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

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«_____»_____ 2023 р.

На правах рукопису
УДК 004.056.53

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: Система двофакторної автентифікації для вебзастосунків

Виконавець: Микола СОКОЛОВСЬКИЙ
Керівник: к.т.н., доцент Микола БРАЇЛОВСЬКИЙ
Консультант розділу «Охорона навколишнього середовища»: к.т.н., доцент Тетяна ДМИТРУХА
Нормоконтролер: к.т.н., доцент Микола БРАЇЛОВСЬКИЙ

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Магістр

Спеціальність: 125 «Кібербезпека»

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

ЗАТВЕРДЖУЮ
Завідувач кафедри Комп'ютеризованих систем захисту інформації
_____ Михайло СТЕПАНОВ
«__» _____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

здобувача вищої освіти Соколовського Миколи Михайловича

1. Тема: *Система двофакторної аутентифікації для вебзастосунків* затверджена наказом ректора від «15» вересня 2023 р. № 1814/ст.
2. Термін виконання: з 16.10.2023 р. по 31.12.2023 р.
3. Вихідні дані: проаналізувати існуючі системи аутентифікації; на основі аналізу виділити основні параметри, завдяки яким можливо виявити їх переваги та недоліки; розробити програмний модуль подвійної аутентифікації.
4. Зміст пояснювальної записки: аналіз основ безпеки для веб застосунку, їх загроз та методів забезпечення аутентифікації; дослідження роботи сучасних програмних інтерфейсів та впровадження найкращих практик для забезпечення більшої безпеки; розробка системи аутентифікації та інтеграція програмного модулю та демонстрація його роботи;

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

№ з/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	16.10.2023	*Виконано*
2.	Аналіз літературних джерел	17.10.2023	*Виконано*
3.	Обґрунтування вибору рішення	18.10.2023	*Виконано*
4.	Збір інформації	19.10.2023 - 26.10.2023	*Виконано*
5.	Аналіз механізмів захисту вебдодатків	27.10.2023 - 02.11.2023	*Виконано*
6.	Створення системи двофакторої автентифікації	03.11.2023 - 20.11.2023	*Виконано*
7.	Реалізація програмного модулю подвійної автентифікації	21.11.2023 - 05.12.2023	*Виконано*
8.	Проведення тестування системи	06.12.2023 - 08.12.2023	*Виконано*
9.	Перевірка на антиплагіат	11.12.2023	*Виконано*
10.	Оформлення і друк пояснювальної записки	12.12.2023 - 14.12.2023	*Виконано*
11.	Оформлення презентації	15.12.2023 - 17.12.2023	*Виконано*
12.	Отримання рецензій від рецензента	22.12.2023	*Виконано*

Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона навколишнього середовища	Дмитруха Т.І.		

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

Здобувач вищої освіти Микола СОКОЛОВСЬКИЙ
(підпис, дата)

Керівник кваліфікаційної роботи Микола БРАІЛОВСЬКИЙ
(підпис, дата)

РЕФЕРАТ

Дипломна робота складається зі вступу, 4 розділів, висновків, списку використаної літератури, додатку і має 80 сторінок основного тексту, 14 рисунків, 2 таблиці, 1 сторінку додатків. Список використаних джерел містить 25 найменувань і займає 3 сторінки. Загальний обсяг роботи 85 сторінок.

Мета роботи: реалізація системи двофакторної автентифікації на основі стандарту FIDO2

Було вирішено такі задачі як:

- Аналіз методів безпеки для вебзастосунку, загроз та способів забезпечення автентифікації;
- Дослідження роботи сучасних програмних інтерфейсів, та впровадження найкращих практик для забезпечення більшої безпеки;
- Створення системи автентифікації, реалізація та інтеграція програмного модулю автентифікації та демонстрація його роботи;

В роботі розроблена система двофакторної аутентифікації, що базується на програмному інтерфейсі WebAuthn та використовує найкращі практики для забезпечення безпеки облікових записів користувачів. Розроблений програмний модуль може бути використаний у компаніях для забезпечення зручної системи реєстрації та аутентифікації у вебзастосунках.

Ключові слова: вебзастосунок, двофакторна аутентифікація, U2F, FIDO2, WebAuthn.

