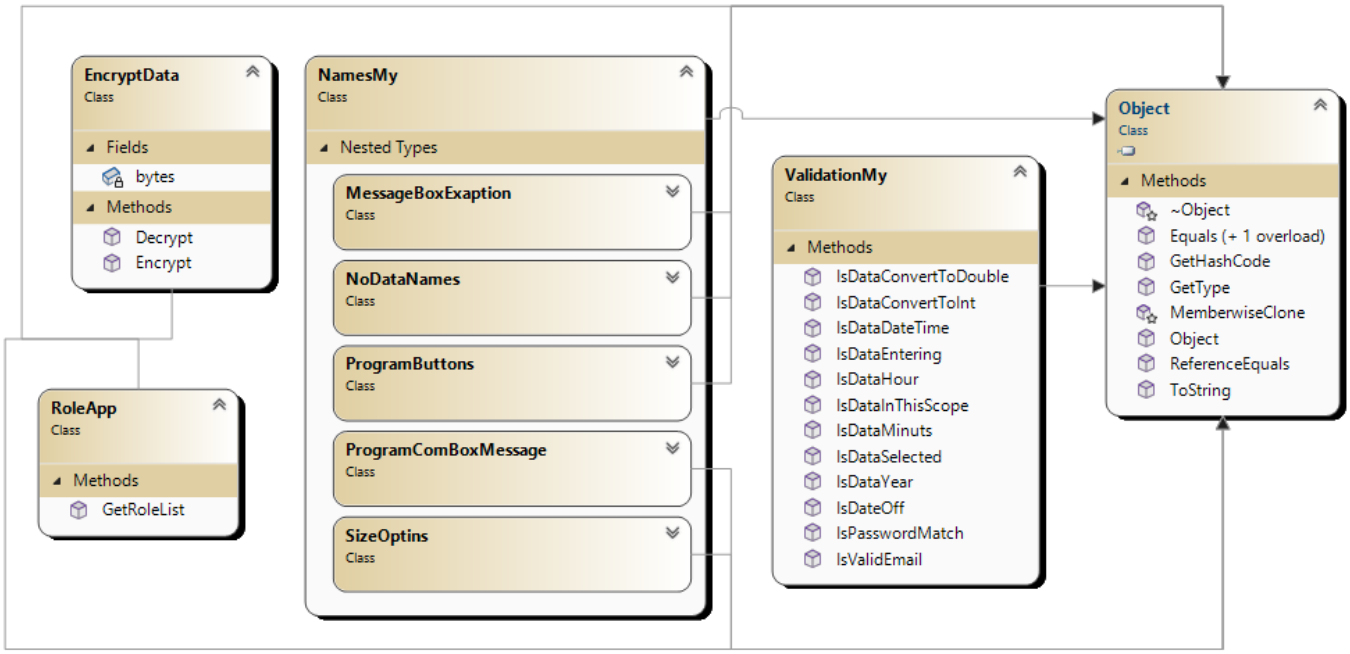

Рисунок 3.5 – Діаграма бізнес-логіки додатку

На цій діаграмі відображено ключові компоненти бізнес-логіки:

– клас EncryptData, який займається шифруванням інформації у системі, є фундаментальним для захисту конфіденційності та цілісності даних. В залежності від потреб безпеки, він може застосовувати різні методики шифрування;

– клас NamesMy призначений для адміністрування номенклатури, яка використовується всередині системи. Цей клас дозволяє створювати, зберігати та обробляти стандартизовану термінологію, забезпечуючи узгоджене використання бізнес-термінів.

– клас Role відіграє роль у визначенні та керуванні ролями користувачів системи, призначаючи рівні доступу відповідно до їх ролей, що є критичним для безпечної роботи системи.

– клас ValidationMy зосереджується на перевірці та підтвердженні валідності інформації, що вноситься або опрацьовується системою, забезпечуючи відповідність введених даних до заданих критеріїв та виявляючи помилки для забезпечення точності даних.

На рис. 3.6 представлено схематичне зображення, що відтворює структуру користувацького інтерфейсу та показує взаємодії між його складовими. Важливість

цієї діаграми полягає у наглядному представленні організації елементів інтерфейсу і шляхів їх спільної роботи. Вона виразно висвітлює компонування і функціонал елементів, таких як меню, кнопки, поля для введення даних та їх візуалізації, обумовлюючи механізми взаємодії користувача з програмою. Ця інформація забезпечує розробникам уявлення про те, як користувач буде взаємодіяти з системою, виконувати операції та доступати до інформації, що є ключем до створення зручного та зрозумілого користувацького інтерфейсу.

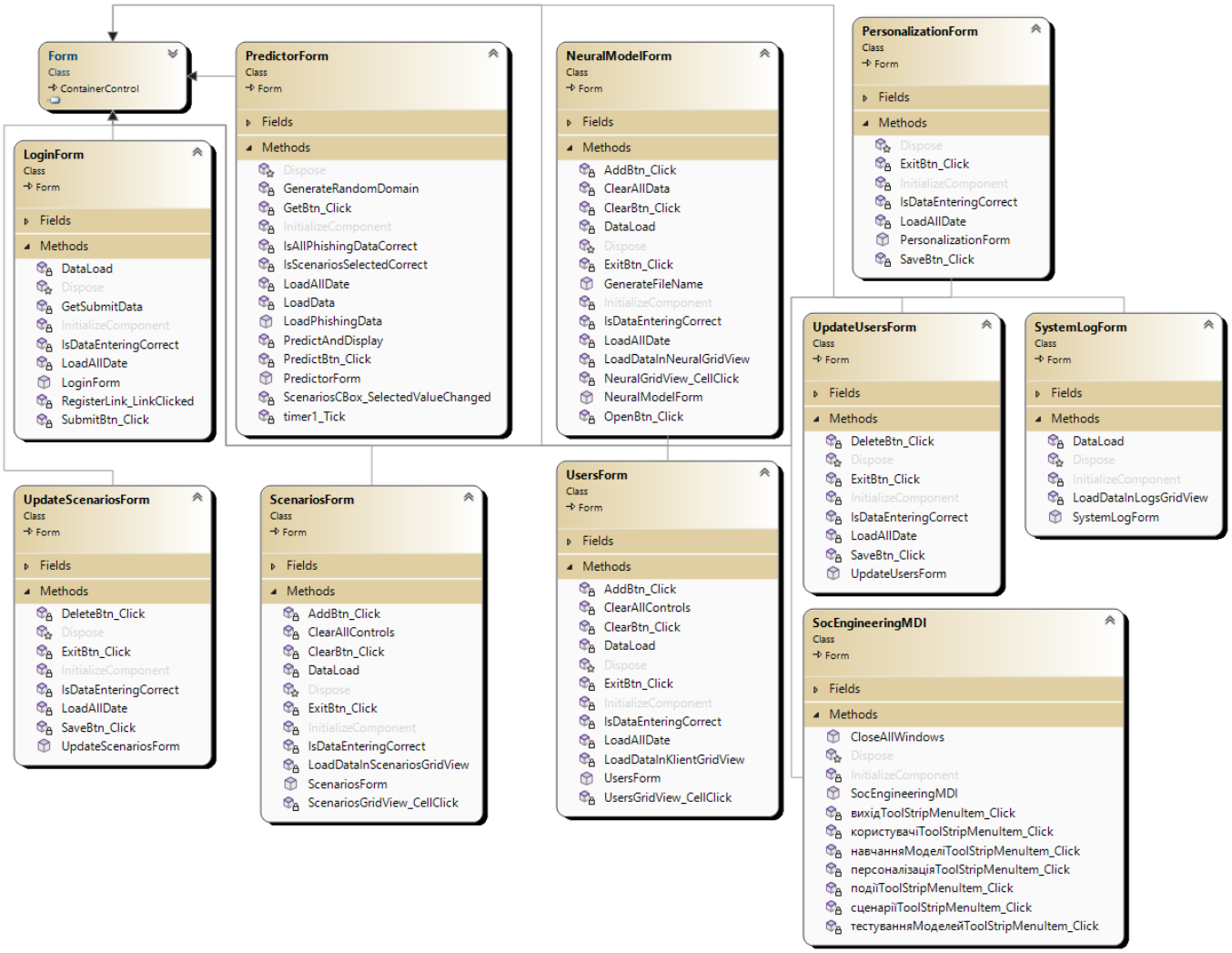

Рисунок 3.6 – Діаграма класів рівня користувацького інтерфейсу

Компоненти бізнес-логічного шару представлені наступними класами:

– клас SocEngineeringMDI, який функціонує як основне вікно системи, координує взаємодію між різноманітними формами та модулями, використовуючи

MDI-підхід для роботи з декількома документами всередині одного головного вікна та надає інструменти навігації по системі.

– клас PredictorForm, призначений для реалізації та візуалізації прогнозів щодо фішингових атак, дає можливість користувачам вибирати параметри для прогнозування та переглядати результати.

– клас NeuralModelForm забезпечує інтерфейс для керування та налаштування моделей машинного навчання, включаючи вибір параметрів моделі та перегляд результатів тренування і тестування.

– клас UpdateScenariosForm займається оновленням та модифікацією сценаріїв, які використовуються для ідентифікації фішингових атак, дозволяючи адаптувати систему до змінюваних загроз.

– клас ScenariosForm пропонує інструменти для перегляду та керування сценаріями, що використовуються для аналізу поведінки та виявлення фішингу.

– клас LoginForm є входом до системи, вітаючи користувачів при авторизації та забезпечуючи безпечний вхід за допомогою логіну та пароля.

– клас PersonalizationForm надає засоби для персоналізації налаштувань користувачем, дозволяючи настроювати індивідуальні преференції та інтерфейс.

– клас SystemLogForm відкриває доступ до системних журналів для моніторингу стану системи та виявлення помилок чи подій безпеки.

– клас UpdateUsersForm дає можливість оновлювати інформацію про користувачів, їх ролі та рівні доступу для забезпечення актуальності даних.

– клас UsersForm надає інструменти для перегляду та управління загальною інформацією про користувачів, забезпечуючи адміністрування доступу і контролю за користувацькою активністю.

В процесі розробки системи за допомогою Visual Studio 2022, критичним моментом стало інтегрування бази даних. Завершивши цей крок, увагу було сконцентровано на створенні бізнес-логіки, особливо на автоматизації управління документацією. Основним елементом цієї фази стало розроблення алгоритмів для взаємодії з метаданими документів, що є ключовим для системи управління документацією.

На першому етапі до файлу конфігурації App.config було додано змінну CONNECTING, яка містить параметри для з'єднання з базою даних, стаючи невід'ємною частиною архітектури системи. Кроки налаштування з'єднання детально представлено на рис. 3.7.

```
<!-- Підключення до бази даних -->
<appSettings>
  <add key="CONNECTING"
       value="Data Source=(LocalDB)\MSSQLLocalDB;
       AttachDbFilename=|DataDirectory|\DB.mdf;
       Integrated Security=True" />
</appSettings>
```

Рисунок 3.7 – Налаштування параметрів з’єднання з базою даних

У ході створення системи «Фішинг детектор» було обрано використання namespace System.Data.SqlClient зі стандартної бібліотеки .NET для забезпечення взаємодії з SQL Server. Простір імен System.Data.SqlClient спеціалізується на роботі з реляційними базами даних і включає класи для здійснення з'єднань та виконання SQL-запитів, що забезпечує надійну взаємодію з базою даних і знижує ризики помилок у обробці даних.

З погляду користувацького інтерфейсу, значним внеском у систему стало додавання компонента MenuStrip до основного вікна. MenuStrip дозволяє створювати візуально привабливі меню з легкою доступністю до важливих функцій системи, полегшуючи користувачам навігацію та доступ до потрібних інструментів, як показано на рис. 3.8.

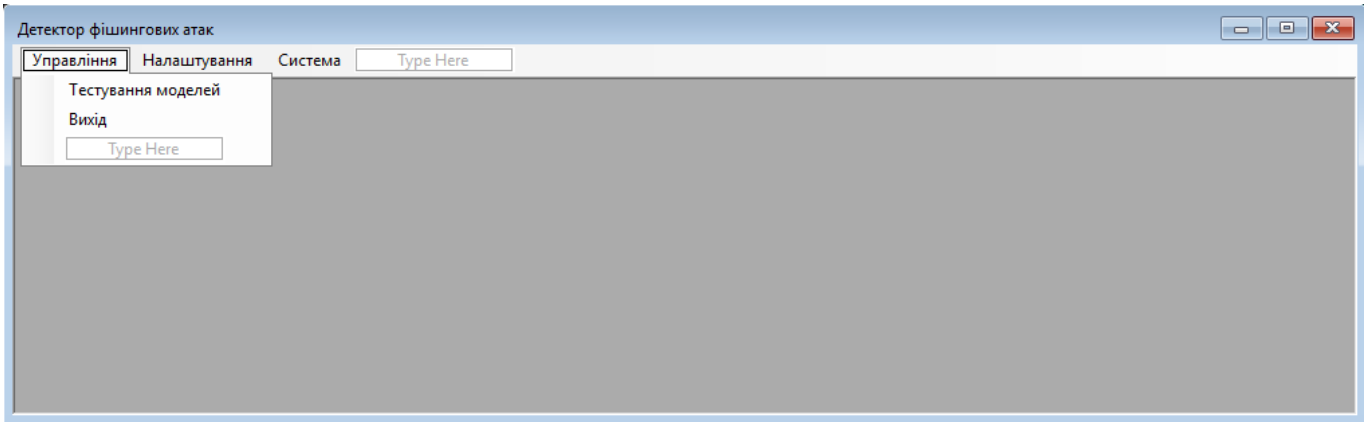

Рисунок 3.8 – Головне вікно системи

В системі була впроваджена реакція на вибір пунктів меню користувачами. Як зазначено на ілюстрації 3.9, вибір пункту «Тестування моделей» активує спеціалізований інструментарій для аналізу та тестування моделей машинного навчання.

```
1 reference
private void тестуванняМоделейToolStripMenuItem_Click(object sender, EventArgs e) {
    CloseAllWindows();
    PredictorForm predictorForm = new PredictorForm();
    predictorForm.MdiParent = this;
    predictorForm.WindowState = FormWindowState.Maximized;
    predictorForm.Show();
}
```

Рисунок 3.9 – Подія для виклику форми «Тестування моделей»

Рис 3.10 демонструє інтерфейс, розроблений для процесу тренування моделей машинного навчання, підкреслюючи його важливість для користувачів.

