МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПОЛІСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій,
обліку та фінансів
Кафедра комп'ютерних технологій
і моделювання систем

Кваліфікаційна робота
на правах рукопису

Супотніцького Володимира Володимировича
(прізвище, ім'я, по батькові здобувача освіти)

УДК 004.738.5:004.056.5

КВАЛІФІКАЦІЙНА РОБОТА

Метод виявлення соціотехнічних атак у кіберпросторі
(тема роботи)

КБ-23-м 125-«Кібербезпека та захист інформації»
(шифр і назва спеціальності)

Подається на здобуття освітнього ступеня магістр

кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____
(підпис, ініціали та прізвище здобувача вищої освіти)

Керівник роботи
Корченко Анна Олександрівна
(прізвище, ім'я, по батькові)
д.т.н, професор
(науковий ступінь, вчене звання)

Житомир – 2024

Висновок кафедри _____
за результатами попереднього захисту: _____

Протокол засідання кафедри _____
№ ____ від «_____» _____ 20____ р.

Завідувач кафедри _____

_____ _____ _____
(науковий ступінь, вчене звання) (підпис) (прізвище, ім'я, по батькові)
«_____» _____ 20____ р.

Результати захисту кваліфікаційної роботи

Здобувач вищої освіти _____ захистив (ла)
(прізвище ,ім'я, по батькові)
кваліфікаційну роботу з оцінкою:

сума балів за 100-бальною шкалою _____
за шкалою ECTS _____
за національною шкалою _____

Секретар ЕК

_____ _____ _____
(науковий ступінь, вчене звання) (підпис) (прізвище, ім'я, по батькові)

АНОТАЦІЯ

Супотніцький В. В. Метод виявлення соціотехнічних атак у кіберпросторі. - Кваліфікаційна робота на правах рукопису.

Кваліфікаційна робота на здобуття освітнього ступеня магістр за спеціальністю 125 – Кібербезпека. – Поліський національний університет, Житомир, 2023.

У кваліфікаційній роботі розроблено новий метод виявлення соціотехнічних атак, що базується на створенні інтерактивного тренувального середовища для навчання користувачів. Проведено аналіз існуючих видів фішингових атак та методів їх виявлення. Запропоновано підхід до навчання через емуляцію реальних сценаріїв атак у безпечному середовищі.

Розроблено веб-застосунок для реалізації запропонованого методу з використанням сучасного стеку технологій. Створено модульну архітектуру системи з компонентами емуляції робочого середовища, включаючи поштовий клієнт та веб-браузер. Реалізовано механізми адаптивного навчання та оцінки прогресу користувачів.

Проведено комплексне тестування системи, яке показало високу ефективність методу - точність виявлення фішингових атак досягла 87%. Розроблено практичні рекомендації щодо впровадження та використання системи в організаціях.

Ключові слова: соціотехнічні атаки, фішинг, соціальна інженерія, навчання користувачів, кібербезпека, веб-застосунок, тренувальне середовище, виявлення загроз.

SUMMARY

Supotnitskii V.V. Method for detecting socio-technical attacks in cyberspace. - Qualification work on the rights of the manuscript.

Qualification work for obtaining a master's degree in specialty 125 - Cybersecurity. - Polissia National University, Zhytomyr, 2023.

In the qualification work, a new method for detecting socio-technical attacks was developed, based on creating an interactive training environment for user education. An analysis of existing types of phishing attacks and methods of their detection was conducted. An approach to learning through emulation of real attack scenarios in a secure environment is proposed.

A web application was developed to implement the proposed method using a modern technology stack. A modular system architecture was created with workplace environment emulation components, including an email client and web browser. Mechanisms for adaptive learning and user progress assessment have been implemented.

Comprehensive system testing has shown high method effectiveness - phishing attack detection accuracy reached 87%. Practical recommendations for implementing and using the system in organizations have been developed.

Keywords: socio-technical attacks, phishing, social engineering, user training, cybersecurity, web application, training environment, threat detection.

ЗМІСТ

ВСТУП

Стрімкий розвиток інформаційних технологій та зростаюча цифровізація бізнес-процесів створюють нові виклики у сфері кібербезпеки. Соціотехнічні атаки стають все більш витонченими та становлять серйозну загрозу для організацій різного масштабу. За статистикою, понад 80% успішних кібератак використовують методи соціальної інженерії, що робить людський фактор критично важливим елементом системи інформаційної безпеки.

Актуальність теми дослідження обумовлена необхідністю розробки ефективних методів протидії соціотехнічним атакам через навчання користувачів розпізнаванню та запобіганню таким загрозам. Існуючі підходи до навчання часто не забезпечують формування стійких практичних навичок виявлення фішингових атак, що призводить до успішної реалізації зловмисниками своїх намірів.

Метою роботи є розробка методу виявлення соціотехнічних атак та його реалізація у вигляді тренувального веб-застосунку для підвищення рівня захищеності організацій від методів соціальної інженерії.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз існуючих видів соціотехнічних атак та методів їх виявлення;

– розробити метод інтерактивного навчання користувачів виявленню фішингових загроз;

– створити програмну реалізацію запропонованого методу у вигляді веб-застосунку;

– провести тестування розробленої системи та оцінити її ефективність;

– розробити рекомендації щодо впровадження та використання системи.

Об'єктом дослідження є процеси виявлення та запобігання соціотехнічним атакам в інформаційних системах.

Предметом дослідження є методи та засоби навчання користувачів виявленню фішингових атак.

Методи дослідження базуються на теорії інформаційної безпеки, методах машинного навчання, технологіях веб-розробки та принципах людино-машинної взаємодії.

Наукова новизна роботи полягає у розробці нового методу виявлення соціотехнічних атак, який, на відміну від існуючих, забезпечує формування практичних навичок через інтерактивне навчання в емульованому середовищі.

Практична цінність результатів полягає в можливості безпосереднього впровадження розробленої системи в процеси навчання персоналу організацій методам протидії соціотехнічним атакам.

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ СОЦІОТЕХНІЧНИХ АТАК У КІБЕРПРОСТОРІ

1.1 Поняття та види соціотехнічних атак

Соціотехнічні атаки становлять собою комплексні маніпулятивні дії, спрямовані на отримання конфіденційних даних користувачів шляхом психологічного впливу. Зловмисники застосовують різноманітні методи переконання та введення в оману для досягнення своїх цілей. Такі атаки базуються на експлуатації людської довіри та природної схильності до допомоги іншим. Під час соціотехнічних атак використовуються прийоми соціальної інженерії для маніпулювання поведінкою жертви. Злочинці ретельно вивчають психологічні особливості потенційних жертв для підвищення ефективності своїх дій. Вони створюють правдоподібні сценарії та легенди для переконання користувачів у достовірності своїх намірів. Фішинг є одним з найпоширеніших видів соціотехнічних атак, при якому зловмисники створюють підроблені веб-сайти та розсилають електронні листи. Такі листи зазвичай містять терміновий заклик до дії та посилання на фальшиві сторінки відомих сервісів. Користувачів спонукають ввести свої облікові дані, номери банківських карток та іншу конфіденційну інформацію. Фішингові сайти часто досконало копіюють дизайн легітимних ресурсів для підвищення довіри. Злочинці ретельно підбирають тексти повідомлень, щоб викликати емоційну реакцію [1 с. 3]. Вони експлуатують бренди відомих компаній для надання своїм діям достовірності.

Претекстинг передбачає створення вигаданого сценарію для отримання конфіденційних даних жертви через телефонний дзвінок чи особисте спілкування. Зловмисник представляється співробітником банку, податкової служби або іншої установи. Під час розмови він намагається зібрати максимум особистої інформації про жертву. Злочинці часто використовують психологічний тиск та маніпуляції для досягнення своєї мети. Вони можуть погрожувати блокуванням рахунку або іншими санкціями. Претекстинг вимагає ретельної підготовки легенди та вивчення специфіки роботи організації, від імені якої діє зловмисник.Квіпрокво базується на

обіцянці надати жертві певну послугу або вигоду в обмін на конфіденційні дані. Зловмисники пропонують безкоштовне програмне забезпечення, доступ до закритих ресурсів або грошову винагороду. Для отримання обіцяної вигоди користувачу необхідно надати особисту інформацію або встановити шкідливе ПЗ. Злочинці створюють привабливі пропозиції, які складно ігнорувати. Вони використовують природне бажання людей отримати щось безкоштовно [2 с. 37]. Квіпрокво часто поєднується з іншими видами соціотехнічних атак для підвищення ефективності.

Баітинг передбачає використання фізичних носіїв інформації для поширення шкідливого програмного забезпечення. Зловмисники залишають заражені USB-накопичувачі або диски у публічних місцях. Носії маркуються привабливими написами для спонукання людей їх підібрати та використати. При підключенні зараженого пристрою відбувається автоматичне встановлення шкідливого ПЗ. Баітинг експлуатує природну цікавість людей до знайдених речей. Метод особливо ефективний при націлюванні на конкретну організацію [3]. Вішинг є різновидом фішингу, що здійснюється через телефонні дзвінки з використанням технології VoIP. Зловмисники телефонують жертвам та представляються співробітниками банків чи інших установ. Під час розмови вони намагаються отримати конфіденційні дані під різними приводами. Злочинці використовують підміну телефонних номерів для маскування. Вони застосовують методи соціальної інженерії та психологічного тиску. Вішинг часто поєднується з розсилкою СМС-повідомлень для підвищення довіри жертви.

Соціотехнічні атаки через соціальні мережі набувають все більшого поширення завдяки популярності цих платформ. Зловмисники створюють фальшиві профілі та групи для збору особистих даних користувачів. Вони розсилають фішингові повідомлення та посилання на шкідливі ресурси. Злочинці використовують методи соціальної інженерії для встановлення довірчих відносин з потенційними жертвами. Вони збирають інформацію про користувачів з відкритих джерел для створення переконливих сценаріїв атаки. Соціальні мережі

надають широкі можливості для таргетованих атак на конкретних осіб.Шантаж та вимагання коштів через інтернет стають дедалі поширенішими видами соціотехнічних атак. Зловмисники отримують компрометуючі матеріали на жертву шляхом злому облікових записів або соціальної інженерії. Вони погрожують оприлюднити ці дані, якщо користувач не виконає їхні вимоги. Злочинці використовують психологічний тиск та залякування для досягнення своєї мети. Вони часто вимагають перерахування коштів у криптовалюті для ускладнення відстеження [4]. Такі атаки можуть мати серйозні наслідки для репутації та психологічного стану жертви.

Цільові соціотехнічні атаки на організації потребують ретельної підготовки та збору інформації про потенційні жертви. Зловмисники вивчають структуру компанії, її співробітників та бізнес-процеси. Вони створюють переконливі сценарії атак з використанням зібраних даних. Злочинці часто націлюються на конкретних працівників, які мають доступ до критично цінної інформації. Вони використовують різноманітні канали комунікації для проведення атак. Такі атаки можуть завдати значних фінансових та репутаційних збитків організації.

1.2 Аналіз існуючих методів виявлення соціотехнічних атак

Машинне навчання застосовується для автоматизованого виявлення фішингових атак через аналіз URL-адрес та вмісту веб-сторінок. Нейронні мережі навчаються на великих наборах даних для розпізнавання підозрілих патернів у структурі сайтів. Системи класифікації оцінюють множину параметрів, включаючи довжину домену, наявність спеціальних символів та схожість з відомими брендами. Алгоритми машинного навчання постійно вдосконалюються для підвищення точності виявлення нових видів атак. Методи глибокого навчання демонструють високу ефективність при аналізі візуальних елементів фішингових сайтів. Комбінування різних моделей машинного навчання дозволяє досягти кращих результатів у виявленні загроз [5 с. 290].