ЗМІСТ

ВСТУП 6

РОЗДІЛ 1. АНАЛІЗ МЕХАНІЗМІВ ЗАХИСТУ ВЕБДОДАТКІВ 8

1.1 Забезпечення безпеки за допомогою автентифікації 8

1.2 Сучасні методи автентифікації 12

1.3 Аналіз основних загроз, вразливостей та атак на методи автентифікації . 22

1.4 Висновки до першого розділу 26

РОЗДІЛ 2. СТВОРЕННЯ СИСТЕМИ ДВОФАКТОРНОЇ АВТЕНТИФІКАЦІЇ . 27

2.1 Аналіз сучасних технологій автентифікації 27

2.2 Проектування архітектури 36

2.3 Принцип роботи протоколу 45

2.4 Висновки до другого розділу 52

РОЗДІЛ 3. ПРОГРАМНИЙ МОДУЛЬ ПОДВІЙНОЇ АУТЕНТИФІКАЦІЇ ДЛЯ ВЕБЗАСТОСУНКІВ 54

3.1 Огляд використаних технологій 54

3.2 Реалізація фреймворку для програмного модулю 59

3.3 Інтеграція систем 68

3.4 Демонстрація роботи програмного модулю 71

3.5 Висновок до третього розділу 76

РОЗДІЛ 4. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА 77

4.1 Екологічні проблеми утилізації побутових відходів. 77

ВИСНОВКИ 81

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 82

Додаток А 85

ВСТУП

Актуальність: Наразі переважаючий метод автентифікації користувачів, який використовується більшістю компаній, покладається виключно на паролі. Тим не менш, у міру того, як обчислювальні можливості продовжують розвиватися, уразливість системи безпеки на основі паролів зростає, що робить злом пароля все більш досяжним. Традиційний спосіб вирішення цієї проблеми полягає в підвищенні складності паролів та інтеграції вторинного фактора автентифікації.

Двофакторна автентифікація є наріжним каменем у рамках моделі безпеки нульової довіри. Щоб захистити конфіденційні дані, система повинна встановити ідентифікацію осіб, які намагаються отримати до них доступ. Двофакторна автентифікація пропонує потужний захист від безлічі загроз безпеці, спрямованих на паролі та облікові записи користувачів, включаючи фішингові атаки, атаки грубої сили, атаки типу "людина посередині" тощо.

Однак користувачі часто відчувають побоювання щодо складних систем входу, що змушує їх шукати обхідні шляхи. Прийняття двофакторної автентифікації сприяє цій складності, а різні методи автентифікації містять в собі вразливі місця. У результаті навіть найпростіша та найпоширеніша форма двофакторної автентифікації, наприклад перевірка на основі SMS, може бути сприйнятливою до використання зловмисниками.

Зважаючи на такі проблеми, ця дипломна робота намагається запропонувати інноваційну та комплексну двофакторну систему автентифікації. Ця система використовує встановлені найсучасніші технології для безперебійного, швидкого та високобезпечного доступу до вебдодатків.

Мета роботи: реалізація системи двофакторної автентифікації на основі стандарту FIDO2

Відповідно були поставленні такі задачі як:

- Аналіз методів безпеки для вебзастосунку, їх загроз та способів забезпечення автентифікації;
- Дослідження роботи сучасних програмних інтерфейсів та впровадження найкращих практик для забезпечення більшої безпеки;
- Створення системи, реалізація програмного модулю для двофакторної автентифікації та демонстрація його роботи;

Об’єкт дослідження: процес автентифікації для вебдодатків.

Предмет дослідження: система двофакторної автентифікації для вебдодатків.

Новизна: вдосконалено систему автентифікації шляхом покращення алгоритму взаємодії з програмним інтерфейсом Webauthn та інтеграцією розробленого програмного модулю з системами OpenWRT та OpenNDS.

Практична цінність полягає у тому, що розроблена система надає можливість впровадження сильної автентифікації не тільки для звичайних користувачів, а й для вбудованих систем таких як: банкомати, пристрої з технологіями «Інтернет речей», плати Raspberry Pi та інші пристрої, які можна використовувати в місцях підвищеного ризику, щоб захистити облікові записи користувачів вебзатосунку.

Галузь застосування. Запропонована система автентифікації може використовуватись у будь-якій сфері, де потребується поліпшення безпеки та зручності автентифікації для вебзастосунків.