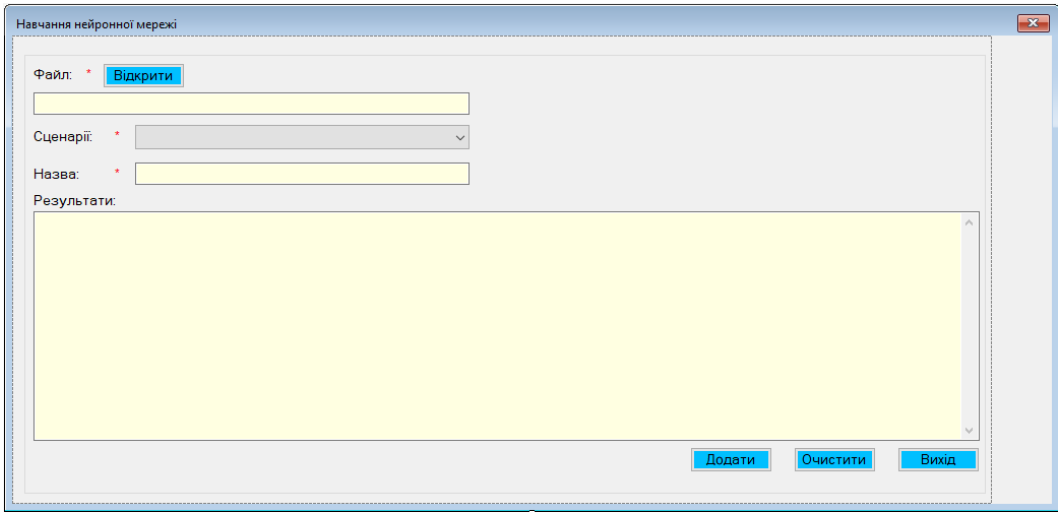

Рисунок 3.10 – Форма для навчання моделей

На рис.3.11 відображено код, який уможливлює користувачам вибір тренувального файлу з їх локальної системи та передачу шляху до цього файлу програмі для подальшого використання.

```
// Відображення діалогового вікна та обробка результату
if (openFileDialog.ShowDialog() == DialogResult.OK) {

    _Path = openFileDialog.FileName;
    FileNameTBox.Text = openFileDialog.FileName;
```

Рисунок 3.11 – Фрагмент коду відкриття файлу із тренувальним набором

Згідно з рис. 3.12, відбувається процес ініціалізації контексту машинного навчання за допомогою фреймворку ML.NET, що є інструментом від Microsoft для роботи з машинним навчанням на платформі .NET. Команда «new MLContext» ініціює новий об'єкт контексту для подальшої роботи з моделями машинного навчання.

```
// Створення контексту ML
mlContext = new MLContext(seed: 0);
```

Рисунок 3.12 – Ініціалізація контексту

У конструкторі MLContext параметр «seed: 0» задає початкове значення для генерації випадкових чисел, що забезпечує повторюваність результатів тренувань. Це критично важливо для об'єктивного тестування та порівняння ефективності моделей машинного навчання.

Наступним кроком було імплементовано процес завантаження текстових даних для їх подальшої обробки у машинному навчанні, як демонструється на рис. 3.13.

```
// Завантаження даних
dataView = mlContext.Data.LoadFromTextFile<PhishingData>(_Path,
  hasHeader: true,
  separatorChar: ',');
```

Рисунок 3.13 – Завантаження даних із файлу

Метод «LoadFromTextFile» з контексту машинного навчання «mlContext», що є частиною фреймворку ML.NET, використовується для імпорту даних з текстового файлу. Це дозволяє залучити текстовий файл як джерело даних для навчання моделей.

У цьому контексті, параметр «PhishingData» задає тип даних, які мають бути завантажені, перетворюючи кожен рядок файлу на екземпляр класу «PhishingData», визначеного в програмі. Шлях до файлу, зазначений змінною «_Path», визначає локацію файлу-джерела. Наявність заголовків у файлі позначається параметром «hasHeader», а роздільником даних слугує «separatorChar», що індикує кому як роздільник у CSV-форматі.

У фрагменті коду на рис. 3.14 виконується розділення датасету на дві частини: тренувальний набір даних та тестовий набір даних. Це стандартний процес у

машинному навчанні, який дозволяє оцінити ефективність моделі на даних, які не використовувалися під час тренування..

```
// Розділіть датасет на тренувальний та тестовий
var splitData = mlContext.Data.TrainTestSplit(dataView, testFraction: 0.2);
```

Рисунок 3.14 – Розділення даних тренувального набору

На наступному кроці відбувається підготовка даних до застосування у машинному навчанні, як демонструється на рис. 3.15. За допомогою інструментів, наданих бібліотекою ML.NET, формується об'єкт для виконання попередньої обробки даних, що є необхідною умовою перед їх введенням у модель. Ключовою операцією є застосування методу «Concatenate» з контексту mlContext. Метод призначений для злиття декількох вхідних атрибутів в один вектор ознак, підготовляючи дані для аналізу моделлю.

```
var dataProcessPipeline = mlContext.Transforms.Concatenate("Features",
    nameof(PhishingData.Have_IP), nameof(PhishingData.Have_At),
    nameof(PhishingData.URL_Length), nameof(PhishingData.URL_Depth),
    nameof(PhishingData.Redirection), nameof(PhishingData.https_Domain),
    nameof(PhishingData.TinyURL), nameof(PhishingData.Prefix_Suffix),
    nameof(PhishingData.DNS_Record),  nameof(PhishingData.Web_Traffic),
    nameof(PhishingData.Domain_Age), nameof(PhishingData.Domain_End),
    nameof(PhishingData.iFrame), nameof(PhishingData.Mouse_Over),
    nameof(PhishingData.Right_Click), nameof(PhishingData.Web_Forwards));
```

Рисунок 3.15 – Підготовка конвеєру даних для навчання

Процес включає злиття різноманітних атрибутів з даних типу «PhishingData» у єдиний вектор. До цих атрибутів відносяться, наприклад, присутність IP адреси в URL (Have_IP), символу '@' (Have_At), довжина URL (URL_Length), серед інших. Кожен атрибут відображає конкретну характеристику даних, що може бути корисною для ідентифікації фішингу.

В результаті отримується пайплайн dataProcessPipeline, який задіяний для виконання стандартизованої обробки даних перед їх використанням у тренувальному процесі. Цей пайплайн сприяє уніфікації та забезпеченню відтворюваності обробки даних, що є критично важливим для ефективного навчання моделей.

Наступний крок полягає у виборі підходящого алгоритму для машинного навчання, як демонструється на рис.3.16.

```
// Вибір алгоритму навчання, наприклад, логістичну регресію
var trainer =
  mlContext.BinaryClassification.Trainers.LbfgsLogisticRegression(labelColumnName: "Label",
  featureColumnName: "Features");
```

Рисунок 3.16 – Вибір алгоритму навчання

У даному випадку для задачі бінарної класифікації використовується відповідний тренер з контексту машинного навчання mlContext. Зокрема, застосовується метод LbfgsLogisticRegression для ініціації моделі тренування, що базується на логістичній регресії з оптимізатором LBFGS.

Параметр «labelColumnName» визначає назву колонки в датасеті, де зберігаються мітки класифікації, ідентифікуючи кожен екземпляр як належний до одного з двох можливих класів (наприклад, вказуючи, чи є сайт фішинговим). «featureColumnName» вказує на колонку, що містить ознаки кожного прикладу, об'єднані в один вектор, який слугує вхідними даними для тренування моделі. Використання такого методу дозволяє ефективно класифікувати приклади, базуючись на їх ознаках, що є стандартною процедурою у вирішенні задач бінарної класифікації, включно з виявленням фішингових сайтів.

Далі в процесі створення системи машинного навчання формується тренувальний пайплайн, який включає об'єкт тренера, обраний на попередніх етапах, до пайплайну попередньої обробки даних, ілюстрований на рис. 3.17.

```
//Створення тренувального пайплайну
var trainingPipeline = dataProcessPipeline.Append(trainer);
```

Рисунок 3.17 – Створення пайплайну

Основною дією є додавання, за допомогою методу Append, тренера до «dataProcessPipeline», що уже містить необхідні дії для підготовки даних, включаючи конкатенацію ознак. Таким чином, об'єкт «trainer», який визначає алгоритм навчання, зокрема LBFGS для логістичної регресії, інтегрується з даними, підготовленими для навчання.

В результаті цього процесу формується «trainingPipeline», комплексний тренувальний пайплайн, що об'єднує етапи попередньої обробки даних та сам процес

тренування. Цей пайплайн застосовується для навчання моделі на відповідному датасеті, де дані спочатку проходять через встановлені обробки в «dataProcessPipeline», а потім використовуються в «trainer» для ефективного навчання моделі.

Процес тренування моделі, показаний на рис. 3.18, включає застосування визначеного пайплайну до тренувального набору даних, що є критичним для створення ефективної моделі, здатної виконувати прогнозування або класифікацію. У цьому кроці модель адаптує свої параметри, навчаючись на основі тренувальних даних для максимізації точності прогнозів.

```
// Навчання модель
trainedModel = trainingPipeline.Fit(splitData.TrainSet);
```

Рисунок 3.18 – Навчання моделі

На наступному етапі здійснюється аналіз ефективності тренованої моделі та демонстрація її метрик у спеціалізованому полі, як показано на рис. 3.19.

```
// Оцінка модель
var predictions = trainedModel.Transform(splitData.TestSet);
var metrics = mlContext.BinaryClassification.Evaluate(predictions, "Label");

// Виведення метрик
RaportTBox.Text += "Метрики:" + "\r\n";
RaportTBox.Text += ($"AUC: {metrics.AreaUnderRocCurve:P2}") + "\r\n";
RaportTBox.Text += ($"F1 Score: {metrics.F1Score:P2}") + "\r\n";
RaportTBox.Text += ($"Precision: {metrics.PositivePrecision:P2}") + "\r\n";
RaportTBox.Text += ($"Recall: {metrics.PositiveRecall:P2}") + "\r\n";
```

Рисунок 3.19 – Оцінка моделі та виведення її метрик

Спершу, застосовується функція Transform з метою перетворення тестових даних за допомогою моделі, що дозволяє отримати прогнози на даних, що раніше не були задіяні в процесі тренування. Ця дія допомагає визначити, як модель впорається з новими даними.

Потім, використовуючи можливості mlContext, відбувається оцінювання отриманих прогнозів. За допомогою методу Evaluate, призначеного для роботи з бінарною класифікацією, проводиться розрахунок метрик продуктивності моделі,

заснований на порівнянні її прогнозів з реальними мітками в тестовому датасеті. До основних метрик належать AUC, F1 Score, точність (Precision) та повнота (Recall), що надалі представляються в текстовому полі RaportTBox. Результати метрик детально форматуються і вносяться до поля, забезпечуючи зручний огляд та аналіз продуктивності моделі.

Цей процес відіграє ключову роль у визначенні рівня ефективності моделі, зокрема в аспекті її спроможності коректно класифікувати нові, не використовувані при тренуванні, екземпляри даних. Оцінка таких метрик, як AUC, F1 Score, точність та повнота, є фундаментальною частиною процесу машинного навчання для аналізу бінарних класифікаторів.

Для тестування моделі розроблено засіб, що дозволяє застосовувати треновану модель машинного навчання для виконання прогнозів, як показано на рис. 3.20, з використанням інструментарію, наданого ML.NET.

```
// Використання моделі для передбачення
var predictionFunction = mlContext.Model.CreatePredictionEngine<PhishingData,
  PhishingPrediction>(trainedModel);
```

Рисунок 3.20 – Оцінка моделі та виведення її метрик

У даному коді реалізовано оцінювання продуктивності тренованої моделі машинного навчання та представлення її основних метрик. Використовуючи метод CreatePredictionEngine з mlContext, створюється механізм для прогнозування. Цей метод приймає типи даних для входу (PhishingData) і виходу (PhishingPrediction), створюючи інструмент для прогнозування, який використовує дані PhishingData, застосовує навчену модель для отримання прогнозів і повертає результати у вигляді PhishingPrediction.

Тут, trainedModel є моделлю, що була тренована на визначеному наборі даних і тепер може бути використана для прогнозування на нових даних. Це призводить до створення predictionFunction, інструменту для здійснення прогнозів на основі нових даних у форматі PhishingData. Таке застосування дозволяє ефективно використовувати навчену модель для оцінювання нових даних, наприклад, для ідентифікації фішингових сайтів.

На наступному кроці проводиться процес прогнозування з використанням тренованої моделі машинного навчання і представлення отриманих результатів у текстовому полі інтерфейсу, що демонструється на рис. 3.21.

```
var prediction = predictionFunction.Predict(phishingExample1);
RaportTBox.Text += "Тестовий приклад 1:" + "\r\n";
RaportTBox.Text += (prediction.Prediction ? "Виявлено фішінг" : "Не виявлено фішінг") + "\r\n";
RaportTBox.Text += ($"Ймовірність фішингової атаки: { prediction.Probability:P2}") + "\r\n";
```

Рисунок 3.21 – Передбачення та виведення результатів

Спершу, через раніше сформований об'єкт predictionFunction проводиться аналіз тестового випадку phishingExample1, який включає різноманітні ознаки, потенційно індикуючі фішинг. Метод Predict цього об'єкта оцінює ці ознаки, видаючи прогноз щодо наявності фішингу в даному прикладі.