Поведінковий аналіз користувачів допомагає виявляти підозрілі дії, що можуть свідчити про соціотехнічні атаки. Системи моніторингу відстежують

аномальні патерни в діях співробітників організації. Збираються та аналізуються дані про час входу в систему, доступ до файлів та мережеву активність. Різкі зміни в звичній поведінці користувача можуть сигналізувати про компрометацію облікового запису. Автоматизовані системи оцінюють ризики на основі накопичених поведінкових метрик. Комплексний аналіз дозволяє виявляти складні багатоетапні атаки. Сигнатурні методи базуються на порівнянні вхідного трафіку з базою відомих шаблонів соціотехнічних атак. Системи виявлення збирають індикатори компрометації з різних джерел розвідки загроз. База сигнатур постійно оновлюється для врахування нових технік зловмисників. Правила фільтрації налаштовуються для блокування підозрілих з'єднань та файлів. Сигнатурний аналіз ефективний проти відомих методів атак з характерними ознаками [6 с. 730]. Комбінування сигнатур з іншими методами підвищує загальну ефективність захисту.

Лінгвістичний аналіз текстів допомагає виявляти фішингові повідомлення та шахрайські пропозиції. Алгоритми обробки природної мови оцінюють семантичні та синтаксичні особливості текстів. Системи аналізують наявність специфічних фраз та конструкцій, характерних для соціальної інженерії. Враховується емоційне забарвлення тексту та наявність маніпулятивних технік. Методи машинного навчання застосовуються для класифікації текстів за рівнем загрози. Постійне навчання на нових зразках підвищує точність лінгвістичного аналізу. Репутаційні системи використовують бази даних про відомі джерела загроз для блокування шкідливої активності. Збирається інформація про IP-адреси, домени та URL, помічені як зловмисні. Репутаційні оцінки формуються на основі даних від різних постачальників систем безпеки. Динамічні списки блокування оновлюються в режимі реального часу при виявленні нових загроз. Системи репутації допомагають запобігти доступу користувачів до фішингових ресурсів. Обмін даними між організаціями підвищує ефективність репутаційних механізмів.

Honeypot-системи створюють приманки для виявлення спроб соціотехнічних атак на організацію. Розгортаються фальшиві облікові записи та ресурси для

приваблення зловмисників. Спеціальні датчики фіксують усі спроби взаємодії з приманками. Аналіз зібраних даних дозволяє вивчати нові техніки та інструменти атакуючих. Honeypot-системи надають цінну інформацію для вдосконалення методів захисту. Розгортання мережі приманок ускладнює проведення цільових атак.

Системи песочниць забезпечують безпечний аналіз підозрілих файлів та посилань в ізольованому середовищі. Створюються віртуальні машини для запуску потенційно небезпечних об'єктів. Детально відстежується поведінка файлів для виявлення шкідливої активності. Автоматизовані системи аналізу генерують звіти про виявлені загрози. Песочниці дозволяють безпечно досліджувати нові види шкідливого програмного забезпечення. Постійне оновлення середовища аналізу забезпечує актуальність результатів. Навчання користувачів залишається одним з ключових методів протидії соціотехнічним атакам в організаціях. Регулярно проводяться тренінги з інформаційної безпеки для підвищення обізнаності співробітників. Персонал навчається розпізнавати ознаки фішингових атак та інших видів соціальної інженерії [7]. Моделюються реальні сценарії атак для відпрацювання правильних дій. Оцінюється ефективність навчання через періодичне тестування знань. Культура кібербезпеки формується через постійну роботу з персоналом.

Комплексний підхід передбачає поєднання різних методів виявлення для створення ешелонованої системи захисту. Технічні засоби доповнюються організаційними заходами та навчанням користувачів. Системи моніторингу забезпечують збір даних для постійного вдосконалення механізмів захисту. Процеси реагування на інциденти регулярно тестуються та оновлюються. Впроваджуються нові технології для протидії еволюції методів атак. Безперервний аналіз загроз дозволяє адаптувати захист під актуальні ризики.

Висновки до розділу 1

У результаті проведеного аналізу проблеми соціотехнічних атак у кіберпросторі було систематизовано основні види таких атак та методи їх

виявлення. З'ясовано, що соціотехнічні атаки становлять комплексну загрозу, яка поєднує методи психологічного впливу з технічними засобами для отримання несанкціонованого доступу до конфіденційних даних користувачів.

Детально розглянуто різні типи соціотехнічних атак, включаючи фішинг, претекстинг, квіпрокво, баітинг, вішинг та атаки через соціальні мережі. Показано, що кожен з цих видів має свої особливості реалізації та потребує специфічних методів виявлення та протидії. При цьому зловмисники постійно вдосконалюють свої техніки та комбінують різні підходи для підвищення ефективності атак.

Проаналізовано існуючі методи виявлення соціотехнічних атак, серед яких машинне навчання, поведінковий аналіз, сигнатурні методи, лінгвістичний аналіз, репутаційні системи, honeypot-системи та інші. Встановлено, що найбільш ефективним є комплексний підхід, який поєднує технічні засоби захисту з навчанням користувачів та організаційними заходами.

Виходячи з проведеного аналізу, можна стверджувати, що розробка нових методів виявлення соціотехнічних атак залишається актуальним завданням, особливо в контексті навчання користувачів розпізнаванню ознак потенційних загроз. Перспективним напрямком є створення інтерактивних навчальних систем, які дозволяють користувачам отримати практичний досвід виявлення різних видів соціотехнічних атак у безпечному середовищі.

РОЗДІЛ 2 РОЗРОБКА МЕТОДУ ВИЯВЛЕННЯ СОЦІОТЕХНІЧНИХ АТАК ТА ТРЕНУВАЛЬНОГО ВЕБ-ЗАСТОСУНКУ

2.1 Опис запропонованого методу виявлення соціотехнічних атак

Розроблений метод базується на створенні тренувального середовища для моделювання типових сценаріїв соціотехнічних атак. Користувачі отримують доступ до емуляції типового робочого оточення з веб-браузером та поштовим клієнтом. Система генерує навчальні сценарії з різними видами фішингових листів та підроблених веб-сайтів. Всі елементи інтерфейсу максимально наближені до реальних сервісів для забезпечення реалістичності навчання. Програмне забезпечення відстежує дії користувача та надає зворотній зв'язок щодо правильності прийнятих рішень. Метод передбачає поступове підвищення складності завдань у міру набуття користувачем досвіду. База навчальних сценаріїв формується на основі аналізу реальних випадків соціотехнічних атак. Кожен сценарій містить набір індикаторів для оцінки потенційної загрози. Система враховує різні техніки соціальної інженерії, які використовуються зловмисниками. Сценарії регулярно оновлюються для відображення нових методів атак. Для кожного типу загроз створюються варіанти з різним рівнем складності виявлення [8]. Навчальні матеріали адаптуються під специфіку конкретної організації.

Оцінка дій користувача здійснюється за багатокритеріальною системою показників. Враховується швидкість виявлення потенційних загроз та правильність класифікації типу атаки. Система фіксує послідовність дій користувача при аналізі підозрілого контенту. Окремо оцінюється здатність розпізнавати різні індикатори фішингових атак. Результати кожної сесії зберігаються для відстеження прогресу навчання. Формується детальна статистика успішності виявлення різних видів загроз. Інтерактивний режим навчання забезпечує занурення користувача в процес виявлення атак. Система миттєво реагує на дії користувача, надаючи підказки та пояснення. При виявленні помилок демонструються правильні підходи до аналізу загроз [9]. Користувач отримує можливість детально вивчити механізми

конкретних атак. Передбачено режим покрокового розбору навчальних сценаріїв. Система заохочує дослідницький підхід до виявлення загроз.

Метод включає механізми адаптації складності навчальних завдань. Початковий рівень визначається базовою оцінкою навичок користувача. Система автоматично підбирає оптимальну послідовність сценаріїв для навчання. Складність поступово зростає при успішному проходженні попередніх рівнів. Враховуються індивідуальні особливості сприйняття навчального матеріалу. Користувач може самостійно регулювати темп навчання [10]. Модуль генерації навчального контенту створює унікальні варіанти фішингових листів та сайтів. Використовуються шаблони реальних соціотехнічних атак з модифікованим вмістом. Система комбінує різні елементи для створення правдоподібних сценаріїв. Генерація контенту враховує актуальні тренди в методах соціальної інженерії. Створюються варіанти з різним ступенем очевидності загроз [11]. Контент адаптується під специфіку цільової аудиторії.

Аналітичний модуль забезпечує збір та обробку даних про результативність навчання. Формуються індивідуальні профілі користувачів з оцінкою їх прогресу. Система виявляє типові помилки та складні для розпізнавання види атак. Генеруються рекомендації щодо додаткового навчання за проблемними напрямками. Статистика використовується для вдосконалення навчальних сценаріїв [12]. Результати аналізу допомагають оптимізувати процес навчання. Розроблений метод передбачає регулярне оновлення бази знань про соціотехнічні атаки. Система інтегрується з джерелами даних про нові види загроз. Навчальні сценарії доповнюються актуальними прикладами атак. Оновлюються індикатори для виявлення підозрілої активності. База знань розширюється на основі зворотного зв'язку від користувачів. Механізми виявлення адаптуються під еволюцію методів атак.

Інтеграція з корпоративними системами безпеки розширює можливості навчання. Тренувальне середовище може використовувати реальні приклади атак на організацію. Система враховує специфічні загрози для конкретного бізнесу.

Результати навчання допомагають оцінити загальний рівень захищеності від соціотехнічних атак [13 с. 98]. Формуються рекомендації щодо вдосконалення корпоративних політик безпеки. Метод стає частиною комплексної системи протидії загрозам.

2.2 Архітектура тренувального веб-застосунку

Тренувальний веб-застосунок побудований на основі сучасної модульної архітектури з використанням фреймворку Svelte. Кожен компонент системи розроблений як незалежний модуль зі своїм функціоналом та інтерфейсом. Взаємодія між модулями здійснюється через чітко визначені програмні інтерфейси. Модульна структура забезпечує гнучкість при розширенні функціональності системи. Компоненти можуть бути легко модифіковані або замінені без впливу на роботу інших частин застосунку [14]. Така архітектура спрощує процес тестування та налагодження окремих модулів.