Апробація. Основні тези даної роботи були розглянуті та опубліковані на наступній конференції:

VII Міжнародна науково-практична конференція «Прикладні системи та технології в інформаційному суспільстві» Київський національний університет імені Тараса Шевченка, 2023.

РОЗДІЛ 1. АНАЛІЗ МЕХАНІЗМІВ ЗАХИСТУ ВЕБДОДАТКІВ

1.1 Забезпечення безпеки за допомогою автентифікації

Більшість онлайн-платформ вимагають від своїх користувачів реєстрації, щоб надати повний доступ до свого функціоналу. Ці ресурси використовують загальні методи захисту облікових записів, що зазвичай включають створення паролів різного рівня складності. Складність паролів визначається довжиною та різноманітністю символів, які в них використовуються. Рекомендується використовувати різні паролі для різних сервісів та періодично змінювати їх для забезпечення високого рівня безпеки облікового запису. Однак, навіть найміцніший пароль може бути недостатньо ефективним у боротьбі з різними видами атак.

Незаконний доступ до облікових записів є однією з найсерйозніших та поширених загроз у цифровому світі, і може мати серйозні наслідки, такі як неочікувана установка шкідливого програмного забезпечення, порушення конфіденційності даних або витік паролів. За даними Symantec, щомісяця більше ніж 4800 веб-сайтів стають жертвами компромісу через атаки, які використовують злоякісний код, впроваджений через HTML-форми. Найпоширенішими причинами порушення конфіденційності даних є компроміс автентифікації та порушення контролю доступу. Запобігання цим загрозам вимагає впровадження надійних засобів захисту даних та міцних механізмів контролю доступу, таких як ідентифікація, автентифікація та авторизація, для забезпечення високого рівня безпеки.

Контроль доступу - це ключовий елемент, який забезпечує можливість доступу до ресурсів лише для ідентифікованих, автентифікованих та авторизованих користувачів. Незважаючи на те, що ці три аспекти тісно пов'язані між собою, важливо розрізняти їх для ефективного застосування.

Початковим етапом цього процесу є ідентифікація, коли користувач розкриває свою ідентичність. Цей етап базується на різних ідентифікаторах, таких як імена користувачів, ідентифікатори процесу, поштові скриньки, смарт-карти або будь-який унікальний засіб, що може ідентифікувати суб'єкта або особу. Системи безпеки використовують цей процес ідентифікації для визначення того, чи має особа право доступу до захищеної інформації.

Методи ідентифікації охоплюють широкий спектр, включаючи інструменти, такі як системи спостереження, відбитки пальців та зразки ДНК, які служать для остаточного встановлення ідентичності особи. В цифровому світі особу також можна ідентифікувати за їхнім стилем письма, клавішними натисками та іншими характерними атрибутами, що додатково сприяють процесу ідентифікації.

Ідентифікація особистості, також відома як перевірка особистості, включає завдання пов’язати людину з певним користувачем. Ця процедура має велике значення, оскільки вона стосується ключових питань щодо автентичності особи, таких як «Чи заявлена особа є точною?», «Чи ця особа раніше мала іншу особу?» або «Чи слід цій особі надати доступ до нашої системи?». Ідентифікація пропонує кілька переваг для організацій, зокрема:

- Повна інтеграція з різними системами.
- Вимоги до економного обслуговування.
- Діє як потужний стримуючий фактор проти потенційних нападників.

Автентифікація — це процедура підтвердження особи, яка зазвичай відбувається, коли особи надають необхідні облікові дані. Наприклад, коли користувач вводить дійсний пароль разом з іменем користувача, пароль служить підтвердженням того, що користувач справді володіє вказаним іменем користувача. По суті, автентифікація служить для підтвердження заявленої особистості.[2]

У системі, захищеній автентифікацією за іменем користувача та паролем, користувачі повинні надати дійсні облікові дані для доступу до системи. Це має подвійну мету: захистити систему від несанкціонованого вторгнення третьої сторони та захист конфіденційності користувача з потенційними правовими наслідками у разі порушення.