Інформація про висновок моделі відображається в текстовому полі RaportTBox. Спочатку вказується назва аналізованого прикладу, а потім, залежно від результату прогнозу моделі (prediction.Prediction), надається повідомлення про виявлення чи не виявлення фішингу.

Додатково, до текстового поля додається відсоткова імовірність того, що розгляданий випадок є фішинговим (prediction.Probability), що надає змогу зрозуміти рівень впевненості моделі в своєму висновку. Це дозволяє користувачам отримати більш глибоке розуміння щодо оцінки ризику фішингу з боку моделі.

Функціональність зберігання тренованої моделі впроваджена через обробник події «AddBtn_Click», деталі якого наведено на рис. 3.22.

```
private void AddBtn_Click(object sender, EventArgs e) {
  if (IsDataEnteringCorrect()) {
    //Save model
    string pathName = @"\teach\" + GenerateFileName() + ".zip";
    string localProj =
      System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location);
    _NeuralProvider.InsertNeural(NeuralNamesTBox.Text,
      Convert.ToInt32(ScenariosCBox.SelectedValue), pathName);
    mlContext.Model.Save(trainedModel, dataView.Schema, localProj + pathName);
    ClearAllData();
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId,
      "Було навчено нейронну мережу " +
      NeuralNamesTBox.Text, DateTime.Now);
    MessageBox.Show("Дані успішно збережено!");
    _IsModelTrain = false;
  }
}
```

Рисунок 3.22 – Метод збереження даних моделі

На початку процедура перевіряє правильність введення даних за допомогою методу «IsDataEnteringCorrect». У разі валідності введення ініціюється процес збереження тренованої моделі. Ім'я файлу моделі формується за допомогою «GenerateFileName», до якого додається розширення «.zip» та шлях до директорії «teach». Встановлюється локальний шлях до проекту (localProj), який використовується далі для збереження файлу моделі. Функція «InsertNeural» слугує для внесення відомостей про модель до системи, включно з її назвою, сценарієм використання та місцем зберігання.

За допомогою «Save» відбувається фіксація моделі у файловій системі за вказаним шляхом. Після збереження запускається «ClearAllData» для очищення полів вводу на формі.

Процедура «InsertLogs» заносить відомості про процес тренування моделі в систему логування. На завершення користувач отримує сповіщення про успішне збереження моделі, а також змінна «_IsModelTrain» переводиться в стан false, що сигналізує про завершення процесу тренування.

Цей механізм відіграє ключову роль у зв'язуванні процесу навчання моделі з інтерфейсом користувача, забезпечуючи зручність у зберіганні, реєстрації та контролі над тренованими моделями в системі.

Для тестування навчених моделей було реалізовано форму «Тестування моделей», зовнішній вигляд якої представлено на рис. 3.23.

Функціонал для оцінки ефективності тренованих моделей був впроваджений через інтерфейс «Тестування моделей», ілюстрація якого знаходиться на рис. 3.24.

Обробник події «ScenariosCBox_SelectedValueChanged» забезпечує можливість автоматичної зміни вибраної моделі для тестування через випадаючий список.

Цей метод автоматизує процес вибору та демонстрації конкретної моделі машинного навчання та відповідного датасету для тестування, що спрощує взаємодію користувача з системою та підвищує ефективність обробки даних.

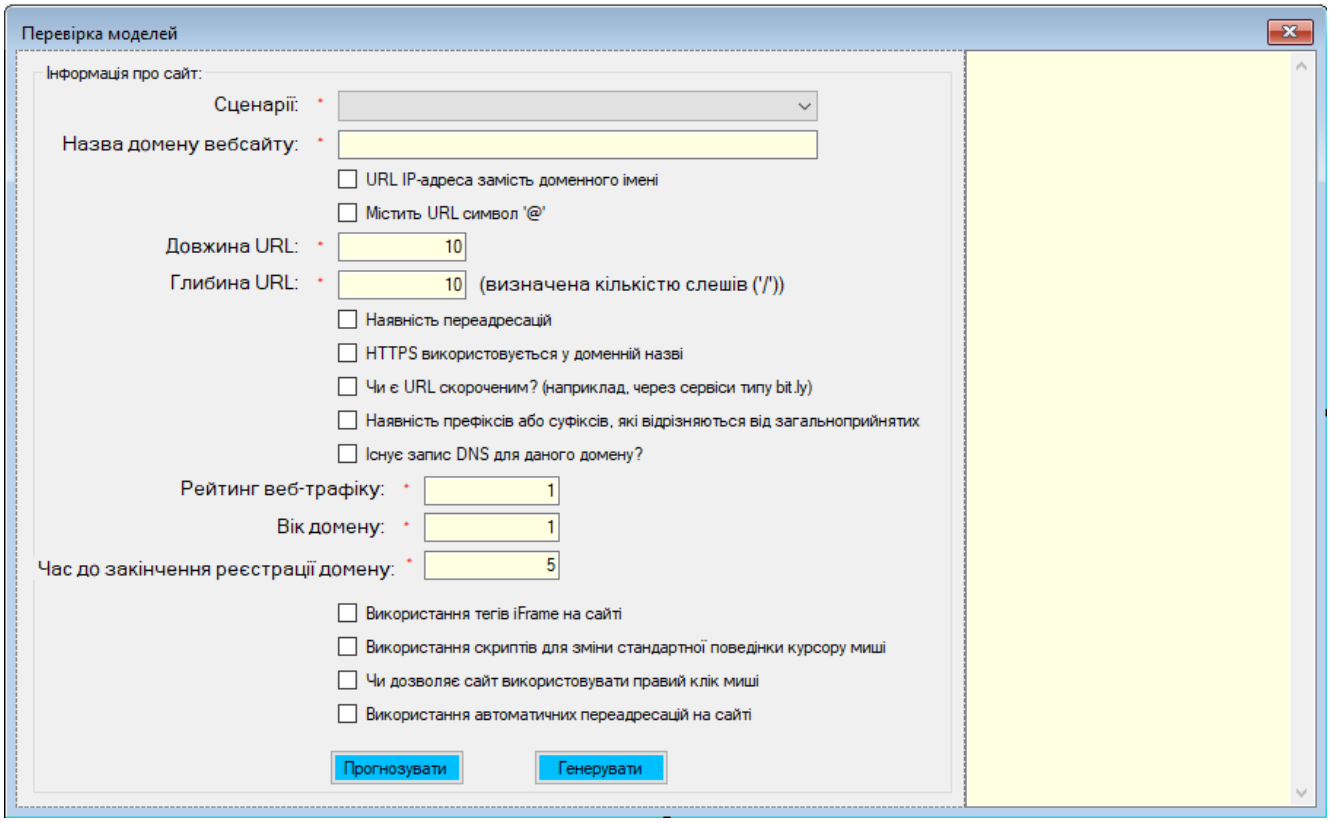

Рисунок 3.23 – Вигляд форми «Тестування моделей»

```
private void ScenariosCBox_SelectedValueChanged(object sender, EventArgs e) {
  if (_IsThemesLoad && _ScenariosList[0].Message != NamesMy.NoDataNames.NoDataInScenarios) {
    _SelectedNeural = _NeuralProvider.SelectedNeuralByScenariosId(
      Convert.ToInt32(ScenariosCBox.SelectedValue));
    LoadData(_SelectedNeural.NeuralFileModel);
  }
}
```

Рисунок 3.24 – Код події зміни сценарію прогнозування

На початку встановлюється перевірка завантаження тем через прапорець «_IsThemesLoad». У разі позитивної відповіді, здійснюється вибір відповідної моделі через «_SelectedNeural», яка ініціалізується методом «SelectedNeuralByScenariosId» на основі ідентифікатора обраного сценарію з «ScenariosCBox», конвертуючи його значення в ціле число.

Далі, функція «LoadData» завантажує треновану модель нейронної мережі з файла, а «_TestPhishingDatas» ініціалізується тестовими даними для перевірки на фішинг, які беруться з визначеного шляху (_testDataPath) через «LoadPhishingData».

Цей підхід дозволяє з легкістю налаштовувати та проводити тестування моделей на різноманітних датасетах, сприяючи глибшій оцінці їхньої продуктивності.

Обробник події «RunBtn_Click», представлений на рис. 3.25, був створений для керування процесом генерації даних для тестування моделі або її призупинення в залежності від актуального статусу системи.

```
private void GetBtn_Click(object sender, EventArgs e) {
  if (IsScenariosSelectedCorrect()) {
    if (timer1.Enabled) {
      timer1.Enabled = false;
      GenBtn.Text = "Генерувати";
    } else {
      timer1.Enabled = true;
      GenBtn.Text = "Зупинити";
    }
  }
}
```

Рисунок 3.25 – Код для події «RunBtn_Click»

Цей метод регулює активність таймера та адаптує надпис на відповідній кнопці відповідно до поточного режиму роботи таймера. Початковим кроком є перевірка коректності вибору сценаріїв через «IsScenariosSelectedCorrect», що забезпечує запуск подальших операцій лише при відповідності вибору заданим критеріям.

При підтвердженні правильного вибору сценаріїв здійснюється аналіз стану таймера «timer1». У випадку, коли таймер вже працює («Enabled» дорівнює true), він буде зупинений («Enabled» стає false), а текст кнопки «GenBtn» змінюється на «Генерувати», сигналізуючи про можливість реініціювати процес заново.

У ситуації, коли таймер не був активний у момент натискання кнопки, він запускається («Enabled» стає true), а текст на кнопці «GenBtn» оновлюється до «Зупинити», надаючи користувачу змогу припинити запущений процес за бажанням.

Розроблено функціонал «Тестування моделей» з можливістю вручну вводити дані для перевірки моделей машинного навчання, демонстрація якого знаходиться на рис. 3.26.

```
private void PredictBtn_Click(object sender, EventArgs e) {
  if (IsAllPhishingDataCorrect()) {
    var prediction = predictionEngine.Predict(new PhishingData {
      Domain = DomainTBox.Text,  Have_IP = Have_IPChBox.Checked ? 1.0f : 0.0f,
      Have_At = Have_AtChBox.Checked ? 1.0f : 0.0f, URL_Length = float.Parse(URL_LengthTBox.Text),
      URL_Depth = float.Parse(URL_DepthTBox.Text),  Redirection = RedirectionChBox.Checked ? 1.0f : 0.0f,
      https_Domain = https_DomainChBox.Checked ? 1.0f : 0.0f,  TinyURL = TinyURLChBox.Checked ? 1.0f : 0.0f,
      Prefix_Suffix = Prefix_SuffixChBox.Checked ? 1.0f : 0.0f,  DNS_Record = DNS_RecordChBox.Checked ? 1.0f : 0.0f,
      Web_Traffic = float.Parse(Web_TrafficTBox.Text), Domain_Age = float.Parse(Domain_AgeTBox.Text),
      Domain_End = float.Parse(Domain_EndTBox.Text), iFrame = iFrameChBox.Checked ? 1.0f : 0.0f,
      Mouse_Over = Mouse_OverChBox.Checked ? 1.0f : 0.0f,  Right_Click = Right_ClickChBox.Checked ? 1.0f : 0.0f,
      Web_Forwards = Web_ForwardsChBox.Checked ? 1.0f : 0.0f
    });
    var PhishingInfo = new StringBuilder();
    // Прогноз до виводу
    PhishingInfo.AppendLine("\r\n--- Прогнозування ---");
    PhishingInfo.AppendLine(prediction.Prediction ? "Виявлено фішінг" : "Не виявлено фішінг");
    PhishingInfo.AppendLine($"Ймовірність фішингової атаки: {prediction.Probability:P2}");
    PhishingInfo.AppendLine($"Бал: {Math.Abs(prediction.Score)}");

    // Виведіть результат у текстове поле
    RaportTBox.Text = PhishingInfo.ToString();
  }
}
```

Рисунок 3.26 – Метод для тестування моделей

При активації кнопки «Predict», відбувається процедура, що аналізує, чи певний сайт має ознаки фішингу. Початковий крок полягає у валідації введених даних через «IsAllPhishingDataCorrect». У разі позитивної перевірки формується інстанція «PhishingData» з користувацькими даними, які охоплюють домен, наявність IP, символ '@' в URL та інші параметри. Ці дані подаються для аналізу об'єктом «predictionEngine». Далі формується змінна «PhishingInfo», що зберігає деталі передбачення, включаючи результат виявлення фішингу, ймовірність такої атаки та точність передбачення.

Інформація про результати прогнозування демонструється у текстовому полі «RaportTBox», надаючи користувачам інформацію про потенційну фішингову активність аналізованого сайту та рівень впевненості моделі.

Цей інструмент надає можливість користувачам системи оцінити, наскільки модель вважає певний сайт фішинговим, та з якою впевненістю.

Окремо згадано два методи аналізу технік прогнозування фішингових атак: метод покриття операторів, який вимірює обсяг виконаного коду під час тестування, і метод припущення про похибку, зосереджений на виявленні типових помилок розробників. Ці підходи сприяють ідентифікації потенційних недоліків та слабких

місць в коді, допомагаючи забезпечити більш високий рівень якості програмного продукту через глибокий аналіз та тестування.

3.6 Моделювання потенційних сценаріїв атак та аналіз вразливостей

У процесі розробки даного проекту було приділено особливу увагу тестуванню програми, що відіграє ключову роль у забезпеченні відповідності коду встановленим критеріям та забезпеченні його надійності. Модульне тестування, зокрема, виступило як центральний елемент цієї стратегії, дозволяючи перевірити індивідуальні частини програми на коректність роботи. З використанням MSTest, були розроблені та реалізовані тести для оцінки важливих аспектів системи. Це охоплювало тестування бізнес-логіки, включно з аналізом ефективності алгоритмів, таких як розрахунок маршрутів, визначення часу паркування, та оцінку вартості послуг, забезпечуючи їх відповідність вимогам. Паралельно було проведено тестування механізмів доступу до даних, перевіряючи надійність їх зберігання та обробки.