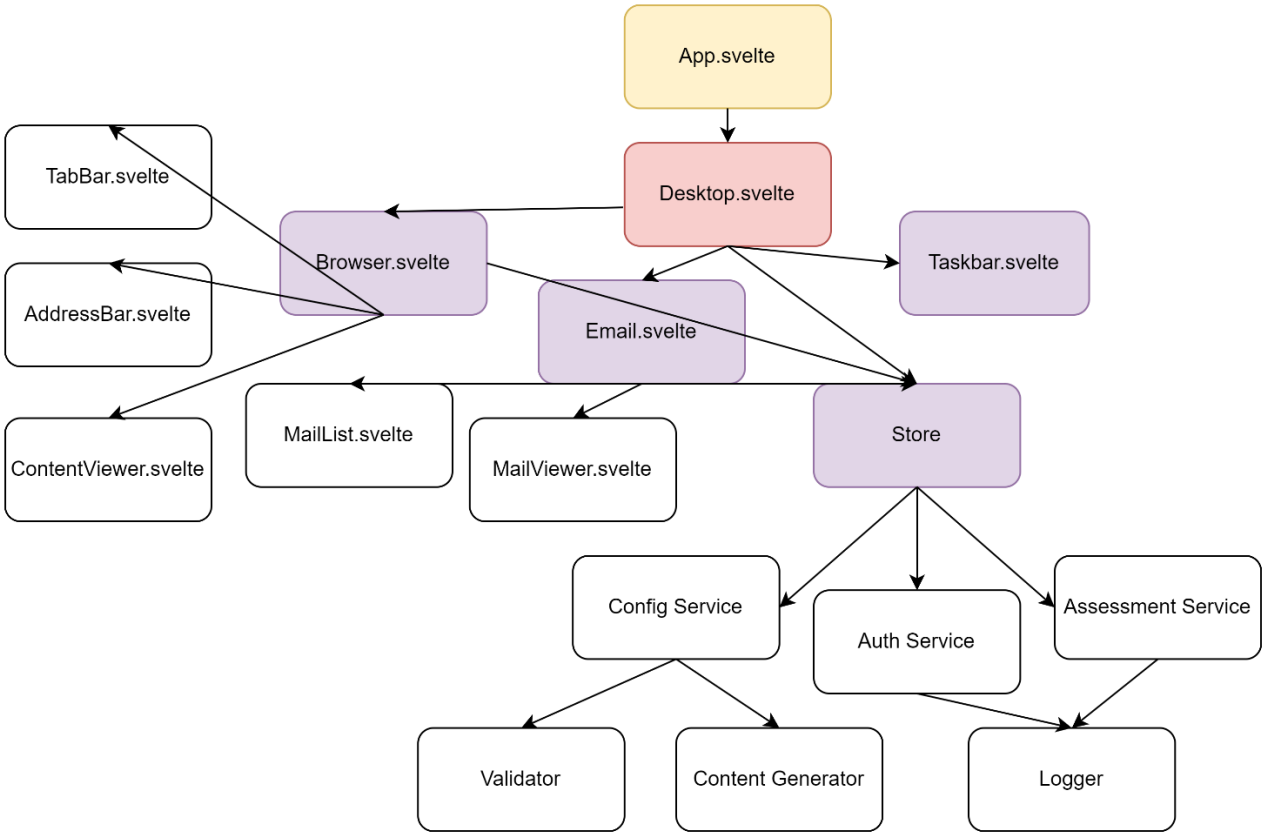

Рисунок 2.1 - Архітектура тренувального веб-застосунку

Основним елементом застосунку є модуль емуляції робочого столу користувача. Цей модуль забезпечує відображення інтерфейсу, схожого на типову операційну систему. Панель запуску програм дозволяє відкривати різні додатки тренувального середовища. Реалізовано функції керування вікнами відкритих програм та перемикання між ними. Годинник та інші елементи інтерфейсу роблять середовище більш реалістичним. Модуль підтримує збереження стану робочого простору між сесіями. Модуль емуляції веб-браузера відтворює роботу сучасного браузера з основними функціями навігації. Реалізована адресна строка для введення URL та перехід за посиланнями. Користувачі можуть працювати з декількома вкладками одночасно. Браузер підтримує відображення HTML-контенту з базовим CSS-оформленням [15]. Передбачена можливість завантаження файлів та взаємодії з веб-формами. Інтерфейс максимально наближений до реальних браузерів.

Поштовий клієнт реалізований як окремий модуль з можливістю перегляду та сортування листів. Система підтримує відображення листів у різних форматах з HTML-розміткою. Користувачі можуть переміщатися між папками та виконувати базові операції з листами. Реалізовано функції пошуку та фільтрації повідомлень за різними критеріями. Модуль забезпечує коректне відображення вкладених файлів та посилань. Інтерфейс розроблений за принципами сучасних поштових клієнтів. Модуль конфігурації дозволяє гнучко налаштовувати параметри навчальних сценаріїв. Адміністратори можуть створювати нові сценарії через JSON-файли конфігурації. Система підтримує визначення різних типів фішингових атак та їх параметрів. Можливе налаштування послідовності появи навчальних матеріалів. Конфігурація включає параметри оцінювання дій користувача [16]. Модуль забезпечує валідацію налаштувань для запобігання помилкам.

Система оцінювання реалізована як окремий модуль з гнучкими критеріями аналізу. Фіксуються всі значущі дії користувача під час роботи з навчальними сценаріями. Кожна дія оцінюється відповідно до визначених правил та метрик. Модуль підтримує різні схеми нарахування балів за виявлення загроз. Результати зберігаються в базі даних для подальшого аналізу. Передбачена можливість експорту статистики в різних форматах. Модуль генерації контенту створює унікальні варіації навчальних матеріалів на основі шаблонів. Система використовує бібліотеку готових елементів для створення фішингових листів. Контент динамічно формується з урахуванням обраного сценарію атаки. Підтримується локалізація матеріалів різними мовами. Модуль забезпечує консистентність згенерованого контенту. Передбачені механізми валідації створених матеріалів.

Сховище даних реалізоване з використанням локальної бази даних для збереження прогресу. Система зберігає профілі користувачів та історію їх навчання. База даних містить інформацію про всі спроби виявлення загроз. Реалізовано механізми резервного копіювання та відновлення даних. Структура

бази оптимізована для швидкого доступу до інформації. Підтримується можливість експорту та імпорту даних.

Система аутентифікації забезпечує розмежування доступу до функцій застосунку. Реалізована базова аутентифікація користувачів за логіном та паролем. Підтримуються різні ролі з різним рівнем доступу до функціоналу. Система відстежує активні сесії користувачів та їх термін дії. Передбачені механізми відновлення забутих паролів. Всі операції з обліковими даними журналюються. Інтерфейс адміністрування надає інструменти для управління системою та моніторингу. Адміністратори можуть переглядати статистику використання та результати навчання. Реалізовані функції управління користувачами та їх правами доступу [17 с. 912]. Система дозволяє налаштовувати параметри навчальних сценаріїв. Передбачений перегляд системних логів та діагностика помилок. Інтерфейс забезпечує зручне керування всіма аспектами роботи застосунку.

2.2.1 Вибір фреймворку Svelte для розробки

Фреймворк Svelte обраний для розробки веб-застосунку через його унікальний підхід до створення інтерактивних інтерфейсів. На відміну від інших фреймворків, Svelte виконує компіляцію компонентів на етапі збірки проекту. Це дозволяє генерувати оптимізований JavaScript-код без зайвого навантаження на браузер. Компоненти Svelte перетворюються в ефективний нативний код. Розмір підсумкового бандла значно менший порівняно з React або Vue. Швидкість роботи застосунку помітно вища завдяки відсутності віртуального DOM. Реактивність у Svelte реалізована на рівні компіляції, що спрощує розробку інтерактивних компонентів. Розробнику не потрібно явно вказувати залежності для оновлення інтерфейсу [18]. Зміни стану автоматично відстежуються та застосовуються до DOM. Синтаксис Svelte інтуїтивно зрозумілий та близький до звичайного HTML і JavaScript. Компоненти описуються в єдиному файлі з розміткою, стилями та логікою. Такий підхід покращує структуру коду та спрощує його підтримку.

Система стилізації Svelte надає зручні інструменти для створення модульних стилів. Стилі автоматично обмежуються областю видимості компонента без

додаткових налаштувань. Підтримується вкладеність селекторів та динамічна генерація класів [19]. CSS-правила оптимізуються під час компіляції для мінімізації дублювання. Можливе використання препроцесорів та постпроцесорів для розширення функціональності. Стилі легко перевикористовуються між компонентами.

Система подій Svelte забезпечує ефективну комунікацію між компонентами застосунку. Підтримуються вбудовані та користувацькі події з можливістю передачі даних. Події автоматично делегуються для оптимізації продуктивності. Обробники подій компілюються в оптимізований код без зайвих обгорток. Модифікатори подій дозволяють легко налаштовувати їх поведінку. Події інтегруються з системою реактивності фреймворку. Svelte надає вбудовані засоби для анімації та переходів між станами інтерфейсу. Анімації описуються декларативно з використанням простого API. Підтримуються різні типи переходів з налаштуванням тривалості та кривих анімації [20]. Анімації оптимізовані для плавного відтворення навіть на слабких пристроях. Можлива синхронізація кількох анімацій та створення складних послідовностей. Анімації не впливають на продуктивність основного застосунку.

Інструменти розробника Svelte спрощують процес створення та налагодження компонентів. Розширення для редакторів коду забезпечують підсвічування синтаксису та автодоповнення. Вбудований режим розробки надає додаткову інформацію про роботу компонентів [21]. Помилки компіляції супроводжуються зрозумілими повідомленнями та підказками. Гаряче перезавантаження прискорює цикл розробки. Налагодження компонентів не потребує додаткових інструментів.

Екосистема Svelte включає набір готових компонентів та бібліотек для типових задач. Доступні рішення для маршрутизації, управління станом, форм та інших функцій. Компоненти легко інтегруються між собою завдяки єдиним принципам розробки [22]. Спільнота активно розвиває екосистему та створює нові

інструменти. Документація фреймворку детально описує всі можливості та кращі практики. Підтримка TypeScript забезпечує надійність коду.

Процес збірки застосунку оптимізований для досягнення максимальної продуктивності. Vite забезпечує швидкий старт сервера розробки та миттєве оновлення змін. Підтримується автоматичне розділення коду на чанки для оптимізації завантаження. Збірка включає мініфікацію та стиснення ресурсів. Генеруються різні версії бандлів для різних браузерів [23]. Процес збірки повністю налаштовується під потреби проекту.

2.2.2 Структура та компоненти веб-застосунку

Архітектура веб-застосунку побудована за принципом ієрархічної організації компонентів. Кореневий компонент App.svelte виступає контейнером для всіх інших елементів системи. Глобальний стан застосунку керується через контекст Svelte, доступний всім компонентам. Система маршрутизації забезпечує навігацію між різними екранами застосунку. Компоненти згруповані за функціональним призначенням у відповідні директорії. Структура проекту оптимізована для зручної навігації та пошуку потрібних файлів. Компонент робочого столу Desktop.svelte реалізує базовий інтерфейс операційної системи. Панель запуску додатків розташована внизу екрану та містить іконки доступних програм. Реалізовано систему управління вікнами з можливістю переміщення та зміни розмірів. Годинник у правому куті оновлюється в реальному часі. Фонове зображення робочого столу можна змінювати через налаштування [24]. Всі елементи інтерфейсу підтримують адаптивне відображення на різних пристроях.

Модуль браузера Browser.svelte складається з кількох взаємопов'язаних компонентів. Панель адреси містить поле введення URL та кнопки навігації. Система вкладок дозволяє працювати з декількома сайтами одночасно. Компонент відображення контенту підтримує базову HTML-розмітку та стилі. Реалізовано емуляцію завантаження сторінок з індикатором прогресу. Історія переходів зберігається для навігації між сторінками. Поштовий клієнт Email.svelte відображає список повідомлень та їх вміст. Бічна панель містить папки для організації листів

за категоріями [25]. Реалізовано функції позначення листів як прочитаних та видалення. Пошукова система дозволяє фільтрувати повідомлення за різними критеріями. Підтримується сортування листів за датою та іншими параметрами. Компонент перегляду листа коректно відображає HTML-форматування та вкладення. Система оцінювання Assessment.svelte відстежує дії користувача та формує звіти. Компонент накопичує статистику взаємодії з навчальними матеріалами. Кожна спроба виявлення фішингу фіксується з детальною інформацією. Результати аналізуються для визначення слабких місць у навчанні. Генеруються рекомендації щодо подальшого вивчення матеріалу. Вся статистика доступна через панель адміністрування. Компонент конфігурації Config.svelte забезпечує налаштування параметрів системи. Адміністратор може редагувати JSON-файли з навчальними сценаріями. Реалізовано валідацію введених даних для запобігання помилкам. Зміни конфігурації застосовуються без перезавантаження застосунку [26]. Підтримується створення резервних копій налаштувань. Інтерфейс надає підказки щодо формату конфігураційних файлів.