Щоб підтвердити ідентифікацію користувача, облікові дані надсилаються в систему контролю доступу, яка згодом оцінює їх автентичність перед наданням автентифікації. По суті, цей процес автентифікації, який виконується з певним ступенем впевненості, засвідчує, що користувач володіє наданими обліковими даними та контролює їх.

Облікові дані автентифікації можна розділити на чотири окремі групи, які зазвичай називають методами або типами автентифікації. Вони охоплюють:

- На основі знань: ця категорія включає щось, що користувач знає, наприклад пароль, персональний ідентифікаційний номер (PIN) або парольну фразу. Автентифікація людей на основі їхніх знань є простою, але загалом менш безпечною.
- На основі володіння: до цієї групи входять предмети, якими фізично володіє користувач, наприклад смарт-карта, USB-ключ, карта пам’яті або маркер. Він часто використовується для доступу до таких місць, як банки та офіси, а також для перевірки облікових даних системи. Незважаючи на те, що ці предмети відносно прості, їх можна вкрасти.
- На основі біометрії: ця категорія спирається на те, ким є користувач за своєю суттю, використовуючи біометричні дані, як-от відбитки пальців, топологію долоні, геометрію руки, сканування райдужної оболонки/сітківки або розпізнавання обличчя. Використання біометрії забезпечує найвищий рівень точності та безпеки під час ідентифікації особи.
- На основі поведінки: ця група включає динамічні біометричні дані, такі як шаблони набору тексту, шаблони підписів або зразки голосу, для

автентифікації осіб на основі їхніх унікальних поведінкових характеристик.

Автентифікація є важливим інструментом для забезпечення безпеки організаційних мереж. Ця процедура дозволяє організаціям контролювати доступ до своїх захищених ресурсів, таких як комп’ютерні системи, мережі, бази даних, вебсайти та інші мережеві програми чи сервіси, надаючи можливість взаємодії лише автентифікованим користувачам.

Авторизація є важливим компонентом безпеки в інформаційних системах, призначеним для визначення та контролю прав доступу користувачів до різних ресурсів системи. Ця процедура визначає, які дії та операції можуть виконувати користувачі після того, як вони успішно автентифікувалися (підтвердили свою ідентичність) як відповідні особи.

Усвідомлення того, що авторизація неможлива без попередньої ідентифікації та автентифікації, вкрай важливе. Ідентифікація дозволяє системі визначити, хто саме намагається отримати доступ, і, зазвичай, використовує інформацію про користувача, таку як ім'я, логін або ідентифікатор. Автентифікація, з іншого боку, перевіряє, чи ця особа дійсно та валідно представляє себе, надаючи коректні облікові дані та відповідаючи запитам безпеки, таким як пароль або біометричні дані.

Після успішної ідентифікації та автентифікації система може визначити рівень доступу або ролі, які надаються користувачеві. Це означає, що користувач може мати обмежений доступ до певних ресурсів або функціональностей системи відповідно до своєї ролі або рівня привілеїв. Ця стратегія дозволяє забезпечити прозорий та ефективний контроль доступу, забезпечуючи конфіденційність, цілісність та доступність ресурсів.

Без процесу авторизації система стає вразливою перед можливими загрозами, оскільки будь-який користувач може мати необмежений доступ до всіх ресурсів, що може призвести до неповного захисту інформації та

можливих порушень безпеки. Тому важливо ретельно налаштовувати правила авторизації для кожного користувача або групи користувачів, щоб забезпечити безпеку системи та обмежити доступ до секретних або конфіденційних даних.

1.2 Сучасні методи автентифікації

Зазвичай, в автентифікаційній системі задіяні різні сутності. Загальна схема включає актора та залежну сторону (RP, або "Relying Party"): перший бажає підтвердити свою ідентичність для створення автентифікованої сесії з другою сутністю, яка може виконувати запити актора лише у випадку автентифікації.

Спочатку актор та Постачальник кредитних послуг (CSP, або "Credential Service Provider") виконують протокол реєстрації; на цій фазі актора називають заявником. Після успішного завершення цієї фази, CSP зберігає атрибути актора, необхідні для перевірки його ідентичності, і пов'язує їх з ним. Іноді заявнику надається аутентифікатор, такий як сертифікат X.509, який служить офіційним доказом ідентичності. У цьому випадку наявність надійного CSP пов'язує аутентифікатор із конкретним користувачем. Після отримання облікових даних актор може запустити протокол аутентифікації з валідатором: ця сутність зв'язується з CSP і запитує атрибути користувача; на цій фазі актора називають претендентом. Якщо фазу аутентифікації успішно завершено, претендент стає підписником, автентифікованою сутністю, і валідатор обмінюється з RP підтвердженнями аутентифікації, які забезпечують, що підписник має певні характеристики.