На рис. 3.27 відображені результати модульного тестування, які демонструють успішність здійснених тестових процедур у забезпеченні високої якості та безпеки розробленого програмного рішення.

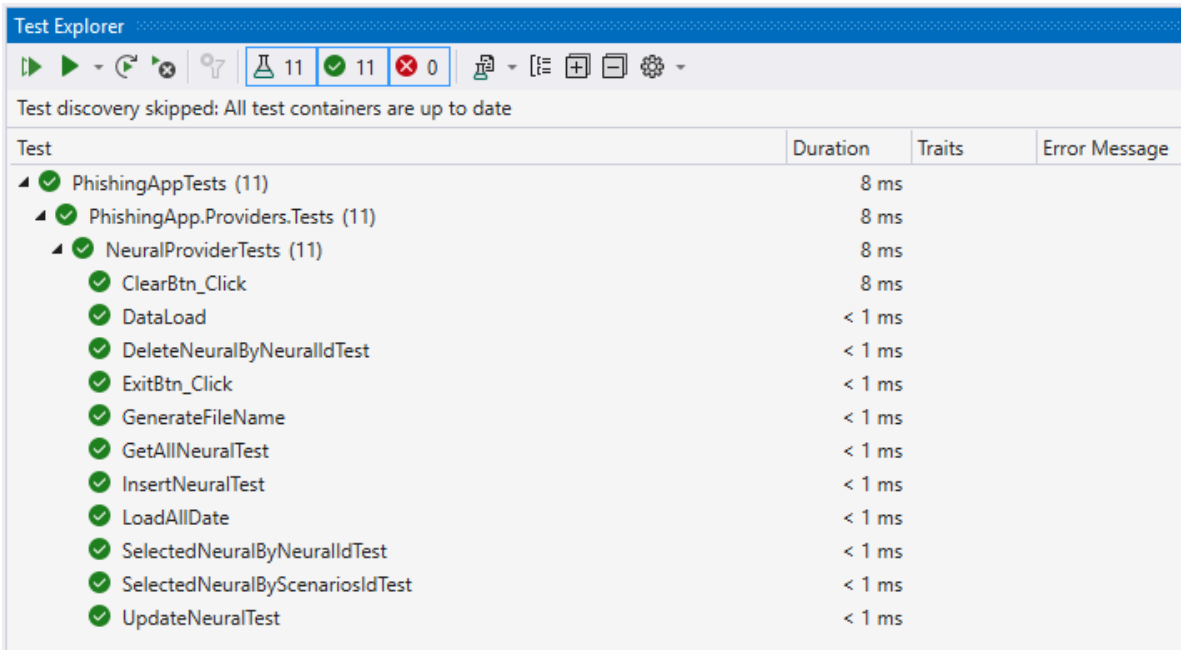

Рисунок 3.27 – Результати проведеного модульного тестування

Ці заходи сприяли формуванню детальної та комплексної тестувальної стратегії, кульмінацією якої стало узагальнення та візуалізація результатів. Візуальні звіти, представлені у вигляді графіків та діаграм, наглядно ілюструють обсяг виконаного тестування, охоплення ключових частин системи тестами, а також ефективність виявлення і виправлення помилок та вразливостей.

3.7 Аналіз точності та надійності системи детекції фішингу

В рамках дослідження щодо ідентифікації фішингових сайтів основну увагу було приділено застосуванню актуальних та якісних даних. Використаний датасет, зібраний з публічно доступних ресурсів, включав ключові атрибути, пов'язані з фішингом, такі як доменні імена, наявність IP-адрес в URL, довжину URL, глибину структури URL, показники редиректів та інше, що сприяло глибокому аналізу елементів, характерних для фішингових атак.

Перед ініціацією експерименту була проведена детальна підготовка даних, включаючи перевірку на аномалії та очищення від застарілих чи некоректних записів. Нормалізація даних забезпечила їх однорідність і зняла ризик надмірного впливу окремих атрибутів на результати обробки. Стандартизація форматів даних зіграла важливу роль у забезпеченні точності подальшої роботи з ними в контексті машинного навчання.

Процес тренування моделі включав використання добре підібраних прикладів з датасету, що дало змогу моделі ефективно ідентифікувати потенційні ознаки фішингу. На рис. 3.28 представлено інтерфейс з даними, підготовленими для тренування, включно з візуалізацією вхідних даних та їх адаптацією для введення в модель, що надало можливість оцінити якість та формат даних перед використанням в машинному навчанні.

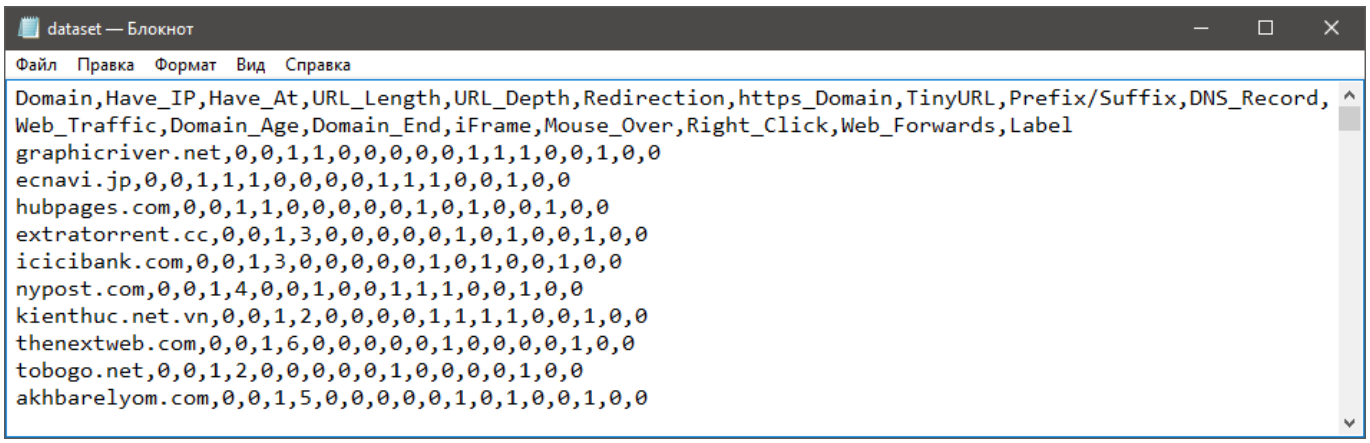

Рисунок 3.28 – Вигляд даних для навчання моделі

У додатку був реалізований процес навчання моделі, під назвою «Виявлення фішингу», що демонструється на рис. 3.29.

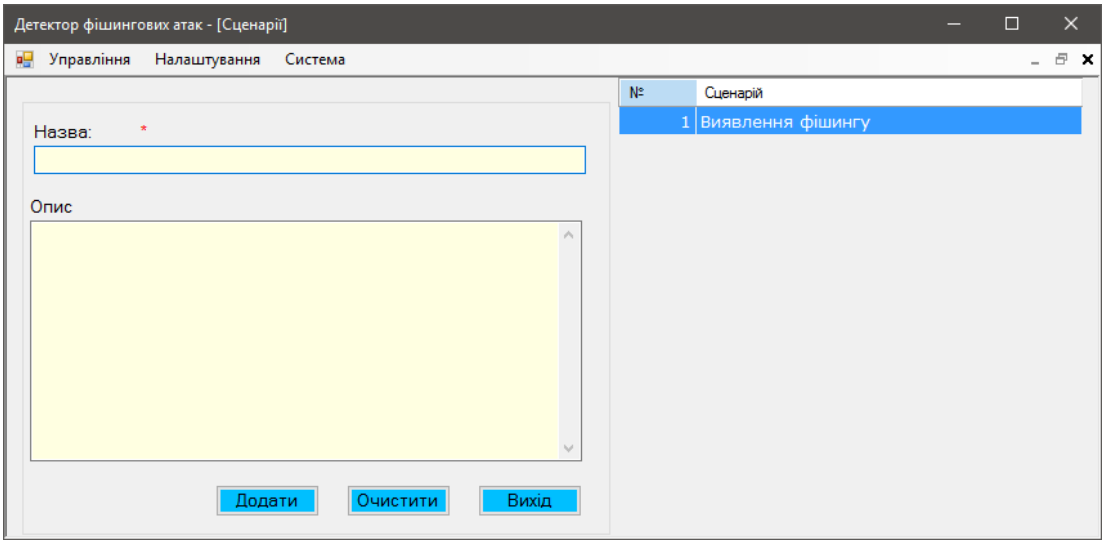

Рисунок 3.29 – Створення сценарію

Для ініціації тренування моделі у відповідності до визначеного сценарію, виконано дії через інтерфейс програми за маршрутом «Налаштування» → «Навчання моделі», де було обрано заздалегідь встановлений сценарій «Виявлення фішингу». Процес вибору необхідного датасету для тренування моделі відбувся через спеціальне діалогове вікно, після чого процедура навчання запустилася автоматично. Цей етап і його результати представлені на рисунку 3.30, де візуалізовано успішне завершення тренування моделі.

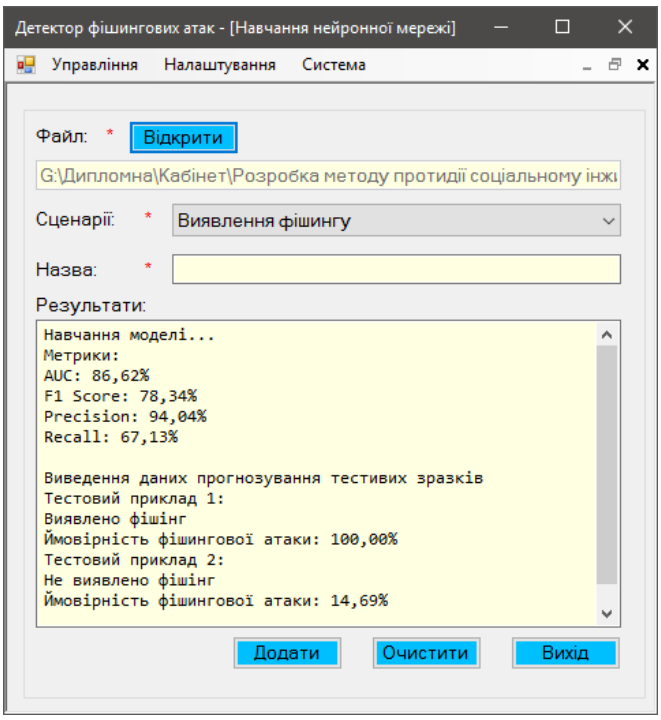

Рисунок 3.30 – Результат навчання моделі

Аналізуючи метрики моделі машинного навчання для виявлення фішингових сайтів, можна зробити детальні висновки щодо її продуктивності та точності:

– AUC (Площа під кривою ROC) зі значенням 86,62% свідчить про високу здатність моделі диференціювати між легітимними та фішинговими сайтами. Це вказує на те, що модель має сильні загальні класифікаційні здібності;

– F1 Score, досягаючи показника 78,34%, вказує на добру балансованість між точністю та повнотою виявлення фішингових сайтів. Це означає, що модель ефективно балансує між уникненням хибнопозитивних та хибнонегативних висновків;

– Precision (Точність) із показником 94,04% свідчить про те, що модель має високу надійність у позитивних прогнозах, але також вказує на потенційну пересторогу моделі в ідентифікації фішингових сайтів, що може призводити до пропуску деяких справжніх випадків фішингу;

– Recall (Повнота) із показником 67,13% ідентифікує відсоток справжніх фішингових випадків, які модель змогла коректно класифікувати. Це значення підкреслює, що модель може не виявляти деяку кількість фішингових атак, що вимагає подальшого покращення для збільшення її ефективності.

В загальному, модель показує солідну продуктивність у виявленні фішингових сайтів, особливо враховуючи її високу точність. Проте, з метою покращення здатності до ідентифікації всіх потенційних фішингових атак, необхідно зосередитися на підвищенні повноти моделі, що може включати оптимізацію її параметрів або навчання на більш широкому чи різноманітному наборі даних.

Успішно тренована модель подальше була інтегрована у систему для здійснення експериментальних досліджень, що демонструється на рис. 3.31, де показано момент збереження моделі у системі.

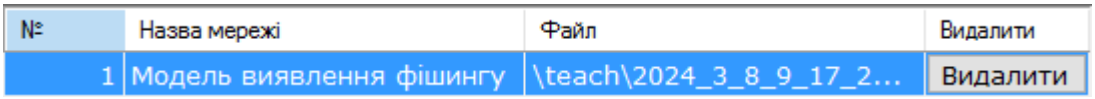

Рисунок 3.31 –Збереження моделі

Експеримент з застосуванням цієї моделі розпочався з доступу до опції «Тестування моделі» у програмному меню, що активувало інтерфейс для симуляції можливих фішингових атак. Це надало користувачам можливість вибрати конкретний сценарій атаки з переліку для аналізу.

Етапи експерименту включали:

– генерацію тестових даних. Скориставшись класом для генерації доменів (GenerateRandomDomain), програма створювала множину варіантів для потенційних фішингових атак, забезпечуючи різноманітність даних для аналізу;

– виконання прогнозів моделлю. Модель отримувала сгенеровані дані та проводила їх аналіз, визначаючи ймовірність кожної атаки бути фішинговою;

– аналіз результатів. Отримані від моделі результати піддавались оцінці, де кожний сценарій розглядався на предмет його відповідності критеріям фішингу;

– візуалізація результатів та їх звітування. Інтерфейс програми надавав візуальне представлення результатів експерименту, включаючи детальну інформацію про кожен випробуваний варіант та висновки моделі.

Цей експеримент мав на меті не тільки продемонструвати здатність моделі ідентифікувати фішингові сайти, а й оцінити її продуктивність та надійність у

широкому спектрі можливих ситуацій, випробовуючи її адаптивність та точність у різних умовах.

Для тестування точності навченої моделі згенеровано декілька потенційних випадків за допомогою розробленого генератора сценаріїв та детально їх проаналізовано. На рис. 3.32 зображено перший згенерований випадок.