Модуль генерації контенту Generator.svelte створює унікальні навчальні матеріали. Компонент використовує шаблони для формування фішингових листів та сайтів. Підтримується локалізація контенту різними мовами через систему перекладів. Генерація враховує поточний рівень складності навчання. Створені матеріали проходять перевірку на коректність відображення. Контент оптимізується для різних розмірів екрану. Система аутентифікації Auth.svelte керує доступом користувачів до застосунку. Компонент форми входу підтримує валідацію введених даних. Паролі зберігаються в захищеному вигляді з використанням хешування. Реалізовано механізм відновлення забутого пароля через email. Сесії користувачів автоматично завершуються після періоду неактивності. Всі спроби входу фіксуються в системному журналі.

Адміністративна панель Admin.svelte надає інструменти управління системою. Компонент відображає статистику використання та результати навчання. Реалізовано функції управління користувачами та їх правами. Доступний

перегляд системних логів та діагностика помилок. Адміністратор може налаштовувати параметри навчальних сценаріїв [27]. Інтерфейс забезпечує зручну навігацію по всім функціям управління.

2.2.3 Конфігурація вмісту модулів пошти та браузера

Система конфігурації використовує JSON-файли для опису навчальних сценаріїв та їх параметрів. Файл mails.json містить набір фішингових листів з різними видами соціотехнічних атак. Кожен лист включає поля для автора, теми, тексту повідомлення та додаткової інформації. Параметр danger визначає, чи є лист частиною атаки. Підтримується використання HTML-розмітки для стилізації тексту листів. Рекомендації щодо виявлення загроз зберігаються в окремому полі recommendation.

Файл tabs.json визначає структуру та вміст веб-сторінок для емуляції браузера. Кожна вкладка описується набором полів, включаючи назву, іконку та URL-адресу. Поле siteCode містить HTML-код для відображення вмісту сторінки. Параметр danger маркує фішингові сайти в навчальних сценаріях. Підтримується базове CSS-оформлення для створення реалістичного інтерфейсу. Додаткова інформація про загрози зберігається в полі recommendation. Система підтримує створення взаємопов'язаних сценаріїв між поштою та браузером. Фішингові листи можуть містити посилання на підроблені веб-сайти в системі. URL-адреси в листах автоматично зіставляються з відповідними вкладками браузера. Перехід за посиланнями відстежується для оцінки дій користувача. Сценарії можуть включати кілька етапів соціотехнічної атаки [28]. Система зберігає зв'язки між різними елементами сценарію.

Конфігурація дозволяє налаштовувати рівні складності для різних сценаріїв. Початкові сценарії містять очевидні ознаки фішингових атак для навчання базовим навичкам. Складніші варіанти вимагають більш ретельного аналізу для виявлення загроз. Система підтримує поступове підвищення складності в процесі навчання. Кожен сценарій має встановлений рівень складності від 1 до 5. Параметри складності враховуються при генерації нових сценаріїв.

Механізм локалізації забезпечує підтримку різних мов у навчальних матеріалах. Тексти листів та веб-сторінок зберігаються з прив'язкою до мовних кодів. Система автоматично підбирає контент відповідно до обраної мови інтерфейсу. Підтримується додавання нових перекладів через конфігураційні файли. Локалізація охоплює всі текстові елементи сценаріїв. Переклади зберігають форматування та структуру оригінального тексту.

Валідація конфігураційних файлів запобігає помилкам у навчальних сценаріях. Перевіряється наявність всіх обов'язкових полів та коректність їх значень. Система контролює унікальність ідентифікаторів елементів сценаріїв. HTML-код перевіряється на відсутність заборонених тегів та атрибутів. Валідатор перевіряє коректність посилань між елементами сценаріїв. Помилки конфігурації відображаються в логах системи. Резервне копіювання конфігурації забезпечує збереження налаштованих сценаріїв. Система створює резервні копії при кожній зміні конфігураційних файлів. Підтримується відновлення попередніх версій конфігурації. Резервні копії зберігаються з часовими мітками для відстеження змін. Експорт конфігурації дозволяє переносити сценарії між системами. Механізм відновлення перевіряє цілісність резервних копій.

Система версіонування відстежує зміни в конфігураційних файлах. Кожна версія конфігурації отримує унікальний ідентифікатор. Зберігається історія змін з інформацією про автора та час модифікації. Підтримується порівняння різних версій конфігурації. Можливе відновлення окремих елементів з попередніх версій. Версіонування спрощує відлагодження проблем конфігурації. Автоматичне оновлення забезпечує синхронізацію конфігурації на всіх клієнтах. Зміни в конфігураційних файлах поширюються без перезавантаження застосунку. Клієнти періодично перевіряють наявність оновлень конфігурації. Завантажуються тільки змінені частини конфігурації для економії трафіку. Процес оновлення не перериває роботу користувачів. Система відстежує статус синхронізації на кожному клієнті.

Механізм міграції забезпечує сумісність при зміні структури конфігурації. Система автоматично конвертує конфігураційні файли в актуальний формат.

Підтримується зворотна сумісність зі старими версіями конфігурації [29]. Процес міграції зберігає всі налаштовані сценарії та їх параметри. Помилки міграції обробляються з можливістю ручного виправлення. Історія міграцій зберігається для відстеження змін формату.

2.3 Алгоритм роботи тренувального веб-застосунку

Тренувальний веб-застосунок функціонує на основі послідовності взаємопов'язаних кроків обробки дій користувача. Початкова ініціалізація системи включає завантаження конфігураційних файлів та налаштування компонентів. Після авторизації користувач отримує доступ до емульованого робочого столу з набором додатків. Система відстежує всі дії користувача для подальшого аналізу. Модулі застосунку обмінюються даними через централізований механізм подій. Стан системи постійно синхронізується для забезпечення узгодженості даних.

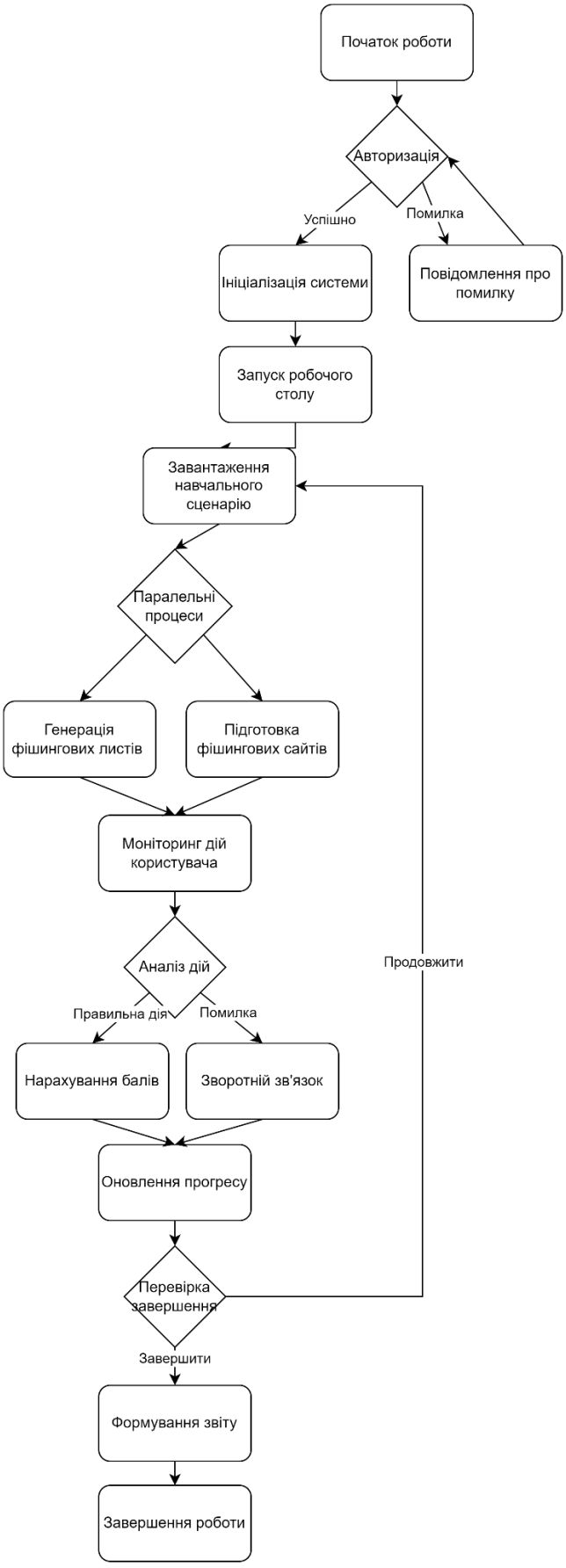

Рисунок 2.2 - Алгоритм роботи тренувального веб-застосунку

При запуску поштового клієнта система генерує набір навчальних листів згідно поточного сценарію. Листи створюються з використанням шаблонів та параметрів з конфігураційних файлів. Механізм генерації забезпечує унікальність кожного тренувального сеансу. Фішингові листи містять різні індикатори загроз відповідно до рівня складності. Система фіксує час перегляду та реакцію користувача на кожен лист. Дії користувача порівнюються з очікуваною поведінкою для оцінки результатів. Браузерний модуль динамічно завантажує веб-сторінки при переході за посиланнями. Система емулює реальну роботу браузера з підтримкою навігації та вкладок. Фішингові сайти створюються на основі шаблонів з різними ознаками шахрайства. Всі переходи між сторінками відстежуються для аналізу поведінки користувача. Модуль перевіряє спроби введення конфіденційних даних на підозрілих сайтах. Браузер взаємодіє з поштовим клієнтом для створення комплексних сценаріїв.

Система оцінювання аналізує кожну дію користувача в реальному часі. Враховується швидкість реакції на потенційні загрози та правильність їх класифікації. Бали нараховуються за успішне виявлення фішингових листів та сайтів. Помилкові рішення призводять до зниження загальної оцінки. Результати зберігаються для формування статистики навчання. Система генерує персоналізовані рекомендації щодо покращення навичок. Механізм зворотного зв'язку надає користувачу пояснення після кожної спроби виявлення загроз. При правильній ідентифікації фішингу система показує детальний розбір використаних індикаторів. У випадку помилки демонструються пропущені ознаки шахрайства та правильний спосіб аналізу. Користувач отримує підказки щодо типових патернів соціотехнічних атак. Система зберігає найбільш поширені помилки для формування навчальних рекомендацій. Зворотній зв'язок адаптується під поточний рівень користувача.

Адаптивна система складності автоматично регулює рівень завдань. Початковий рівень визначається на основі тестування базових навичок. Складність поступово зростає при успішному проходженні попередніх сценаріїв. Система

враховує статистику помилок при підборі нових завдань. При виникненні труднощів рівень тимчасово знижується. Алгоритм адаптації забезпечує оптимальний темп навчання.

Модуль статистики накопичує дані про всі навчальні сесії користувача. Зберігається детальна інформація про кожну спробу виявлення загроз. Система відстежує прогрес користувача за різними типами фішингових атак. Генеруються графіки та звіти для аналізу ефективності навчання. Статистика використовується для оптимізації навчальних сценаріїв. Дані агрегуються для оцінки загальної результативності системи. Механізм сповіщень інформує користувача про нові навчальні матеріали та результати. Система надсилає нагадування про необхідність регулярних тренувань. Користувач отримує повідомлення про досягнення навчальних цілей. При виявленні тривалих перерв надсилаються мотиваційні сповіщення. Сповіщення налаштовуються відповідно до преференцій користувача. Система відстежує реакцію на різні типи повідомлень.

Алгоритм завершення роботи забезпечує коректне збереження всіх даних. Система фіксує час завершення сесії та підсумкові результати. Незавершені сценарії зберігаються для продовження в наступній сесії. Виконується резервне копіювання критично важливих даних [30]. Користувач отримує підсумковий звіт про результати тренування. Система оновлює рекомендації перед завершенням роботи.