1.2.1 Автентифікація за SMS

Процес автентифікації через SMS зазвичай починається з успішного входу користувача в програму за допомогою свого імені користувача та пароля. Після цього сервіс формує та відправляє на мобільний телефон користувача SMS-повідомлення, що містить одноразовий пароль. Коли користувач вводить одноразовий пароль, отриманий у повідомленні, служба порівнює його з надісланим паролем. Якщо цей другий крок виконано успішно, користувач проходить автентифікацію та отримує доступ до програми.

Однак цей метод створює значні проблеми з надійністю, оскільки навіть у випадку підключення за контрактом користувачі не мають права власності на телефонні номери, пов’язані з їхніми SIM-картами. Хоча користувачі можуть фізично обмежити доступ до свого мобільного пристрою, сама SIM-карта за своєю суттю не прив’язана до певного номера. Оператори мобільного зв’язку можуть у будь-який час змінити номер на іншу SIM-карту, створюючи потенційну загрозу безпеці.

Подібний процес автентифікації також можна виконати через електронну пошту. Підхід цілком аналогічний, за винятком того, що одноразовий пароль передається на адресу електронної пошти користувача, а не на його мобільний телефон.

1.2.2 Біометрична автентифікація

Біометрична автентифікація охоплює низку біологічних атрибутів, які служать для унікальної ідентифікації людей. Ці атрибути можна розділити на дві окремі групи:

Статична біометрична автентифікація – це метод безпеки, який використовує фізичні чи поведінкові особливості людини для перевірки її особи. На відміну від динамічної біометричної автентифікації, яка передбачає

безперервний моніторинг біометричних характеристик особи (наприклад, моніторинг натискань клавіш або ходи людини), статична біометрична автентифікація базується на одноразовому зборі біометричних даних для підтвердження особи. Ці дані зазвичай зберігаються в базі даних і порівнюються з біометричними даними, наданими під час спроби автентифікації.

Ось кілька поширених прикладів методів статичної біометричної автентифікації:

- Розпізнавання відбитків пальців: цей метод ґрунтується на захопленні та порівнянні зображень, що зображують унікальні папілярні візерунки на пальцях людини. Спеціальні сканери виділяють кінцеві точки та точки розгалужень, перетворюючи їх у цифровий код для порівняння з попередньо збереженими даними для підтвердження ідентичності.
- Розпізнавання геометрії руки: ця техніка передбачає розпізнавання геометричних характеристик руки людини, як правило, включаючи такі вимірювання, як ширина долоні та радіус кола, яке може помістится на долоні. Для більш точних результатів також можна використовувати атрибути пальців, що дозволяє більш детально аналізувати зображення.
- Сканування сітківки: використовуючи інфрачервоне світло, сканери сітківки фіксують дані про кровоносні судини в задній частині ока. Цей метод забезпечує одну з найнадійніших систем автентифікації з ймовірністю помилкової ідентифікації, що не перевищує однієї десятої відсотка.
- Розпізнавання райдужної оболонки: райдужна оболонка кожної людини має унікальну текстуру, яку можна використовувати для ідентифікації. Подібно до інших біометричних методів, інфрачервоні сканери використовуються для отримання зображення райдужної оболонки, яке потім перетворюється на спеціальний код для порівняння.

- Розпізнавання обличчя: цей підхід використовує відеокамери, фотоапарати або спеціалізовані пристрої, здатні знімати 2D або 3D зображення обличчя. Після отримання зображення виділяються ключові характеристики обличчя, такі як ніс, губи, брови, контур очей, і обчислюється відстань між ними. Потім ці дані використовуються для створення цифрового представлення обличчя з метою автентифікації.
- Автентифікація ДНК: Хоча автентифікація на основі ДНК є найскладнішим і найточнішим методом ідентифікації осіб, автентифікація на основі ДНК ще не отримала широкого поширення через відсутність зручних і широкодоступних систем розпізнавання ДНК.