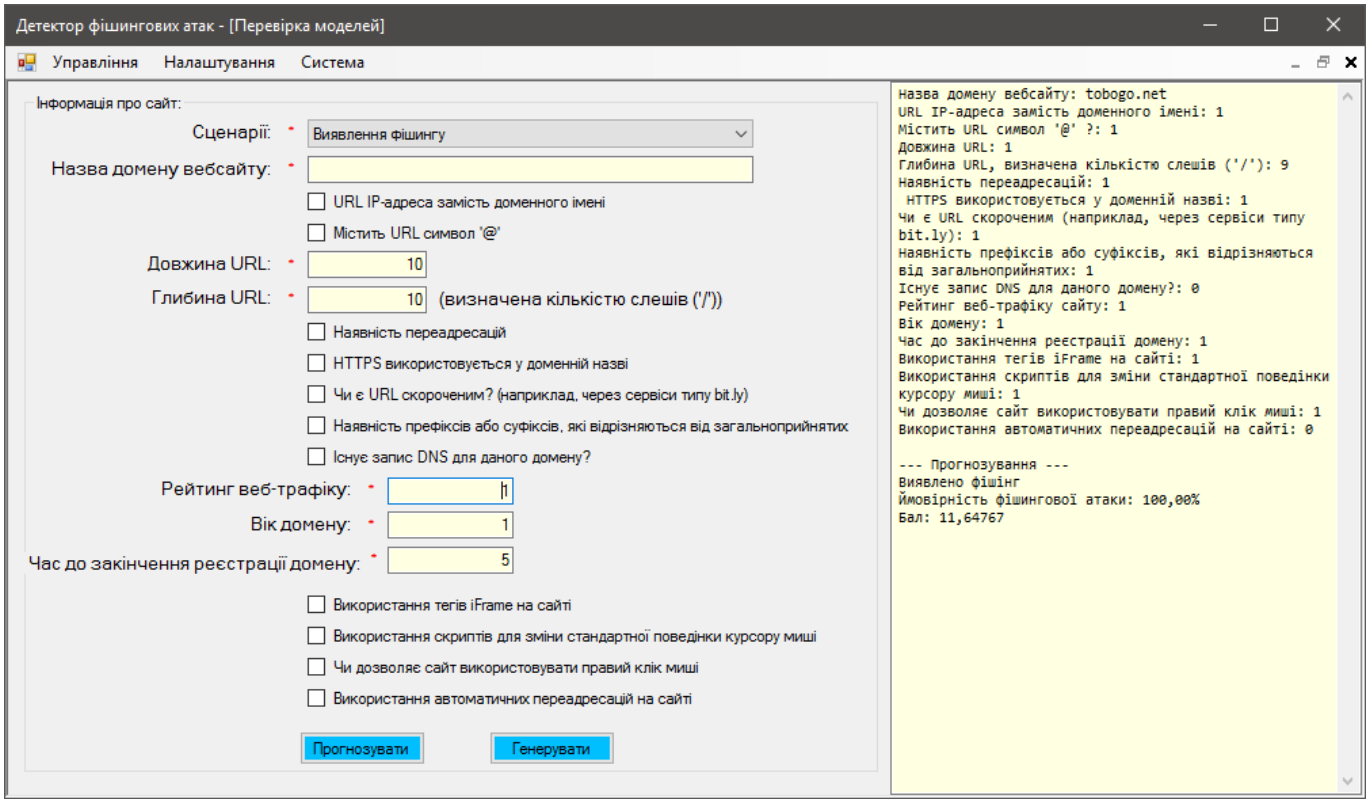

Рисунок 3.32 –Результат прогнозування першого випадку

Аналізуючи надані характеристики веб-сайту та результати прогнозування, можна виділити кілька ключових аспектів щодо ефективності моделі машинного навчання у виявленні фішингових атак:

1. Ознаки веб-сайту "tobogo.net" вказують на значну кількість індикаторів, які зазвичай асоціюються з фішингом, включаючи:

– використання IP-адреси замість доменного імені і символ '@' в URL, що є нетиповим для більшості легітимних веб-сайтів;

– висока довжина URL і значна глибина URL можуть свідчити про спроби приховати справжній домен чи вмістити в адресу шкідливі параметри;

– наявність переадресацій, скорочення URL, і використання HTTPS у доменній назві разом з іншими ознаками, такими як невикористання DNS-запису, можуть вказувати на потенційно шкідливу активність.

2. Метрики прогнозування підтверджують високу ймовірність фішингової атаки (100,00%) з високим балом 11,64767, що вказує на велику впевненість моделі у своєму висновку. Це демонструє, що модель ефективно ідентифікує характеристики, типові для фішингових сайтів, і здатна з великою точністю класифікувати подібні випадки.

Враховуючи аналіз, можна зробити висновок, що модель має сильні показники у виявленні фішингових сайтів, ефективно використовуючи набір ознак для аналізу та класифікації. Проте, важливо зазначити, що для забезпечення максимальної ефективності та адаптивності моделі до постійно змінюючихся методів фішингу, рекомендується регулярне оновлення датасету та фінтюнінг моделі.

На рис 3.33 зображено скріншот результат виконання 2-го випадку згенерованого сценарію фішингового сайту.

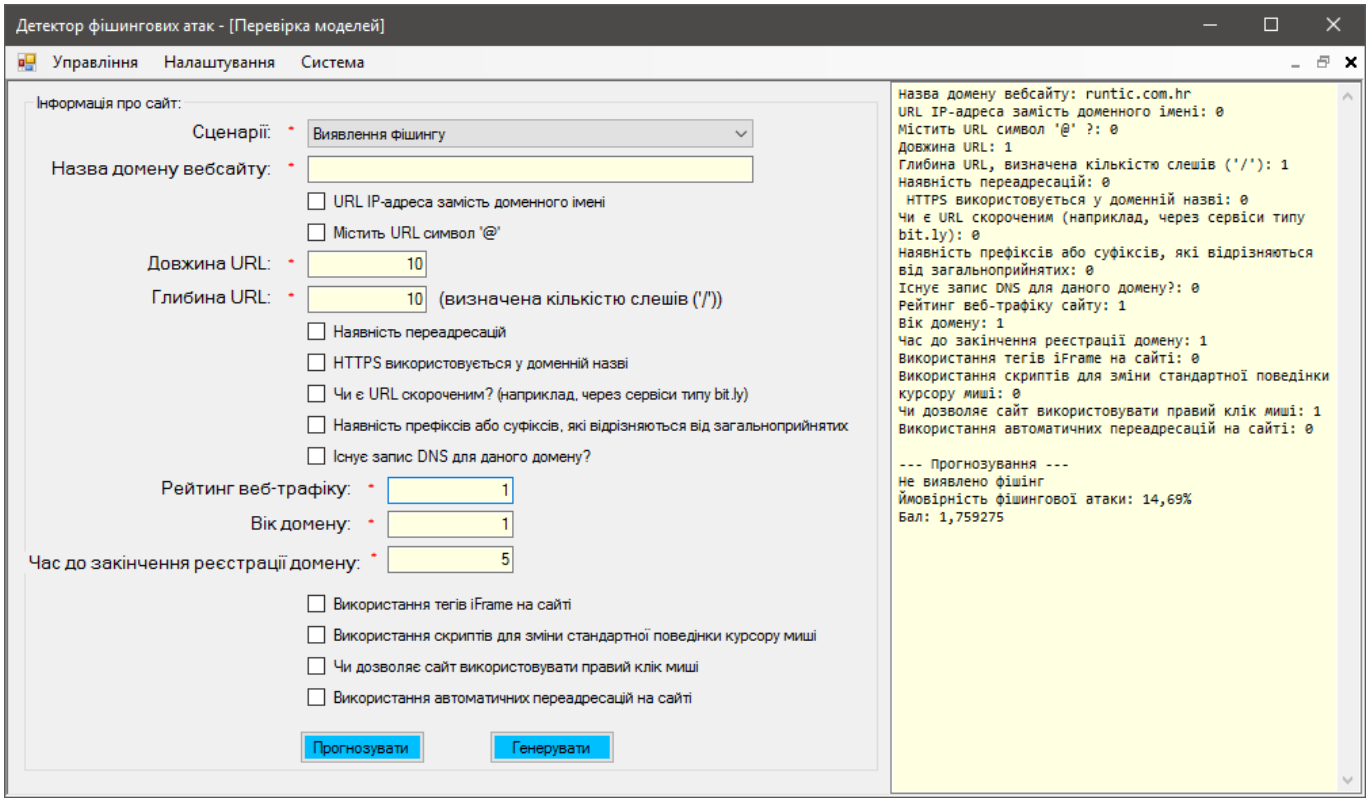

Рисунок 3.33 –Результат прогнозування 2-го сценарію

Аналізуючи характеристики веб-сайту "runtic.com.hr" та отримані результати прогнозування, можна зробити наступні висновки:

1. Ознаки веб-сайту надають змішану картину потенційних ризиків:

– відсутність IP-адреси замість доменного імені та символу '@' в URL вказує на менші шанси на фішинг у порівнянні з типовими фішинговими сайтами;

– довжина URL і глибина URL є помірними, що не є явними індикаторами фішингової діяльності в цьому випадку;

– відсутність переадресацій та скорочених URL, а також невикористання префіксів або суфіксів, що відрізняються від загальноприйнятих, сприяють позитивній оцінці безпеки сайту;

– ознаки, як рейтинг веб-трафіку, вік домену, і час до закінчення реєстрації домену вказують на стабільність та легітимність сайту.

2. Результати прогнозування вказують на низьку ймовірність фішингової атаки (14,69%) з невеликим балом 1,759275, що свідчить про значну впевненість моделі в тому, що сайт не є фішинговим. Це демонструє, що модель адекватно інтерпретує надані характеристики як нехарактерні для фішингових сайтів.

Узагальнюючи, можна зазначити, що веб-сайт "runtic.com.hr" має ознаки, які модель машинного навчання вважає безпечними, відповідно до низької ймовірності фішингової атаки. Отримані дані та прогноз моделі підтверджують легітимність сайту і відсутність явних ризиків фішингу, що робить його безпечним для користувачів. Такі результати вказують на високу точність та надійність моделі у виявленні фішингових вебсайтів, а також її здатність коректно класифікувати легітимні сайти.

В контексті проведеного дослідження, критичним аспектом виступає детальний аналіз роботи системи прогнозування фішингових вебсайтів. Такий аналіз дозволяє оцінити здатність системи акуратно розпізнавати потенційно шкідливі вебсайти, відокремлюючи їх від безпечних. Ключову роль у визначенні ефективності системи відіграють різноманітні метрики, отримані під час тренування машинного навчання, які ілюструють точність роботи системи:

– метрики тренування. Високе значення AUC (86,62%) підкреслює здатність системи відмежовувати фішингові сайти від легітимних. F1 Score (78,34%)

свідчить про досягнення балансу між точністю та відновленням, хоча і існує простір для удосконалення, особливо в аспекті відновлення. Показник точності (Precision) на рівні 94,04% вказує на високу вірогідність коректного ідентифікування фішингових сайтів системою, але порівняно низький показник відновлення (Recall) в 67,13% наголошує на необхідності покращення здатності системи виявляти всі потенційні загрози;

– результати прогнозування. Система продемонструвала спроможність точно ідентифікувати як потенційно небезпечні, так і безпечні вебсайти, що підтверджено аналізом тестових випадків. В одному зі сценаріїв система визначила високу ймовірність фішингу (100%), тоді як в іншому вказала на низьку ймовірність (14,68%), демонструючи впевненість у своїх оцінках;

– загальна продуктивність. Система показала обіцяючі результати у розпізнаванні фішингових сайтів, особливо з огляду на високу точність. Проте, для підвищення ефективності роботи системи рекомендується зосередитися на покращенні показника відновлення, щоб мінімізувати кількість пропущених фішингових сайтів;

– рекомендації для оптимізації. Налаштування та фінтюнінг алгоритмів, з метою підвищення відновлення без значної втрати в точності, можуть допомогти у покращенні загальних показників системи. Також корисним буде детальний аналіз помилково позитивних та помилково негативних результатів, щоб зрозуміти причини невірних класифікацій та оптимізувати процес виявлення фішингу.

Отже, система виявила високу ефективність у класифікації фішингових сайтів, втім існує простір для подальших удосконалень, спрямованих на збільшення відновлення та підвищення точності прогнозування в комплексі.

3.8 Рекомендації щодо застосування розробленого рішення в практиці

Для оптимізації роботи створеної системи з виявлення фішингових сайтів слід уважно підходити до вибору та обробки даних, використовуваних для тренування. Рекомендації для підвищення ефективності системи:

Вибір та підготовка датасетів

Для ефективного вибору та підготовки датасетів у системі протидії методам соціального інжинірингу необхідно зосередитися на таких ключових аспектах:

– акуратний вибір даних для тренування. Ефективність моделі залежить від якості тренувальних даних. Необхідно використовувати датасети, що містять комплексні характеристики фішингових вебсайтів, включаючи структуру URL, наявність SSL, доменну інформацію, метадані сторінок тощо, щоб охопити різноманітні тактики фішингу;

– очистка та оптимізація даних. Перед тренуванням моделі важливо очистити датасет від аномалій та пошкоджених записів. Це забезпечить вищу точність і якість тренування, видаляючи потенційно спотворюючі вхідні дані;

– уніфікація та нормалізація даних. Для забезпечення точного аналізу необхідно стандартизувати формат та масштаб даних. Нормалізація допоможе уникнути надмірного впливу певних ознак на результати тренування та класифікації моделі.

Застосування цих підходів допоможе забезпечити, що модель буде навчена на основі точних, актуальних та відповідно підготовлених даних. Це підвищить здатність системи адекватно розпізнавати фішингові атаки, спираючись на високоякісний аналіз та класифікацію.