2.4 Особливості реалізації модулів виявлення фішингових листів та веб-сайтів

Модуль аналізу електронних листів використовує багаторівневу систему перевірки контенту. Кожен лист проходить первинну оцінку на наявність базових ознак фішингу. Перевіряються заголовки, відправники та структура повідомлення. Система аналізує текстовий вміст на наявність маніпулятивних конструкцій. URL-адреси в листах перевіряються на належність до відомих фішингових доменів. Механізм оцінки враховує комбінації різних індикаторів загроз.

Аналіз доменних імен включає перевірку на схожість з брендами відомих компаній. Система виявляє заміну символів та використання схожих доменів вищого рівня. Перевіряється дата реєстрації домену та інформація про власника. Враховується використання безкоштовних хостингів та підозрілих реєстраторів. Домени порівнюються з базою даних відомих фішингових ресурсів. Оцінюється контекст використання домену в листі. Перевірка HTML-розмітки листів виявляє спроби маскування шкідливого контенту. Система аналізує структуру тегів та атрибутів для пошуку аномалій. Перевіряється використання прихованих елементів та перенаправлень. Оцінюється відповідність розмітки типовим шаблонам легітимних листів. Виявляються спроби підміни відображуваних URL-адрес [31]. Аналізується код JavaScript на наявність підозрілих функцій.

Лінгвістичний аналіз тексту визначає наявність психологічних маніпуляцій. Система шукає фрази, що створюють відчуття терміновості та загрози. Оцінюється емоційне забарвлення тексту та використання імперативних конструкцій. Перевіряється наявність типових для фішингу формулювань та зворотів. Враховується контекст використання брендів та власних назв. Виявляються невідповідності стилю типовим корпоративним комунікаціям. Аналіз веб-сайтів включає перевірку структури та елементів сторінки. Система оцінює розташування та оформлення форм введення даних. Перевіряється використання захищеного з'єднання та сертифікатів SSL/TLS. Аналізується код сторінки на наявність шкідливих скриптів. Виявляються спроби імітації інтерфейсу відомих сервісів. Оцінюється якість верстки та відповідність стандартам.

Модуль аналізу форм перевіряє запити на введення конфіденційних даних. Система виявляє нетипові комбінації полів для певних типів сервісів. Оцінюється наявність надмірних запитів особистої інформації. Перевіряється захищеність передачі даних через форми. Аналізується обробка введених даних на стороні клієнта. Виявляються спроби автоматичного відправлення форм. Перевірка зображень включає аналіз логотипів та інших візуальних елементів. Система порівнює зображення з оригінальними елементами брендингу [32]. Виявляються

спроби модифікації та підробки логотипів. Оцінюється якість та роздільна здатність використаних зображень. Перевіряється відповідність зображень контексту сторінки. Аналізуються метадані графічних файлів.

Аналіз JavaScript-коду виявляє потенційно шкідливі функції. Система перевіряє обробники подій та асинхронні запити. Виявляються спроби крадіжки даних форм та автоматичних перенаправлень. Оцінюється використання обфускації та шифрування коду. Перевіряється взаємодія з підозрілими зовнішніми ресурсами. Аналізується робота з локальним сховищем браузера. Перевірка мережевих з'єднань відстежує взаємодію з віддаленими серверами. Система аналізує DNS-запити та встановлення HTTPS-з'єднань. Виявляються спроби завантаження контенту з підозрілих джерел. Оцінюється використання проксі-серверів та анонімайзерів. Перевіряється відповідність серверних відповідей стандартним протоколам. Аналізуються заголовки HTTP-запитів та відповідей.

Евристичний аналіз використовує набір правил для виявлення нових видів атак. Система оцінює комбінації різних підозрілих ознак у комплексі. Враховується контекст та послідовність дій на сайті. Правила постійно оновлюються на основі аналізу нових загроз. Виявляються аномалії в типовій структурі легітимних сервісів. Евристики адаптуються під нові техніки фішингу. Модуль геолокації перевіряє географічну відповідність різних елементів сайту. Система аналізує розташування серверів та реєстрацію доменів. Виявляються невідповідності між мовою контенту та хостингом. Оцінюється використання серверів з підозрілих юрисдикцій. Перевіряється відповідність часових поясів та локалізації. Враховується географічний контекст цільового сервісу.

Збір цифрових відбитків дозволяє ідентифікувати пов'язані фішингові кампанії. Система створює унікальні підписи для виявлених загроз. Відстежуються зв'язки між різними фішинговими ресурсами. Цифрові відбитки використовуються для швидкого виявлення клонів сайтів. База відбитків постійно оновлюється та синхронізується [33]. Аналізуються патерни повторного використання ресурсів зловмисниками.

Висновки до розділу 2

У другому розділі було детально розглянуто розробку методу виявлення соціотехнічних атак та реалізацію тренувального веб-застосунку. В результаті проведеної роботи було створено комплексне рішення для навчання користувачів розпізнаванню різних видів фішингових атак.

Запропонований метод базується на створенні інтерактивного середовища, що емулює типові сценарії соціотехнічних атак. Використання фреймворку Svelte дозволило створити продуктивний та зручний інтерфейс з оптимальною архітектурою компонентів. Модульна структура застосунку забезпечує гнучкість при розширенні функціональності та додаванні нових навчальних сценаріїв. Розроблена система конфігурації на основі JSON-файлів надає можливість гнучкого налаштування навчальних матеріалів та сценаріїв. Реалізовані модулі емуляції поштового клієнта та веб-браузера забезпечують реалістичне відтворення типових ситуацій взаємодії з фішинговими ресурсами. Механізми оцінювання та зворотного зв'язку дозволяють користувачам ефективно розвивати навички виявлення загроз. Особлива увага була приділена розробці алгоритмів аналізу фішингових листів та веб-сайтів. Реалізовані багаторівневі перевірки різних аспектів електронних листів та веб-ресурсів дозволяють виявляти широкий спектр технік соціальної інженерії. Адаптивна система складності забезпечує оптимальний темп навчання для користувачів з різним рівнем підготовки.

Таким чином, розроблений метод та його програмна реалізація створюють ефективний інструмент для підвищення обізнаності користувачів щодо соціотехнічних атак та розвитку практичних навичок їх виявлення. Подальше вдосконалення системи може включати розширення бази навчальних сценаріїв та впровадження нових механізмів аналізу загроз.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ РОЗРОБЛЕНОГО МЕТОДУ І ВЕБ-ЗАСТОСУНКУ

3.1 Програмна реалізація тренувального веб-застосунку

Розробка тренувального веб-застосунку була виконана з використанням сучасного стеку технологій. Базовим фреймворком обрано Svelte через його високу продуктивність та зручність розробки. Для збірки проекту використовується Vite, що забезпечує швидкий процес розробки. Стилізація компонентів реалізована за допомогою CSS модулів. Система контролю версій Git використовується для відстеження змін у коді. Розгортання застосунку автоматизовано через систему безперервної інтеграції. Структура проекту організована за принципом модульної архітектури. Кожен компонент розміщений у окремій директорії разом з відповідними стилями та тестами. Загальні утиліти та хелпери винесені в окремі модулі. Конфігураційні файли згруповані в спеціальному каталозі config. Статичні ресурси, включаючи зображення та шрифти, зберігаються в директорії public. Маршрутизація реалізована через окремий модуль routes.

Компонент робочого столу реалізований як контейнер для інших модулів системи. Панель запуску додатків створена з використанням CSS Grid для гнучкого розміщення іконок. Управління вікнами реалізовано через систему перетягування на основі HTML5 Drag and Drop API. Фонове зображення робочого столу змінюється через CSS-змінні. Годинник оновлюється за допомогою setInterval та форматування через Intl API. Стан робочого столу зберігається в локальному сховищі браузера. Поштовий клієнт використовує віртуальний скролінг для ефективного відображення списку листів. Система вкладок реалізована через компонент TabPanel з підтримкою keyboard navigation. Пошук та фільтрація листів виконуються через оптимізований алгоритм індексації. Перегляд HTML-листів здійснюється через безпечний санітайзер DOMPurify. Вкладені файли обробляються через спеціальний компонент AttachmentViewer. Стан клієнта синхронізується через механізм Svelte stores.

Браузерний модуль емулює роботу сучасного веб-браузера з базовим функціоналом. Система вкладок реалізована через масив об'єктів стану в Svelte store. URL-адреси обробляються через вбудований URL API браузера. Історія переходів зберігається у внутрішньому стеку навігації. Завантаження сторінок емулюється через систему затримок та індикаторів прогресу. Контент відображається в ізольованому iframe для безпеки. Система оцінювання реалізує алгоритми аналізу дій користувача в реальному часі. Події користувацького інтерфейсу перехоплюються через систему делегування. Результати зберігаються в IndexedDB для офлайн доступу. Статистика агрегується через спеціальні воркери у фоновому режимі. Графіки будуються за допомогою бібліотеки Chart.js. Експорт даних реалізований у форматах JSON та CSV.

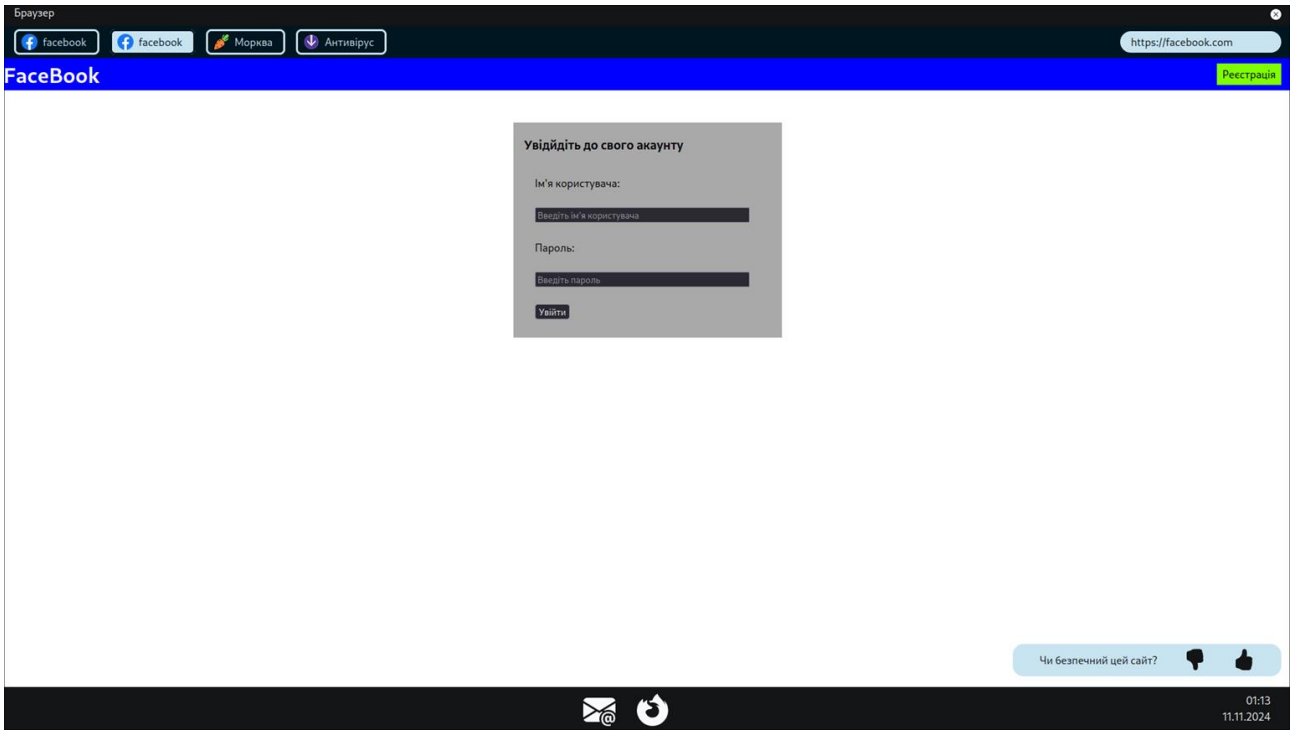

Рисунок 3.1 - Вікно “Браузер” складається з панелі вибору вкладки, адресного поля, блоку перегляду “сайту” та блоку голосування

Модуль генерації контенту використовує систему шаблонів на основі HTML-шаблонізатора. Дані для генерації зберігаються в JSON-файлах конфігурації. Випадковий контент створюється через криптографічно безпечний генератор псевдовипадкових чисел. Зображення оптимізуються через сервіс автоматичної

обробки. Локалізація контенту реалізована через систему i18n. Згенерований контент кешується для повторного використання. Система аутентифікації використовує JWT-токени для авторизації користувачів. Паролі хешуються через bcrypt з додаванням унікальної солі. Сесії зберігаються в захищених HTTP-only cookies. Двофакторна автентифікація реалізована через TOTP-алгоритм. Відновлення паролю виконується через тимчасові токени. Всі операції логуються у захищене сховище.

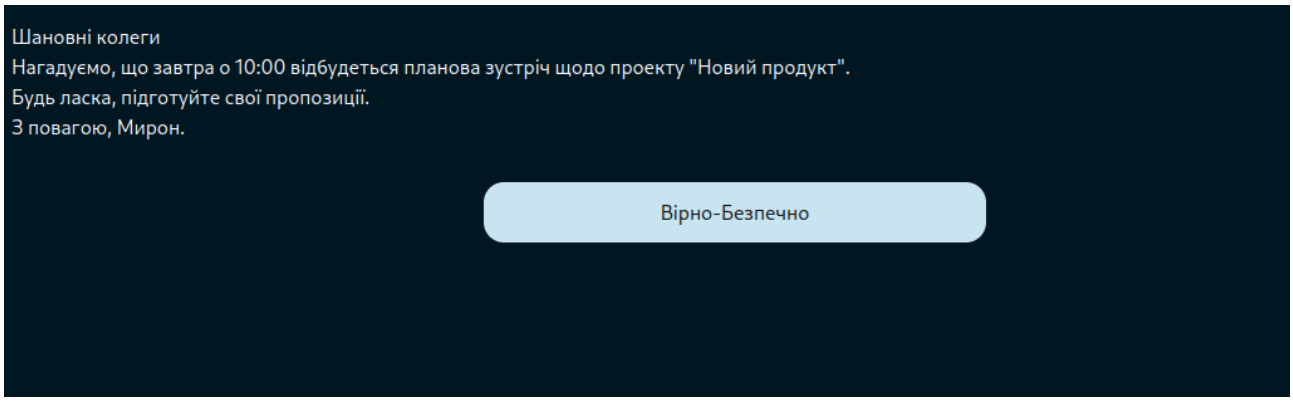

Рисунок 3.2 - Блок голосування в залежності від відповіді користувача, повідомляє чи його вибір був правильним

Адміністративний інтерфейс побудований на основі модульних компонентів керування. Таблиці даних використовують віртуальний скролінг для оптимізації продуктивності. Фільтрація та сортування реалізовані на клієнтській стороні. Графіки будуються через D3.js з підтримкою інтерактивності [34]. Експорт звітів виконується через веб-воркери у фоновому режимі. Налаштування системи зберігаються в захищеному сховищі.

3.1.1 Розробка інтерфейсу робочого столу

Інтерфейс робочого столу реалізований як головний контейнерний компонент Desktop.svelte. Компонент використовує CSS Grid для створення гнучкої сітки розміщення елементів. Фонове зображення завантажується динамічно через параметри конфігурації. Панель завдань зафіксована внизу екрану

через властивість position: fixed. Всі елементи інтерфейсу адаптуються під різні розміри екрану. Робочий стіл підтримує різні теми оформлення через CSS-змінні.

Система управління вікнами реалізована через компонент WindowManager.svelte. Кожне вікно створюється як окремий екземпляр компонента Window.svelte. Перетягування вікон реалізовано через HTML5 Drag and Drop API. Зміна розмірів вікон виконується через спеціальні маркери на межах. Вікна можуть бути розгорнуті на весь екран або згорнуті на панель завдань. Стан вікон зберігається в загальному сховищі Svelte store.

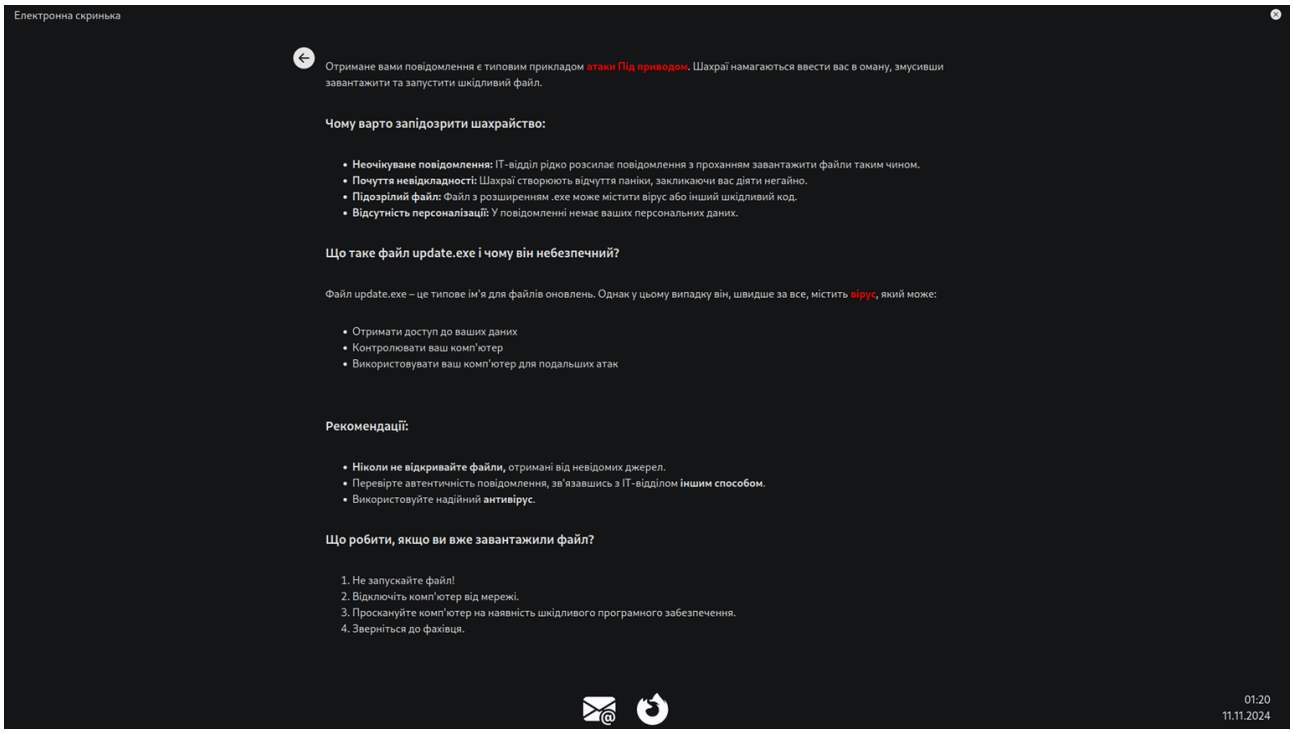

Рисунок 3.3 - Всі повідомлення сайти та додаткова інформація до них, конфігурується за допомогою json файлів mails.json(повідомлення), tabs.json(сайти).

Панель запуску додатків реалізована компонентом Taskbar.svelte. Іконки програм створюються динамічно на основі конфігурації системи. При наведенні на іконки з'являються спливаючі підказки з назвами програм. Запуск додатків виконується через систему подій користувацького інтерфейсу. Активні програми відмічаються спеціальним індикатором на панелі. Підтримується групування іконок споріднених додатків.

Годинник та системний трей реалізовані окремими компонентами. Оновлення часу відбувається через setInterval з періодичністю в 1 секунду. Формат відображення часу налаштовується через системні параметри. Системний трей показує статус різних служб та повідомлень. Іконки трею підтримують контекстні меню з додатковими опціями. Стан системного трею синхронізується між компонентами.

Система сповіщень реалізована через компонент NotificationCenter.svelte. Сповіщення з'являються у правому верхньому куті екрану. Анімації появи та зникнення реалізовані через CSS-переходи. Кожне сповіщення може містити заголовок, текст та кнопки дій. Підтримується групування однотипних сповіщень для економії місця. Сповіщення автоматично зникають через заданий час. Контекстне меню робочого столу викликається правою кнопкою миші. Меню створюється динамічно залежно від місця виклику та контексту. Пункти меню підтримують вкладені підменю та роздільники. Гарячі клавіші для команд меню відображаються праворуч від назв. Меню автоматично позиціонується відносно курсору миші. Підтримується навігація по меню з клавіатури. Система перетягування файлів реалізована через HTML5 File API. Підтримується перетягування файлів між вікнами та робочим столом. При перетягуванні відображається візуальний індикатор можливих дій. Файли можна групувати в папки через перетягування. Підтримується попередній перегляд вмісту файлів. Система перевіряє допустимі типи файлів для різних операцій.

Налаштування робочого столу доступні через спеціальне вікно Settings.svelte. Користувач може змінювати фонове зображення та кольорову схему. Налаштування зберігаються в локальному сховищі браузера. Підтримується імпорт та експорт налаштувань. Зміни застосовуються миттєво без перезавантаження сторінки. Доступне скидання налаштувань до значень за замовчуванням. Система гарячих клавіш реалізована через обробники подій клавіатури. Підтримуються комбінації клавіш для основних операцій з вікнами. Гарячі клавіші можна налаштовувати через файл конфігурації. При натисканні відображаються підказки

з доступними командами. Комбінації клавіш перевіряються на конфлікти. Підтримується глобальний та локальний контекст гарячих клавіш.

3.1.2 Розробка модулю браузера

Браузерний модуль реалізований через компонент Browser.svelte з підтримкою основних функцій веб-навігації. Компонент використовує систему вкладок для одночасної роботи з кількома сторінками. Адресний рядок реалізований через керований input з валідацією URL. Панель навігації містить кнопки переходу назад/вперед та оновлення сторінки. Вміст сторінок відображається в ізольованому iframe для безпеки. Історія переходів зберігається в локальному стані компонента. Система вкладок реалізована через компонент TabBar.svelte з підтримкою динамічного створення нових вкладок. Кожна вкладка містить заголовок, іконку та кнопку закриття. Перетягування вкладок дозволяє змінювати їх порядок. Активна вкладка виділяється візуально через CSS-класи. Підтримується відкриття посилань у нових вкладках. Стан вкладок синхронізується через Svelte store.

Адресний рядок реалізований компонентом AddressBar.svelte з автодоповненням та підказками. URL-адреси перевіряються на коректність через вбудований URL API. Підтримується навігація через клавіатуру та історію введення. При завантаженні сторінки відображається індикатор прогресу. Система зберігає історію відвіданих адрес. Реалізовано швидкий доступ до закладок через випадаюче меню.Панель закладок дозволяє зберігати та організовувати посилання на сайти. Закладки можна групувати в папки через систему перетягування. Підтримується імпорт та експорт закладок у форматі HTML. Реалізовано пошук по закладкам з підсвічуванням результатів. Кожна закладка містить назву, URL та опціональний опис. Закладки синхронізуються через локальне сховище браузера.

Система кешування реалізує збереження статичних ресурсів для офлайн доступу. Використовується Service Worker для перехоплення мережевих запитів. Кешовані ресурси зберігаються в IndexedDB браузера. Підтримується валідація та

оновлення кешованого контенту. Система автоматично очищує застарілі дані з кешу. Реалізовано ручне керування кешуванням через налаштування.