Навчання моделі

Процес тренування моделей для ідентифікації фішингових атак в системі передбачення включає дотримання декількох фундаментальних етапів:

– використання комплексних датасетів. Ефективність моделей значною мірою залежить від якості та різноманітності тренувальних даних. Навчання моделей на датасетах, що містять широкий спектр прикладів фішингових атак, забезпечує їхню здатність розпізнавати широкий діапазон зловмисних технік і адаптуватися до нових тенденцій;

– оновлення та переоцінка моделей. У відповідь на постійні зміни у методах фішингу, важливо регулярно переглядати та адаптувати тренувальні датасети, а

також коригувати параметри моделей. Це дозволяє зберігати актуальність моделей і підвищувати їхню точність у розпізнаванні фішингових сайтів;

– застосування методів крос-валідації. Використання крос-валідації під час тренування моделі допомагає оцінити її здатність до узагальнення на нових даних, мінімізуючи ризик перенавчання. Розділення датасету на частини для послідовного тренування та тестування моделі на кожній з них дозволяє більш точно визначити її продуктивність.

Врахування цих критеріїв забезпечує високу надійність, адаптивність та точність моделей при виявленні фішингових атак, підвищуючи їх ефективність у різних умовах.

Тестування та оцінка ефективності

Для гарантування високої ефективності системи виявлення фішингу необхідно систематично проводити оцінку її продуктивності та проводити тестування, дотримуючись наступних важливих принципів:

– неперервне тестування на оновлених даних. Щоб уникнути зниження точності прогнозів системи з часом, рекомендується регулярно перевіряти її продуктивність на нових та різноманітних датасетах. Це допоможе виявити здатність системи адаптуватися до нових методів фішингу та підтримувати її актуальність;

– зосередження на ключових показниках ефективності. Визначення продуктивності системи через аналіз таких метрик, як AUC, F1 Score, Precision, та Recall, є критично важливим. Вони дозволяють оцінити здатність системи диференціювати між фішинговими та легітимними веб-сайтами та забезпечити оптимальний баланс між виявленням фішингу та уникненням помилкових спрацьовувань;

– аналіз помилкових спрацьовувань та пропущених випадків. Вивчення ситуацій, коли система надає хибнопозитивні або хибнонегативні результати, може виявити потенційні слабкі сторони в алгоритмах та сприяти їх удосконаленню. Особлива увага до цих аспектів дозволить ідентифікувати конкретні типи фішингу, які можуть бути не виявлені або невірно класифіковані, та внести корективи для покращення точності системи.

Врахування цих критеріїв дозволить створити більш точну та надійну систему виявлення фішингу, яка здатна ефективно протистояти сучасним загрозам у цифровому просторі.

Висновки за розділом 3

У рамках даного розділу було виконано комплексну розробку та верифікацію методики протидії фішинговим атакам, яка включала формування математичної бази для розпізнавання фішингу, збір та аналіз даних для тренування алгоритмів детектування, розробку алгоритмів роботи методу, проєктування бази даних системи, розробку основних модулів системи, моделювання потенційних сценаріїв атак та аналіз вразливостей. Розроблено математичну модель, що дозволяє із високою точністю ідентифікувати фішингові сайти, що підтверджено результатами тестування моделі з використанням набору даних з Kaggle. Важливим аспектом роботи стало створення трьохрівневої архітектури системи, що сприяло чіткому розподілу функціоналу та полегшенню інтеграції модулів. Проведено експериментальне моделювання сценаріїв атак, що дозволило виявити потенційні вразливості системи та внести необхідні корективи для підвищення її надійності та точності. Аналіз точності та надійності системи показав високу ефективність розробленого рішення, що засвідчено метриками, отриманими в результаті навчання моделі. На основі проведеної роботи були сформульовані рекомендації щодо застосування розробленого рішення в практиці, що включають рекомендації по вибору та підготовці датасетів, навчанню моделі та її тестуванню. Завдяки цьому, розроблена система демонструє не тільки теоретичну значущість, але й практичну застосовність для ефективної протидії фішинговим атакам.

ВИСНОВКИ

У результаті виконання дипломної роботи було розроблено систему виявлення фішингових сайтів з використанням методу Бройдена-Флетчера-Гольдфарба-Шанно, що дозволяє ефективно ідентифікувати потенційні фішингові атаки через аналіз характеристик веб-сайтів. Система виявилася ефективною у класифікації веб-сайтів за допомогою машинного навчання, що підтверджено високими показниками метрик точності та надійності.

У першому розділі дипломної роботи проведено ґрунтовний аналітичний огляд методів соціального інженірингу, акцентуючи увагу на їх здатності впливати на рівень захисту інформації в сучасному цифровому світі. Визначені основні напрями досліджень, які необхідно розробити для забезпечення ефективної протидії таким атакам. Розділ починається з глибокого аналізу історичних та сучасних технік соціального інженірингу, які використовуються для маніпуляції користувачами та обходу традиційних систем безпеки.

Особлива увага приділена дослідженню різноманітних сценаріїв фішингових атак, включаючи використання троянських програм, методи "кві про кво", претекстинг та використання маніпулятивної реклами. Кожен із цих сценаріїв детально аналізується з точки зору механізмів реалізації, цілей зловмисників та потенційного впливу на інформаційну безпеку. Це дозволяє не тільки краще зрозуміти методики ведення атак, але й виділити ефективні шляхи їх нейтралізації.

У другому розділі зосереджено увагу на методологічних аспектах дослідження та валідації підходів. Проаналізовано та обґрунтовано використання технік машинного навчання, зокрема методу БФГШ, для ідентифікації фішингових атак, виходячи з оцінки їхньої ефективності та адаптивності до сучасних викликів у сфері інформаційної безпеки.

У третьому розділі дипломної роботи розглянуто процес створення та перевірки методології, що має на меті протистояти фішинговим атакам. Основну увагу приділено розробці математичної моделі для точного розпізнавання фішингових

сайтів. Модель заснована на алгоритмі БФГШ, який виявився ефективним у класифікації та аналізі характеристик веб-сайтів для ідентифікації потенційних фішингових ресурсів.

Для тренування та оцінки алгоритмів було зібрано великий набір даних, отриманий з публічного ресурсу Kaggle. Даний набір включав в себе різноманітні атрибути веб-сайтів, що є важливими для розрізнення легітимних веб-сторінок від фішингових. В процесі аналізу даних було здійснено очищення від некоректних записів, а також нормалізацію та стандартизацію інформації для подальшої ефективної обробки моделлю.

Було розроблено трьохрівневу архітектуру системи, що включала модулі взаємодії з користувачем, бізнес-логіку обробки даних та рівень доступу до даних, що дозволило забезпечити гнучкість та масштабованість рішення.

Аналіз точності та надійності системи, заснований на результатів навчання моделі, продемонстрував високу ефективність у виявленні фішингових сайтів, що підтверджено значеннями ключових метрик продуктивності, зокрема AUC, F1 Score, Precision та Recall.

Були сформульовані рекомендації щодо практичного застосування розробленої системи. Окрему увагу було приділено методам оцінки ефективності системи, включаючи аналіз хибних позитивних та негативних результатів, що дозволяє постійно покращувати точність та надійність моделі протидії фішинговим атакам.

Для оптимального використання розробленої системи в практиці рекомендується забезпечити її інтеграцію з іншими заходами кібербезпеки в організації, такими як фільтрація веб-трафіку, системи виявлення вторгнень та антивірусний захист. Важливо також проводити регулярні навчання для співробітників з основ кібербезпеки, акцентуючи увагу на розпізнаванні фішингових атак. Використання розробленої системи разом з комплексними заходами безпеки та поінформованістю співробітників забезпечить надійний захист від фішингових атак та інших загроз кібербезпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Почепцов Г. Г. Соціальний інжиніринг : соціо- і психотехніки управління великими масами людей – Київ, Альтерпрес, 2010. – 254 с

2. Нікітська О. В. Методи запобігання фішингу та його різновиди. Вісник Академії адвокатури України. – 2017. – 133с.

3. Скибун Олександр. Фішинг та фішери в сучасному світі. Grail of Science, 2022.–264с.

4. Шнит Роман. Троянські програми у сучасному інформаційному просторі. Розвиток сучасної науки та освіти: реалії, проблеми якості, 2022.– 515с.

5. Загорняк В. Ю. Дослідження механізмів захисту від соціально-інженерних атак та розробка методів їх виявлення – Тернопіль: ТНТУ, 2023. – 77 с.

6. Mikhaleva, U. Pretexting and means to counter it. Herald of Dagestan State Technical University. Technical Sciences. 50. 2023. 116с. [Електронний ресурс] Режим доступу : https://doi.org/10.21822/2073-6185-2023-50-2-109-116 (дата звернення 05.03.2024).

7. Яковенко В. С.; Казеян Н. К. Соціальна інженерія в Інтернет-просторі. Вісник Чернівецького торговельно-економічного інституту. Економічні науки, 2016, 3-4: 126 с.

8. Counteracting socials engineering attacks : [Електронний ресурс] Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S1361372321001081#:~:text=According%20to%20the%20latest%20studies%2C,understanding%20of%20social%20engineering (дата звернення 05.03.2024).

9. Study on the psychology of social engineering cyberattacks: [Електронний ресурс] Режим доступу:https://www.mdpi.com/2076-3417/12/12/6042#:~:text=Recent%20studies%20have%20highlighted%20the,security%20measures%20for%20malicious%20acts (дата звернення 05.03.2024).

10. Social engineering attack using Kali Linux: [Електронний ресурс] Режим доступу: https://link.springer.com/article/10.1007/s42979-023-02321-y#:~:text=The%20study%20highlighted%20the%20need,risks%20of%20social%20engineering%20attacks (дата звернення 05.03.2024).

11. Detecting phishing websites using machine learning technique: [Електронний ресурс] Режим доступу : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8504731/#:~:text=In%20this%20study%2C%20the%20author,and%205800%20legitimate%20sites%2C%20respectively (дата звернення 05.03.2024).

12. Phishing attacks detection with machine learning: [Електронний ресурс] Режим доступу : https://ieeexplore.ieee.org/document/9214225 (дата звернення 05.03.2024).

13. Random Forest: [Електронний ресурс] Режим доступу: https://www.ibm.com/topics/random-forest (дата звернення 05.03.2024).

14. Decision Tree: [Електронний ресурс] Режим доступу : https://www.ibm.com/topics/decision-trees (дата звернення 05.03.2024).

15. Phishing detection using machine learning technique: [Електронний ресурс] Режим доступу : https://www.researchgate.net/publication/347308591_Phishing_Detection_Using_Machine_Learning_Technique (дата звернення 05.03.2024).

16. Support Vector Machines (SVM) : [Електронний ресурс] Режим доступу : https://scikit-learn.org/stable/modules/svm.html (дата звернення 05.03.2024).

17. Сєвєрінов О. В.; Хринов А. Г. Аналіз сучасних систем виявлення вторгнень. Системи обробки інформації, 2014, 6: 124 с.

18. Домарєв В. В.; Домарєв Д. В.; Гордієнко С. Б. Обгрунтування основних функцій системи управління інформаційною безпекою. Вісник Державного університету інформаційно-комунікаційних технологій, 2012, 10,№ 2: 104 с.

19. VPN (Virtual Private Network) [Электронный ресурс]. Режим доступа : https://www.wizcase.com/top-post/what-is-a-vpn-a-beginners-guide/?keyword=what%20is%20vpn%20connection&campaignID=15356342126&matcht

ype=e&adgroupID=137084173091&adpos=&extension=&kwd=aud-607267681815:kwd-13525356277&location=&geo=1030310&matchtype=e&device=&ad=570860194240&placement=&adposition=&gclid=CjwKCAjw7c2pBhAZEiwA88pOF7nectbRAfifHEtzBM7R0qi52cUXee1mDOZIbqV8r0ji9HdFRZ4f7hoCTgsQAvD_BwE (дата звернення 05.03.2024).

20. Остапов С. Е. Технологія захисту інформації : навчальний посібник / Х. : Вид. ХНЕУ, 2013. – 476 с.

21. Agbefu Ralph Edem; Hori Yoshiaki; Sakurai Kouichi. Domain information based blacklisting method for the detection of malicious webpages. International Journal of Cyber-Security and Digital Forensics, 2013, 2.2: 48c.

22. M. Akiyama, T. Yagi and M. Itoh, «Searching Structural Neighborhood of Malicious URLs to Improve Blacklisting,» 2011 IEEE/IPSJ International Symposium on Applications and the Internet, Munich, Germany, 2011. 10 с., [Електронний ресурс] Режим доступу : http://doi.org/10.1109/SAINT.2011.11 (дата звернення 05.03.2024).

23. Rao Routhu Srinivasa; Pais Alwyn Roshan. An enhanced blacklist method to detect phishing websites. In: Information Systems Security: 13th International Conference, ICISS 2017, Mumbai, India, December 16-20, 2017, Proceedings 13. Springer International Publishing, 2017. 333 p.

24. Koide Takashi, et al. To Get Lost is to Learn the Way: An Analysis of Multi-Step Social Engineering Attacks on the Web. IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, 2021, 104.1: 181 c.

25. Mills David. Analysis of a social engineering threat to information security exacerbated by vulnerabilities exposed through the inherent nature of social networking websites. In: 2009 Information Security Curriculum Development Conference. 2009. 141c.

26. Kubat Miroslav; Michalski Ryszard S. A review of machine learning methods. Machine Learning and Data Mining: Methods and Applications, 1998, 69 c.

27. Cliff Norman. Ordinal methods for behavioral data analysis. Psychology Press, 2014.79 c.

28. VISSERS Thomas. Large-scale analysis of attack techniques on Internet domain names. 2018. 216 c.

29. Метод Бройдена-Флетчера-Гольдфарба-Шанно (BFGS) : [Електронний ресурс] Режим доступу : https://matica.org.ua/metodichki-i-knigi-po-matematike/metody-mnogomernoi-bezuslovnoi-minimizatcii-v-p-severin/36-metod-broidena-nbsp-nbsp-fletchera-nbsp-nbsp-goldfarba-nbsp-nbsp-shanno (дата звернення 05.03.2024).

30. Марценюк В. П.; Мілян Н. В. Огляд методів оптимізації в машинному навчанні: градієнтний спуск та стохастичний градієнтний спуск. Збірник тез доповідей IX Міжнародної науково-технічної конференції молодих учених та студентів «Актуальні задачі сучасних технологій», 2020, 2: 45с.

31. Zabinsky Zelda B., et al. Random search algorithms. Department of Industrial and Systems Engineering, University of Washington, USA, 2009.16 с.

32. Andradóttir Sigrún. A review of random search methods. Handbook of Simulation Optimization, 2014, 277 с.

33. Mitnick, K. D., & Simon, W. L. The Art of Deception: Controlling the Human Element of Security. Wiley , 2002 , 368 с.

34. Hadnagy, C. Social Engineering: The Art of Human Hacking. Wiley, 2011, 322 с.

35. Kumar, A. Hands- On Artificial Intelligence for Cybersecurity: Implement smart AI systems for preventing cyber attacks and detecting threats and network anomalies. Packt Publishing, 2018, 342 с.

36. Malhorta, V. Artificial Intelligence for Security: Building a Safer World. CRC Press. 2019.

37. Nitzberg, B., & Pfleeger, C.P. Security in Computing (5th Edition). Pearson. 2016, 1043 с.

38. Ross, P. The Cybersecurity Handbook: A Practical Guide to Securing the Cyberworld. Wiley, 2018, 448 с.

39. Russel, S.J., & Norving, P. Artificial Intelligence: A Modern Approach (3rd edition). Pearson, 2009 , 1151 с.

40. Baier, J.A., & Whateley, R. Social Engineering in IT Security: Tools, Tactics, and Techniques. McGraw-Hill Education, 2015, 272 с.

41. Jones, B., & Bailey, C. Phishing Exposed. Syngress, 2010

42. Janczweski, L.J., & Colarik, A.M. Cyber Warfare and Cyber Terrorism. IGI Global, 2008, 564 c.

43. Dyer, J.,& Aitel, D. The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory. Wiley, 2008 , 914 c.

44. Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A.A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Computers & Security, 2012,

45. Cova, M., Kruegel, C., & Vigna, G. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In Proceedings of the 19th international conference on World wide web(pp.281-290). ACM. , 2010

46. SANS Institute. SEC487: Open-Source Intelligence (OSINT) Gathering and Analysis. SANS., 2018

47. Clarke, N. Social Engineering Penetration Testing: Executing Social Engineering Pen Tests, Assessments and Defense. Wiley, 2017, 390 c.

48. Kim, J., & Solomon, M. Fundamentals of Information Systems Security. Jones & Bartlett Learning, 2015, 548 c.

49. Schneier, B. Liars and Outliers: Enabling the Trust that Society Needs to Thrive. Wiley, 2012, 384 c.

50. Franklin, C. H., & Graeme, A. Building a Digital Forensic Laboratory: Establishing and Managing a Successful Facility. Syngress, 2006, 312 c.

ДОДАТКИ

ДОДАТОК А. Скрипти створення бази даних

```
USE [master]
GO
/****** Object:  Database [DB]    Script Date: 08.03.2024 10:24:59 ******/
CREATE DATABASE [DB]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'DB', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\DB.mdf' , SIZE = 8192KB , MAXSIZE =
UNLIMITED, FILEGROWTH = 65536KB )
 LOG ON
( NAME = N'DB_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\DB_log.ldf' , SIZE = 8192KB , MAXSIZE =
2048GB , FILEGROWTH = 65536KB )
 WITH CATALOG_COLLATION = DATABASE_DEFAULT, LEDGER = OFF
GO
ALTER DATABASE [DB] SET COMPATIBILITY_LEVEL = 160
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [DB].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [DB] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [DB] SET ANSI_NULLS OFF
GO
ALTER DATABASE [DB] SET ANSI_PADDING OFF
GO
ALTER DATABASE [DB] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [DB] SET ARITHABORT OFF
GO
ALTER DATABASE [DB] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [DB] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [DB] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [DB] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [DB] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [DB] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [DB] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [DB] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [DB] SET RECURSIVE_TRIGGERS OFF
```

```
GO
ALTER DATABASE [DB] SET  DISABLE_BROKER
GO
ALTER DATABASE [DB] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [DB] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [DB] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [DB] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [DB] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [DB] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [DB] SET HONOR_BROKER_PRIORITY OFF
GO
ALTER DATABASE [DB] SET RECOVERY SIMPLE
GO
ALTER DATABASE [DB] SET  MULTI_USER
GO
ALTER DATABASE [DB] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [DB] SET DB_CHAINING OFF
GO
ALTER DATABASE [DB] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )
GO
ALTER DATABASE [DB] SET TARGET_RECOVERY_TIME = 60 SECONDS
GO
ALTER DATABASE [DB] SET DELAYED_DURABILITY = DISABLED
GO
ALTER DATABASE [DB] SET ACCELERATED_DATABASE_RECOVERY = OFF
GO
ALTER DATABASE [DB] SET QUERY_STORE = ON
GO
ALTER DATABASE [DB] SET QUERY_STORE (OPERATION_MODE = READ_WRITE,
CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 30),
DATA_FLUSH_INTERVAL_SECONDS = 900, INTERVAL_LENGTH_MINUTES = 60,
MAX_STORAGE_SIZE_MB = 1000, QUERY_CAPTURE_MODE = AUTO,
SIZE_BASED_CLEANUP_MODE = AUTO, MAX_PLANS_PER_QUERY = 200,
WAIT_STATS_CAPTURE_MODE = ON)
GO
USE [DB]
GO
/****** Object:  Table [dbo].[Logs]    Script Date: 08.03.2024 10:24:59 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Logs](
        [LogsId] [int] IDENTITY(1,1) NOT NULL,
        [UsersId] [int] NULL,
```

```
	[EventNameShow] [nvarchar](max) NULL,
	[EventDate] [datetime] NULL,
PRIMARY KEY CLUSTERED
( [LogsId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Neural]    Script Date: 08.03.2024 10:24:59 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Neural](
	[NeuralId] [int] IDENTITY(1,1) NOT NULL,
	[NeuralNames] [nvarchar](200) NULL,
	[ScenariosId] [int] NULL,
	[NeuralFileModel] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
( [NeuralId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[PhishingData]    Script Date: 08.03.2024 10:24:59 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[PhishingData](
	[PhishingDataId] [int] NOT NULL,
	[Domain] [nvarchar](max) NULL,
	[Have_IP] [float] NULL,
	[Have_At] [float] NULL,
	[URL_Length] [float] NULL,
	[URL_Depth] [float] NULL,
	[Redirection] [float] NULL,
	[https_Domain] [float] NULL,
	[TinyURL] [float] NULL,
	[Prefix_Suffix] [float] NULL,
	[DNS_Record] [float] NULL,
	[Web_Traffic] [float] NULL,
	[Domain_Age] [float] NULL,
	[Domain_End] [float] NULL,
	[iFrame] [float] NULL,
	[Mouse_Over] [float] NULL,
	[Right_Click] [float] NULL,
	[Web_Forwards] [float] NULL,
	[Label] [bit] NULL,
	[UsersId] [int] NULL,
```

```
 CONSTRAINT [PK_PhishingData] PRIMARY KEY CLUSTERED
( [PhishingDataId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Scenarios]    Script Date: 08.03.2024 10:24:59 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Scenarios](
	[ScenariosId] [int] IDENTITY(1,1) NOT NULL,
	[ScenariosName] [nvarchar](250) NULL,
	[Description] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
( [ScenariosId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Users]    Script Date: 08.03.2024 10:24:59 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Users](
	[UsersId] [int] IDENTITY(1,1) NOT NULL,
	[FirstName] [nvarchar](50) NULL,
	[LastName] [nvarchar](50) NULL,
	[UsersName] [nvarchar](50) NULL,
	[UsersPassword] [nvarchar](50) NULL,
	[RoleId] [int] NULL,
	[Description] [nvarchar](1000) NULL,
	[Email] [nvarchar](150) NULL,
PRIMARY KEY CLUSTERED
(
	[UsersId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
USE [master]
GO
ALTER DATABASE [DB] SET  READ_WRITE
GO
	}
```

ДОДАТОК Б. Лістинги програмних засобів

Лістинг 1. Код класу «NeuralProvider»

```
using SocEngineeringApp.AppCode;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SocEngineeringApp.Providers {
 public class NeuralProvider {
   private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECTING"];
   public void InsertNeural(string NeuralNames, int ScenariosId, string NeuralFileModel) {
     string SqlString = "INSERT INTO Neural (NeuralNames, ScenariosId, NeuralFileModel) VALUES
(@NeuralNames, @ScenariosId, @NeuralFileModel)";

     using (SqlConnection conn = new SqlConnection(_ConnString)) {
      using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
       cmd.CommandType = CommandType.Text;
       cmd.Parameters.AddWithValue("@NeuralNames", NeuralNames);
       cmd.Parameters.AddWithValue("@ScenariosId", ScenariosId);
       cmd.Parameters.AddWithValue("@NeuralFileModel", NeuralFileModel);
       conn.Open();
       cmd.ExecuteNonQuery();
       conn.Close();
      }
     }
   }

   public List<Neural> GetAllNeural() {
    int i = 0;
    string SqlString = "SELECT * FROM Neural";
    List<Neural> listAllNeural = new List<Neural>();

    using (SqlConnection conn = new SqlConnection(_ConnString)) {
     using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
      conn.Open();
      using (SqlDataReader reader = cmd.ExecuteReader()) {
       while (reader.Read()) {
        Neural oneNeural = new Neural();
        oneNeural.Number = ++i;
        oneNeural.NeuralId = Convert.ToInt32(reader["NeuralId"]);
        oneNeural.NeuralNames = reader["NeuralNames"].ToString();
        oneNeural.ScenariosId = Convert.ToInt32(reader["ScenariosId"]);
        oneNeural.NeuralFileModel = reader["NeuralFileModel"].ToString();
        listAllNeural.Add(oneNeural);
       }
      }
      conn.Close();
```

```
    }
   }

   if (listAllNeural.Count == 0) {
    Neural noNeural = new Neural();
    noNeural.NeuralId = 0;
    noNeural.Message = NamesMy.NoDataNames.NoDataInNeural;
    listAllNeural.Add(noNeural);
   }
   return listAllNeural;
  }

  public Neural SelectedNeuralByNeuralId(int NeuralId) {
   string SqlString = "SELECT * FROM Neural WHERE NeuralId = @NeuralId";

   Neural oneNeural = new Neural();
   using (SqlConnection conn = new SqlConnection(_ConnString)) {
    using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
     cmd.Parameters.AddWithValue("@NeuralId", NeuralId);
     conn.Open();
     using (SqlDataReader reader = cmd.ExecuteReader()) {
      while (reader.Read()) {
       oneNeural.NeuralId = Convert.ToInt32(reader["NeuralId"].ToString());
       oneNeural.NeuralNames = reader["NeuralNames"].ToString();
       oneNeural.ScenariosId = Convert.ToInt32(reader["ScenariosId"]);
       oneNeural.NeuralFileModel = reader["NeuralFileModel"].ToString();
      }
     }
     conn.Close();
    }
   }
   return oneNeural;
  }

  public Neural SelectedNeuralByScenariosId(int ScenariosId) {
   string SqlString = "SELECT * FROM Neural WHERE ScenariosId = @ScenariosId";

   Neural oneNeural = new Neural();
   using (SqlConnection conn = new SqlConnection(_ConnString)) {
    using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
     cmd.Parameters.AddWithValue("@ScenariosId", ScenariosId);
     conn.Open();
     using (SqlDataReader reader = cmd.ExecuteReader()) {
      while (reader.Read()) {
       oneNeural.NeuralId = Convert.ToInt32(reader["NeuralId"].ToString());
       oneNeural.NeuralNames = reader["NeuralNames"].ToString();
       oneNeural.ScenariosId = Convert.ToInt32(reader["ScenariosId"]);
       oneNeural.NeuralFileModel = reader["NeuralFileModel"].ToString();
      }
     }
     conn.Close();
    }
```

```
      }
      return oneNeural;
    }

    public void UpdateNeural(string NeuralNames, int ScenariosId, string NeuralFileModel, int NeuralId)
{
      string SqlString = "UPDATE Neural SET NeuralNames=@NeuralNames,
ScenariosId=@ScenariosId, NeuralFileModel=@NeuralFileModel WHERE NeuralId=@NeuralId";

      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("@NeuralNames", NeuralNames);
          cmd.Parameters.AddWithValue("@ScenariosId", ScenariosId);
          cmd.Parameters.AddWithValue("@NeuralFileModel", NeuralFileModel);
          cmd.Parameters.AddWithValue("@NeuralId", NeuralId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public void DeleteNeuralByNeuralId(int NeuralId) {
      string SqlString = "DELETE FROM Neural WHERE NeuralId=@NeuralId";

      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.Parameters.AddWithValue("@NeuralId", NeuralId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

  }
}

public class Neural {
  private int _Number;
  private int _NeuralId;
  private string _NeuralNames;
  private int _ScenariosId;
  private string _NeuralFileModel;
  private string _Message;

  public Neural() {
    _Number = 0;
    _NeuralId = 0;
    _NeuralFileModel = String.Empty;
    _ScenariosId = 0;
```

```
    _NeuralNames = String.Empty;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int NeuralId {
    set { _NeuralId = value; }
    get { return _NeuralId; }
  }
  public string NeuralFileModel {
    set { _NeuralFileModel = value; }
    get { return _NeuralFileModel; }
  }
  public int ScenariosId {
    set { _ScenariosId = value; }
    get { return _ScenariosId; }
  }
  public string NeuralNames {
    set { _NeuralNames = value; }
    get { return _NeuralNames; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг 2. Код класу «PhishingDataProvider»