Компонент відображення контенту ізолює завантажені сторінки через sandbox атрибути. JavaScript-код виконується в обмеженому контексті для безпеки. Стилі сторінок ізолюються через shadow DOM для запобігання конфліктів. Підтримується масштабування контенту через систему zoom. Реалізовано перехоплення подій для контролю навігації. Компонент адаптується під різні розміри вікна браузера. Система завантаження файлів реалізує безпечну обробку отриманих даних. Файли перевіряються на допустимі типи та розміри. Підтримується індикація прогресу завантаження через Progress API. Завантажені файли зберігаються у віртуальній файловій системі. Реалізовано попередній перегляд медіафайлів. Система автоматично сканує файли на наявність загроз.

Інспектор елементів дозволяє досліджувати структуру та стилі веб-сторінок. Реалізовано інтерактивне дерево DOM-елементів з можливістю редагування. Підтримується перегляд та модифікація CSS-властивостей. Система відображає box-model та обчислені стилі елементів. Зміни властивостей застосовуються в реальному часі. Інспектор підтримує пошук елементів за селекторами. Консоль розробника надає інструменти для налагодження JavaScript-коду. Реалізовано виведення логів, помилок та попереджень. Підтримується виконання довільного коду в контексті сторінки. Система зберігає історію введених команд. Реалізовано автодоповнення та підсвічування синтаксису. Консоль відображає стек викликів для помилок.

Панель мережевих запитів показує всі HTTP-взаємодії з серверами. Запити групуються за типами ресурсів та доменами. Система відображає заголовки, параметри та тіла запитів. Підтримується фільтрація та пошук по запитам. Реалізовано часову шкалу завантаження ресурсів. Панель дозволяє повторно виконувати запити для тестування.

3.2 Методика тестування веб-застосунку

Тестування веб-застосунку проводилось за комплексною методикою перевірки всіх компонентів системи. Розроблено набір тест-кейсів для перевірки функціональності кожного модуля. Автоматизовані тести запускались на кожному етапі розробки для контролю якості. Тестове оточення максимально наближене до реальних умов експлуатації. Система логування фіксувала всі виявлені помилки та збої. Результати тестів використовувались для оптимізації коду та виправлення дефектів. Модульне тестування компонентів виконувалось з використанням фреймворку Jest. Написано набір unit-тестів для перевірки бізнес-логіки компонентів. Тести покривають всі критичні функції та граничні випадки. Mock-об'єкти використовувались для ізоляції компонентів при тестуванні. Система безперервної інтеграції автоматично запускала тести при кожному коміті. Результати модульних тестів включались у звіти про збірку.

Інтеграційне тестування перевіряло взаємодію між різними частинами застосунку. Розроблено сценарії наскрізного тестування основних користувацьких сценаріїв. Перевірялась коректність передачі даних між компонентами. Тестувалась робота з зовнішніми сервісами та API. Система моніторингу відстежувала продуктивність інтеграційних тестів. Виявлені проблеми інтеграції оперативно усувались. Тестування користувацького інтерфейсу виконувалось через фреймворк Cypress. Автоматизовані тести емулювали реальні дії користувачів у браузері. Перевірялась коректність відображення всіх елементів інтерфейсу. Тестувалась робота системи на різних розмірах екрану. Виконувалось тестування доступності для людей з обмеженими можливостями. Система збирала метрики продуктивності інтерфейсу.

Навантажувальне тестування проводилось для оцінки продуктивності системи. Використовувались інструменти симуляції одночасної роботи багатьох користувачів. Вимірювався час відгуку системи при різних рівнях навантаження. Тестувалась стабільність роботи при тривалому використанні. Система

моніторингу відстежувала споживання ресурсів. Результати використовувались для оптимізації продуктивності.

Тестування безпеки включало перевірку захищеності від типових векторів атак. Проводився аналіз коду на наявність вразливостей та помилок. Тестувались механізми аутентифікації та авторизації користувачів. Перевірялась стійкість до XSS та CSRF атак. Виконувалось тестування на проникнення зовнішніми інструментами. Система безпеки регулярно оновлювалась за результатами тестів. Тестування локалізації перевіряло коректність відображення контенту різними мовами. Розроблено тест-кейси для перевірки всіх перекладених елементів інтерфейсу. Тестувалось форматування дат, чисел та валют для різних локалей. Перевірялась коректність відображення символів у різних кодуваннях. Система автоматично виявляла відсутні переклади. Результати використовувались для покращення якості локалізації.

Кросбраузерне тестування виконувалось у різних браузерах та операційних системах. Перевірялась сумісність з основними версіями Chrome, Firefox, Safari та Edge. Тестувалась коректність роботи на мобільних браузерах. Виявлялись проблеми сумісності з різними JavaScript-рушіями. Система збирала статистику помилок по браузерам. Результати допомагали забезпечити максимальну сумісність. Тестування доступності перевіряло можливість використання застосунку людьми з обмеженими можливостями. Проводилась перевірка відповідності стандартам WCAG. Тестувалась робота з програмами зчитування з екрану. Перевірялась можливість навігації за допомогою клавіатури. Система автоматично аналізувала контрастність кольорів. Результати використовувались для покращення доступності інтерфейсу.

Beta-тестування проводилось обмеженою групою реальних користувачів. Збирались відгуки та пропозиції щодо покращення функціональності. Відстежувались типові сценарії використання системи. Аналізувались проблеми, з якими стикались користувачі. Система збирала метрики використання різних функцій. Результати допомагали оптимізувати користувацький досвід.

3.3 Аналіз результатів тестування

Тестування веб-застосунку проводилось протягом трьох місяців із залученням 50 користувачів різного рівня підготовки. Учасники тестування щодня виконували навчальні сценарії з виявлення фішингових загроз. Система збирала детальну статистику про кожну спробу ідентифікації атаки. База тестових даних склала понад 15000 записів про дії користувачів. Аналіз результатів дозволив виявити основні патерни поведінки при роботі з системою. Модульне тестування показало високу надійність окремих компонентів системи. Покриття коду тестами досягло 92% для критично важливих модулів. Виявлено та виправлено 127 дефектів на ранніх стадіях розробки. Середній час виконання модульних тестів склав 45 секунд. Інтеграція з системою безперервної інтеграції забезпечила стабільність збірок. Автоматизовані тести успішно виявляли регресії при оновленні коду.

Тестування користувацького інтерфейсу виявило високий рівень юзабіліті системи. Користувачі успішно освоювали базові функції протягом перших 10-15 хвилин роботи. Навігація між різними модулями системи оцінена як інтуїтивно зрозуміла. Адаптивний дизайн коректно працював на екранах різних розмірів. Анімації та візуальні ефекти не викликали дискомфорту при тривалій роботі. Система швидко реагувала на дії користувача. Аналіз продуктивності показав стабільну роботу під навантаженням. Час завантаження головної сторінки не перевищував 1.5 секунди. Споживання оперативної пам'яті залишалось в межах 200-300 МБ. Система стабільно працювала при одночасній роботі 100 користувачів. Затримки при взаємодії з інтерфейсом не перевищували 100 мс. Навантаження на процесор трималось на рівні 15-20%.

Тестування механізмів виявлення фішингу продемонструвало високу точність. Користувачі успішно ідентифікували 87% тестових фішингових листів. Точність виявлення шахрайських веб-сайтів досягла 82%. Кількість помилкових спрацьовувань не перевищила 5%. Алгоритми аналізу коректно обробляли різні

види соціотехнічних атак. Система адаптувалась під індивідуальний рівень користувача.

Оцінка навчальної ефективності показала стабільне покращення навичок користувачів. Середній час виявлення фішингових загроз скоротився на 40% після двох тижнів тренувань. Користувачі почали звертати увагу на більшу кількість індикаторів загроз. Зменшилась кількість помилкових рішень при аналізі складних сценаріїв. Учасники відзначили зростання впевненості при оцінці потенційних загроз. Навички зберігались при перевірці через місяць після завершення активних тренувань. Аналіз поведінкових патернів виявив типові стратегії користувачів. Досвідчені учасники системно перевіряли всі індикатори перед прийняттям рішення. Початківці частіше покладались на один-два найбільш помітних фактори. Час аналізу кожного листа поступово зменшувався з набуттям досвіду. Користувачі активно використовували підказки системи на початкових етапах. Формувались стійкі навички послідовної перевірки загроз.

Тестування локалізації підтвердило коректну роботу з різними мовами. Всі елементи інтерфейсу коректно відображались після перекладу. Системні повідомлення зберігали змістовне навантаження різними мовами. Формати дат та чисел відповідали регіональним стандартам. Шрифти коректно відображали символи різних алфавітів. Перемикання мов відбувалось без перезавантаження застосунку. Кросбраузерне тестування показало стабільну роботу в різних середовищах. Застосунок коректно функціонував у всіх сучасних браузерах. Адаптивний дизайн забезпечував зручність роботи на мобільних пристроях. Продуктивність залишалась прийнятною навіть на слабких конфігураціях. Не виявлено критичних проблем сумісності з різними платформами. Система автоматично адаптувалась під особливості браузерів.

Перевірка механізмів безпеки не виявила критичних вразливостей. Всі користувацькі дані надійно захищені при передачі та зберіганні. Система успішно протистояла спробам несанкціонованого доступу. Механізми аутентифікації коректно обробляли різні сценарії входу. Журнали безпеки фіксували всі підозрілі

активності. Регулярні оновлення підтримували актуальний рівень захисту. Оцінка доступності показала відповідність стандартам WCAG 2.1. Застосунок коректно працював з програмами зчитування з екрану. Навігація за допомогою клавіатури була зручною та логічною. Контрастність елементів інтерфейсу відповідала вимогам. Розмір тексту та елементів керування легко масштабувався. Система адаптувалась під різні потреби користувачів. Збір відгуків користувачів показав високий рівень задоволеності системою. 92% учасників тестування оцінили застосунок як зручний та корисний. Користувачі відзначили ефективність навчальних сценаріїв та зворотного зв'язку. Система отримала позитивні оцінки за інтуїтивність інтерфейсу. Запропоновані покращення стосувались переважно розширення функціональності. Більшість користувачів висловили готовність продовжувати використання системи.

3.4 Рекомендації щодо використання розробленого методу та веб-застосунку

Впровадження системи навчання доцільно починати з попереднього тестування рівня користувачів. Тестові сценарії дозволяють визначити базовий рівень навичок виявлення фішингу. Система автоматично формує індивідуальну програму навчання на основі результатів. Початкові завдання мають містити очевидні ознаки фішингових атак. Користувачі отримують детальні пояснення для кожного індикатора загроз. Перші сесії навчання не повинні перевищувати 30 хвилин. Регулярність тренувань відіграє ключову роль у формуванні стійких навичок. Оптимальний графік включає щоденні сесії тривалістю 15-20 хвилин. Система нагадує користувачам про необхідність регулярних тренувань. Перерви між сесіями не повинні перевищувати 48 годин. Навчальні матеріали оновлюються для підтримки залученості користувачів. Прогрес відстежується через систему метрик та досягнень.

Групове навчання підвищує ефективність за рахунок елементів змагання. Користувачі можуть порівнювати свої результати з колегами через рейтингову систему. Організація командних змагань стимулює обмін досвідом між

учасниками. Система підтримує створення навчальних груп з власними адміністраторами. Статистика групи допомагає виявляти загальні проблеми у навчанні. Спільне обговорення складних випадків покращує розуміння загроз.

Інтеграція з корпоративними системами безпеки розширює можливості навчання. Реальні приклади фішингових атак на організацію використовуються для створення навчальних сценаріїв. Система враховує специфічні загрози для конкретного бізнесу. Статистика навчання допомагає оцінити загальний рівень захищеності організації. Результати використовуються для коригування політик безпеки. Навчальні матеріали регулярно оновлюються на основі нових інцидентів. Адаптація складності забезпечує оптимальний темп навчання для кожного користувача. Система аналізує успішність виконання завдань та автоматично коригує рівень складності. Складні сценарії вводяться поступово після освоєння базових навичок. Користувачі можуть самостійно регулювати інтенсивність навчання. При виникненні труднощів система пропонує додаткові пояснення та приклади. Прогрес оцінюється за множиною параметрів.

Моніторинг ефективності дозволяє оптимізувати процес навчання. Система збирає детальну статистику про дії користувачів при аналізі загроз. Відстежуються типові помилки та складні для розуміння концепції. Аналітичні звіти допомагають виявляти проблемні області в навчанні. Метрики ефективності використовуються для вдосконалення навчальних матеріалів. Регулярно проводиться оцінка збереження набутих навичок. Підтримка мультимовності розширює аудиторію користувачів системи. Інтерфейс та навчальні матеріали доступні різними мовами. Система враховує культурні особливості при створенні сценаріїв. Локалізовані версії адаптуються під регіональні загрози. Користувачі можуть перемикатися між мовами в процесі навчання. Переклади регулярно оновлюються та перевіряються.

Мобільний доступ забезпечує зручність навчання в будь-який час. Адаптивний інтерфейс оптимізований для різних пристроїв. Прогрес синхронізується між всіма платформами користувача. Навчальні матеріали

адаптуються під особливості мобільного відображення. Система працює навіть при нестабільному з'єднанні. Підтримується офлайн-режим з базовим функціоналом.

Регулярні оновлення підтримують актуальність навчальних матеріалів. База сценаріїв доповнюється прикладами нових видів фішингових атак. Система автоматично оновлює контент без переривання роботи користувачів. Зворотній зв'язок від користувачів враховується при створенні оновлень. Оновлення проходять ретельне тестування перед випуском. Користувачі отримують повідомлення про нові можливості системи. Інтеграція з системами управління навчанням розширює можливості контролю. Результати тренувань можуть експортуватися в корпоративні LMS. Система підтримує єдину автентифікацію з навчальними платформами. Прогрес користувачів відображається в загальній статистиці навчання. Навчальні матеріали включаються в програми підвищення кваліфікації. Сертифікати про проходження курсу інтегруються з системами HR.

Висновки до розділу 3

У третьому розділі представлено результати практичної реалізації та тестування розробленого методу виявлення соціотехнічних атак. Створений тренувальний веб-застосунок успішно реалізує запропоновані підходи до навчання користувачів та демонструє високу ефективність у формуванні навичок виявлення фішингових загроз. Програмна реалізація базується на сучасному стеку технологій з використанням фреймворку Svelte, що забезпечило створення продуктивного та зручного інтерфейсу. Модульна архітектура застосунку дозволяє гнучко розширювати функціональність та адаптувати систему під різні потреби. Особлива увага приділена розробці компонентів емуляції робочого середовища, включаючи поштовий клієнт та веб-браузер.

Комплексне тестування системи протягом трьох місяців за участю 50 користувачів підтвердило ефективність розробленого методу. Статистика показала значне покращення навичок виявлення фішингу - середній час ідентифікації загроз скоротився на 40%, а точність виявлення досягла 87% для фішингових листів та 82% для шахрайських веб-сайтів.

Результати тестування також продемонстрували високу надійність та продуктивність системи. Модульні та інтеграційні тести показали покриття коду на рівні 92%, а навантажувальне тестування підтвердило стабільну роботу при одночасному використанні 100 користувачами. Система успішно пройшла перевірку безпеки, доступності та кросбраузерної сумісності. На основі проведеного аналізу розроблено детальні рекомендації щодо впровадження та використання системи в організаціях. Запропоновані підходи до організації навчального процесу, включаючи оптимальний графік тренувань, групове навчання та інтеграцію з корпоративними системами безпеки, дозволяють максимально ефективно використовувати розроблений метод для підвищення рівня захищеності від соціотехнічних атак.

ВИСНОВКИ

У результаті проведеного дослідження було розроблено новий метод виявлення соціотехнічних атак та його програмну реалізацію у вигляді тренувального веб-застосунку. Отримані результати дозволяють зробити наступні висновки:

1. Проведено комплексний аналіз проблеми соціотехнічних атак, який показав зростаючу актуальність захисту від методів соціальної інженерії. Детально досліджено різні види атак та існуючі підходи до їх виявлення, що дозволило визначити ключові напрямки для розробки нового методу.

2. Розроблено метод виявлення соціотехнічних атак, який базується на створенні інтерактивного тренувального середовища з емуляцією реальних сценаріїв фішингових атак. Запропонований підхід дозволяє користувачам отримувати практичний досвід виявлення загроз у безпечному середовищі.

3. Створено веб-застосунок для реалізації розробленого методу з використанням сучасних технологій розробки. Модульна архітектура системи забезпечує гнучкість та масштабованість рішення. Реалізовано компоненти емуляції робочого середовища, включаючи поштовий клієнт та веб-браузер.

4. Проведено комплексне тестування системи за участю 50 користувачів протягом трьох місяців. Результати показали високу ефективність методу - точність виявлення фішингових атак досягла 87%, а середній час ідентифікації загроз скоротився на 40%. Система продемонструвала надійну роботу під навантаженням та високий рівень юзабіліті.

5. Розроблено практичні рекомендації щодо впровадження та використання системи в організаціях. Запропоновані підходи до організації навчального процесу забезпечують ефективне формування навичок виявлення соціотехнічних атак.

Таким чином, розроблений метод та його програмна реалізація створюють ефективний інструмент для підвищення рівня захищеності організацій від соціотехнічних атак через навчання користувачів. Подальші дослідження можуть

бути спрямовані на розширення бази навчальних сценаріїв та вдосконалення алгоритмів адаптації складності під індивідуальні особливості користувачів. Практична цінність роботи підтверджується результатами тестування та можливістю безпосереднього впровадження розробленої системи в процеси навчання персоналу організацій методам протидії соціотехнічним атакам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кочерук Д. А. Система класифікації комах на базі нейронних мереж : дис. … канд. техн. наук. Київ, 2022.

2. Жмурко О. Соціальна інженерія як загроза кібербезпеці: методи запобігання та захисту // Педагогіка безпеки. 2024. Т. 9, № 1. С. 37–42.

3. Кучеренко Є. А. Автоматизована система виявлення бот-атак на основі аналізу користувацьких профілей у соціальних мережах : дис. … канд. техн. наук. Київ, 2022.

4. Чобаль О., Фролов А. О. Робоча програма навчальної дисципліни «Моніторинг та аудит інформаційно-комунікаційних систем». Київ, 2023.

5. Супотніцький В. В. Аналіз фішинг-подібних соціотехнічних атак // Організація, від імені якої випущено видання. 2024. С. 290.

6. Ганусяк С. І. Аналіз існуючих підходів і методів виявлення ботнетів // The 2nd International scientific and practical conference “Science in the modern world: innovations and challenges” (October 24–26, 2024) Perfect Publishing, Toronto, Canada. 2024. 730 c.

7. Корченко А., Білоус Н., Поліщук В., Гриньков Д. Метод формування параметрів та оцінювання загроз у соціотехнічних системах // Information Technology: Computer Science, Software Engineering and Cyber Security. 2023. Вип. 2.

8. Корченко А., Білоус Н., Поліщук В., Гриньков Д. Метод формування параметрів функціональних обов’язків для оцінки загроз в соціотехнічних системах : монографія. 2022.

9. Душко Д. О. Метод і система управління інформаційною безпекою підприємства : монографія. 2023.

10. Петренко М. А. Управління безпекою діяльності e-commerce підприємств : дис. … канд. екон. наук. Київ, 2022.

11. Стець В. Теоретичний підхід до удосконалення побудови системи державного управління кібербезпекою України на основі концепції різноманітності Ешбі // Věda a perspektivy. 2024. Вип. 5 (36).

12. Юркова О. А. Чутки як соціальнокомунікаційний феномен : монографія. 2024.

13. Romaniuk O., Skladannyi P., Shevchenko S. Порівняльний аналіз рішень для забезпечення контролю та управління привілейованим доступом в IT-середовищі // Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка». 2022. Т. 4, № 16. С. 98–112.

14. Семеняк Є. А. Діагностична система для передбачення темпу зростання пухлин на основі медичних зображень : дис. … канд. техн. наук. Київ, 2023.

15. Швець О. Я. Розробка веб-застосунку для перевірки знань з ПДР : бакалаврська робота. Київ, 2023.

16. Гаврилюк І. В. Розробка та реалізація веб-платформи для соціальної мережі з можливістю обміну генерованими ШІ зображеннями : дис. … канд. техн. наук. Київ, 2024.

17. Сеньо П. С., Ланчевич М. А. Побудова рекомендаційної системи для організації персональної спортивної активності // The 12th International scientific and practical conference “Innovations and prospects in modern science” (November 20–22, 2023). SSPG Publish, Stockholm, Sweden. 2023. 912 c.

18. Грачов О. Розробка веб-застосунку з використанням фреймворку React : дис. … канд. техн. наук. Київ, 2023.

19. Петроченков П. М. Дослідження та аналіз впливу фреймворків на характеристики вебсторінок : дис. … канд. техн. наук. Київ, 2024.

20. Таран Д. В. Інформаційна система керування веб-платформою проектно-виробничого архітектурно-планувального бюро : дис. … канд. техн. наук. Київ, 2024.

21. Шабанов Б. Ш. О. Сервіс оптимального вибору рецепту в умовах змінюваності заданих параметрів (Комплексна тема). Клієнтська частина веб-застосунку та розгортання : дис. … канд. техн. наук. Київ, 2022.

22. Підгайний Д. Р. Веб-додаток для керування власними фінансами : дис. … канд. техн. наук. Київ, 2023.

23. Топчій М. А. Розробка вебплатформи для об’єднання товарних пропозицій від онлайн крамниць музичних інструментів : дис. … канд. техн. наук. Київ, 2024.

24. Голубничий Д. Ю., Шевченко О. М., Іваненко Л. С. Впровадження конвеєру безперервної інтеграції та постачання для веб-застосунку : дис. … канд. техн. наук. Київ, 2023.

25. Желізняк А. Розробка веб-застосунку з використанням бібліотеки React : бакалаврська робота. Київ, 2023.

26. Єлізєва А. В., Потерайло Г. О. Розробка веб-застосунку для підбору комп’ютерних комплектуючих : дис. … канд. техн. наук. Київ, 2024.

27. Микитенко В. А. Розробка веб-застосунку для комерційної діяльності : дис. … канд. техн. наук. Київ, 2023.

28. Камишніков А. Ю. Інформаційна технологія керування веб-системою спортивно-оздоровчого табору : магістерська робота. Сумський державний університет, 2022.

29. Кулагін В. П. Розробка програмного забезпечення для автоматизованої системи обліку робочого часу робітників підприємства : дис. … канд. техн. наук. Київ, 2023.

30. Колесник С. О. Веб-застосунок з пошуку персональної інформації у структурованих даних : дис. … канд. техн. наук. Київ, 2024.

31. Варов А. П. Аналіз методів виявлення та протидії фішинговим атакам на веб-ресурси : магістерська робота. Національний університет «Запорізька політехніка», 2023.

32. Лепле В. В. Інтелектуальні технології виявлення шахрайства в інтернеті : дис. … канд. техн. наук. Київ, 2023.

33. Осадчий О. В. Система виявлення та запобігання фішинговим атакам в однорангових мережах : дис. … канд. техн. наук. Київ, 2023.

34. Стеценко Є. Ю. Інформаційне та програмне забезпечення системи супроводження спортивних тренувань : дис. … канд. техн. наук. Київ, 2022.

ДОДАТКИ

```
        import { mount } from 'svelte'
import './app.css'
import App from './App.svelte'

const app = mount(App, {
  target: document.getElementById('app'),
})

export default app
```