```
using CsvHelper.Configuration.Attributes;
using Microsoft.ML.Data;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SocEngineeringApp.Providers {
  internal class PhishingDataProvider {
    public void InsertPhishingDataP() {
    }

    public void GetAllPhishingDataP() {
    }
  }
}


public class PhishingData {
```

```
  [LoadColumn(0)]
  public string Domain { get; set; } //Назва домену вебсайту.
  [LoadColumn(1)]
  public float Have_IP { get; set; } //Вказує, чи містить URL IP-адресу замість доменного імені

  [LoadColumn(2)]
  public float Have_At { get; set; } //Чи містить URL символ '@'
  [LoadColumn(3)]
  public float URL_Length { get; set; } //Довжина URL
  [LoadColumn(4)]
  public float URL_Depth { get; set; } //Глибина URL, визначена кількістю слешів ('/')
  [LoadColumn(5)]
  public float Redirection { get; set; } //Наявність переадресацій
  [LoadColumn(6)]
  public float https_Domain { get; set; } //Перевірка, чи використовується HTTPS у доменній назві
  [LoadColumn(7)]
  public float TinyURL { get; set; } //Чи є URL скороченим (наприклад, через сервіси типу bit.ly)
  [LoadColumn(8)]
  [Name("Prefix/Suffix")]
  public float Prefix_Suffix { get; set; } //Вказує на наявність префіксів або суфіксів, які
відрізняються від загальноприйнятих
  [LoadColumn(9)]
  public float DNS_Record { get; set; } // Чи існує запис DNS для даного домену
  [LoadColumn(10)]
  public float Web_Traffic { get; set; } //Рейтинг веб-трафіку сайту.
  [LoadColumn(11)]
  public float Domain_Age { get; set; } // Вік домену
  [LoadColumn(12)]
  public float Domain_End { get; set; } //Час до закінчення реєстрації домену
  [LoadColumn(13)]
  public float iFrame { get; set; } //Використання тегів iFrame на сайті.
  [LoadColumn(14)]
  public float Mouse_Over { get; set; } //Використання скриптів для зміни стандартної поведінки
курсору миші
  [LoadColumn(15)]
  public float Right_Click { get; set; } //Чи дозволяє сайт використовувати правий клік миші
  [LoadColumn(16)]
  public float Web_Forwards { get; set; } // Використання автоматичних переадресацій на сайті
  [LoadColumn(17)]
  public bool Label { get; set; } //Мітка, яка вказує, чи є сайт фішинговим (1) або законним (0)
}


public class PhishingPrediction {
  [ColumnName("PredictedLabel")]
  public bool Prediction { get; set; }
  public float Probability { get; set; }
  public float Score { get; set; }
}
```

Лістинг 3. Код класу «PredictorForm»

```
using CsvHelper;
using SocEngineeringApp.AppCode;
```

```
using SocEngineeringApp.Providers;
using Microsoft.ML;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Runtime.Remoting.Metadata.W3cXsd2001;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace SocEngineeringApp.Forms.Controls {
  public partial class PredictorForm : Form {
    private ValidationMy _Validation = new ValidationMy();
    private Neural _SelectedNeural = new Neural();
    private MLContext context = new MLContext();
    private PredictionEngine<PhishingData, PhishingPrediction> predictionEngine;
    private NeuralProvider _NeuralProvider = new NeuralProvider();

    private ScenariosProvider _ScenariosProvider = new ScenariosProvider();
    private List<Scenarios> _ScenariosList = new List<Scenarios>();
    private bool _IsThemesLoad = false;

    private List<PhishingData> _PhishingDataL = new List<PhishingData>();
    List<PhishingData> _TestPhishingDatas = new List<PhishingData>();

    static readonly string _testDataPath = Path.Combine(Environment.CurrentDirectory, "Data",
"test_dataset.csv");


    public PredictorForm() {
      InitializeComponent();
      LoadAllDate();
    }

    private void LoadAllDate() {
      _ScenariosList = _ScenariosProvider.GetAllScenarios();
      ScenariosCBox.DataSource = _ScenariosList;
      ScenariosCBox.ValueMember = "ScenariosId";
      ScenariosCBox.DisplayMember = "ScenariosName";
      _IsThemesLoad = true;
      ScenariosCBox_SelectedValueChanged(ScenariosCBox, EventArgs.Empty);
    }

    private void ScenariosCBox_SelectedValueChanged(object sender, EventArgs e) {
      if (_IsThemesLoad && IsScenariosSelectedCorrect()) {
        _SelectedNeural = _NeuralProvider.SelectedNeuralByScenariosId(
```

```
        Convert.ToInt32(ScenariosCBox.SelectedValue));
       LoadData(_SelectedNeural.NeuralFileModel);
       _TestPhishingDatas = LoadPhishingData(_testDataPath);
      }
    }

    private void LoadData(string FilePath) {
      try {
       string localProj = Application.StartupPath + FilePath;
       // Define DataViewSchema for data preparation pipeline and trained model
       DataViewSchema modelSchema;

       // Load trained model
       ITransformer model = context.Model.Load(localProj, out modelSchema);
       // Evaluate the model
       // Use the model to make predictions
       predictionEngine = context.Model.CreatePredictionEngine<PhishingData,
PhishingPrediction>(model);
      } catch (Exception ex) {
       MessageBox.Show(ex.ToString());
      }
    }

    private void GetBtn_Click(object sender, EventArgs e) {
      if (IsScenariosSelectedCorrect()) {
       if (timer1.Enabled) {
        timer1.Enabled = false;
        GenBtn.Text = "Генерувати";
       } else {
        timer1.Enabled = true;
        GenBtn.Text = "Зупинити";
       }
      }
    }

    private void timer1_Tick(object sender, EventArgs e) {
      PredictAndDisplay();
    }

    private void PredictAndDisplay() {
      //список не порожній
      if (_TestPhishingDatas.Count == 0) {
       RaportTBox.Text = "Список тестових даних порожній.";
       return;
      }

      // Вибір випадкового запису зі списку
      var random = new Random();

      // Випадковий вибір, чи генерувати фішинговий випадок чи ні
      bool isPhishingExample = random.Next(0, 2) == 0;
```

```
    var randomPhishingData = new PhishingData {
      Domain = GenerateRandomDomain(),
      Have_IP = isPhishingExample ? random.Next(0, 2) : 0,
      Have_At = isPhishingExample ? random.Next(0, 2) : 0,
      URL_Length = isPhishingExample ? random.Next(0, 2) : 1,
      URL_Depth = isPhishingExample ? random.Next(0, 10) : 1,
      Redirection = isPhishingExample ? random.Next(0, 2) : 0,
      https_Domain = isPhishingExample ? random.Next(0, 2) : 0,
      TinyURL = isPhishingExample ? random.Next(0, 2) : 0,
      Prefix_Suffix = isPhishingExample ? random.Next(0, 2) : 0,
      DNS_Record = isPhishingExample ? random.Next(0, 2) : 0,
      Web_Traffic = isPhishingExample ? random.Next(0, 2) : 1,
      Domain_Age = isPhishingExample ? random.Next(0, 2) : 1,
      Domain_End = isPhishingExample ? random.Next(0, 2) : 1,
      iFrame = isPhishingExample ? random.Next(0, 2) : 0,
      Mouse_Over = isPhishingExample ? random.Next(0, 2) : 0,
      Right_Click = isPhishingExample ? random.Next(0, 2) : 1,
      Web_Forwards = isPhishingExample ? random.Next(0, 2) : 0
    };

    // прогноз за допомогою вже створеного predictionEngine, не передаючи Churn
    var prediction = predictionEngine.Predict(randomPhishingData);


    // Вивід даних клієнта
    var PhishingInfo = new StringBuilder();
    PhishingInfo.AppendLine($"Назва домену вебсайту: {randomPhishingData.Domain}");
    PhishingInfo.AppendLine($"URL IP-адреса замість доменного імені:
{randomPhishingData.Have_IP}");
    PhishingInfo.AppendLine($"Містить URL символ '@' ?: {randomPhishingData.Have_At}");
    PhishingInfo.AppendLine($"Довжина URL: {randomPhishingData.URL_Length}");
    PhishingInfo.AppendLine($"Глибина URL, визначена кількістю слешів ('/'):
{randomPhishingData.URL_Depth}");
    PhishingInfo.AppendLine($"Наявність переадресацій: {randomPhishingData.Redirection}");
    PhishingInfo.AppendLine($" HTTPS використовується у доменній назві:
{randomPhishingData.https_Domain}");
    PhishingInfo.AppendLine($"Чи є URL скороченим (наприклад, через сервіси типу bit.ly):
{randomPhishingData.TinyURL}");
    PhishingInfo.AppendLine($"Наявність префіксів або суфіксів, які відрізняються від
загальноприйнятих: {randomPhishingData.Prefix_Suffix}");
    PhishingInfo.AppendLine($"Існує запис DNS для даного домену?:
{randomPhishingData.DNS_Record}");
    PhishingInfo.AppendLine($"Рейтинг веб-трафіку сайту: {randomPhishingData.Web_Traffic}");
    PhishingInfo.AppendLine($"Вік домену: {randomPhishingData.Domain_Age}");
    PhishingInfo.AppendLine($"Час до закінчення реєстрації домену:
{randomPhishingData.Domain_End}");
    PhishingInfo.AppendLine($"Використання тегів iFrame на сайті: {randomPhishingData.iFrame}");
    PhishingInfo.AppendLine($"Використання скриптів для зміни стандартної поведінки курсору
миші: {randomPhishingData.Mouse_Over}");
    PhishingInfo.AppendLine($"Чи дозволяє сайт використовувати правий клік миші:
{randomPhishingData.Right_Click}");
```

```
    PhishingInfo.AppendLine($"Використання автоматичних переадресацій на сайті:
{randomPhishingData.Web_Forwards}");

    // Прогноз до виводу
    PhishingInfo.AppendLine("\r\n--- Прогнозування ---");
    PhishingInfo.AppendLine(prediction.Prediction ? "Виявлено фішінг" : "Не виявлено фішінг");
    PhishingInfo.AppendLine($"Ймовірність фішингової атаки: {prediction.Probability:P2}");
    PhishingInfo.AppendLine($"Бал: {Math.Abs(prediction.Score)}");

    // Виведіть результат у текстове поле
    RaportTBox.Text = PhishingInfo.ToString();
  }

  string GenerateRandomDomain() {
    var randoms = new Random();
    var randomPhishingData = _TestPhishingDatas[randoms.Next(_TestPhishingDatas.Count)];
    return randomPhishingData.Domain;
  }

  private void PredictBtn_Click(object sender, EventArgs e) {
    if (IsAllPhishingDataCorrect()) {
      var prediction = predictionEngine.Predict(new PhishingData {
        Domain = DomainTBox.Text,  Have_IP = Have_IPChBox.Checked ? 1.0f : 0.0f,
        Have_At = Have_AtChBox.Checked ? 1.0f : 0.0f, URL_Length =
float.Parse(URL_LengthTBox.Text),
        URL_Depth = float.Parse(URL_DepthTBox.Text),  Redirection = RedirectionChBox.Checked ?
1.0f : 0.0f,
        https_Domain = https_DomainChBox.Checked ? 1.0f : 0.0f,  TinyURL =
TinyURLChBox.Checked ? 1.0f : 0.0f,
        Prefix_Suffix = Prefix_SuffixChBox.Checked ? 1.0f : 0.0f,  DNS_Record =
DNS_RecordChBox.Checked ? 1.0f : 0.0f,
        Web_Traffic = float.Parse(Web_TrafficTBox.Text), Domain_Age =
float.Parse(Domain_AgeTBox.Text),
        Domain_End = float.Parse(Domain_EndTBox.Text), iFrame = iFrameChBox.Checked ? 1.0f : 0.0f,
        Mouse_Over = Mouse_OverChBox.Checked ? 1.0f : 0.0f,  Right_Click =
Right_ClickChBox.Checked ? 1.0f : 0.0f,
        Web_Forwards = Web_ForwardsChBox.Checked ? 1.0f : 0.0f
      });
      var PhishingInfo = new StringBuilder();
      // Прогноз до виводу
      PhishingInfo.AppendLine("\r\n--- Прогнозування ---");
      PhishingInfo.AppendLine(prediction.Prediction ? "Виявлено фішінг" : "Не виявлено фішінг");
      PhishingInfo.AppendLine($"Ймовірність фішингової атаки: {prediction.Probability:P2}");
      PhishingInfo.AppendLine($"Бал: {Math.Abs(prediction.Score)}");

      // Виведіть результат у текстове поле
      RaportTBox.Text = PhishingInfo.ToString();
    }
  }


  public static List<PhishingData> LoadPhishingData(string filePath) {
```

```
  using (var reader = new StreamReader(filePath))
  using (var csv = new CsvReader(reader, CultureInfo.InvariantCulture)) {
    var records = csv.GetRecords<PhishingData>().ToList();
    return records;
  }
}

private bool IsScenariosSelectedCorrect() {
  bool isCorrect = true;
  if (Convert.ToInt32(ScenariosCBox.SelectedValue) > 0) {
    ScenariosValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    ScenariosValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  return isCorrect;
}

private bool IsAllPhishingDataCorrect() {
  bool isCorrect = true;
  if (_Validation.IsDataEntering(DomainTBox.Text)) {
    DomainValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    DomainValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_Validation.IsDataConvertToDouble(URL_LengthTBox.Text)) {
    URL_LengthValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    URL_LengthValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_Validation.IsDataConvertToDouble(URL_DepthTBox.Text)) {
    URL_DepthValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    URL_DepthValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_Validation.IsDataConvertToDouble(Web_TrafficTBox.Text)) {
    Web_TrafficValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    Web_TrafficValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_Validation.IsDataConvertToDouble(Domain_AgeTBox.Text)) {
    Domain_AgeValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    Domain_AgeValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_Validation.IsDataConvertToDouble(Domain_EndTBox.Text)) {
    Domain_EndValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
```

```
      } else {
        Domain_EndValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

  }
}
```