Таким чином, біометрична автентифікація охоплює різні біологічні атрибути для унікальної ідентифікації користувача, починаючи від розпізнавання відбитків пальців і геометрії руки до методів на основі сітківки, райдужної оболонки ока, обличчя та ДНК, кожен з яких має свої переваги та проблеми.[3]

Статична біометрична автентифікація має кілька переваг:

Висока точність. Статичні біометричні методи зазвичай вважаються високоточними, оскільки вони спираються на унікальні фізичні чи поведінкові риси, які важко підробити чи відтворити.

Зручність: користувачам не потрібно запам’ятовувати паролі чи носити фізичні маркери, як смарт-картки, що робить автентифікацію зручнішою.

Ненав’язливість: багато статичних біометричних методів, як-от розпізнавання відбитків пальців або обличчя, є ненав’язливими та не потребують фізичного контакту з пристроєм.

Однак існують також деякі проблеми та міркування, пов’язані зі статичною біометричною автентифікацією:

Занепокоєння щодо конфіденційності. Зберігання біометричних даних викликає занепокоєння щодо конфіденційності, оскільки вони є конфіденційними та можуть бути використані неналежним чином, якщо їх не захистити належним чином.

Безпека. Біометричні дані можуть бути вразливими до крадіжки або підробки, хоча сучасні системи часто включають заходи проти підробки.

Змінність точності: на точність статичних біометричних методів можуть впливати такі фактори, як умови освітлення, старіння та зміни фізичних характеристик людини.

Вартість: Впровадження біометричних систем автентифікації може бути дорогим через потребу в спеціалізованому апаратному та програмному забезпеченні.

Незважаючи на ці труднощі, статична біометрична автентифікація широко використовується в різних програмах, включаючи розблокування смартфонів, системи контролю доступу та перевірку особи для фінансових транзакцій, завдяки сильній безпеці та перевагам зручності. Щоб забезпечити конфіденційність і безпеку біометричних даних, організації, які впроваджують ці системи, повинні дотримуватися найкращих практик і відповідних норм.

Динамічна біометрична автентифікація, також відома як біометрія поведінки або постійна автентифікація, є методом забезпечення безпеки, який ґрунтується на моніторингу та аналізі поведінкових паттернів та дій особи в реальному часі для підтвердження її ідентичності. На відміну від статичної біометричної автентифікації, яка використовує одноразове збереження біометричних даних для перевірки особи, динамічна біометрична автентифікація безперервно оцінює та автентифікує користувача на основі їхньої поточної діяльності та поведінкових характеристик. Цей підхід додає додатковий рівень безпеки, постійно моніторинг та адаптація до змін у поведінці особи.

Ось деякі загальні приклади методів динамічної біометричної автентифікації:

- Динаміка натискання клавіш: Динаміка натискання клавіш полягає в аналізі унікального способу, яким особа набирає текст на клавіатурі або сенсорному екрані. Фактори, такі як швидкість набору, ритм, тривалість натискання клавіш та сила натискання, можуть бути використані для створення профілю поведінки для перевірки ідентичності.
- Рухи та взаємодія мишею: Цей метод включає в себе аналіз способу руху та взаємодії особи з комп'ютерною мишею або тачпадом. До цього враховуються фактори, такі як швидкість руху, прискорення та патерни кліків мишею.
- Визначення ходи: Визначення ходи оцінює унікальні патерни ходи та рухів особи. Це може бути використано в системах безпеки, які моніторять відеопотік для ідентифікації людей за їхньою походкою.
- Перевірка підпису: Динамічна перевірка підпису порівнює поточний підпис особи, включаючи тиск, порядок виконання та швидкість, зі збереженим профілем підпису.
- Біометрия дотику та жестів: Ці методи аналізують спосіб взаємодії особи з сенсорними пристроями, такими як сенсорні екрани, враховуючи фактори, такі як сила натискання пальця, патерни свайпів та розпізнавання жестів.
- Голосовий та мовний аналіз: Динамічне голосове визначення безперервно моніторить голос та мовні патерни особи під час розмови для перевірки її ідентичності.
- Біометрика поведінки: Біометрика поведінки може охоплювати широкий спектр дій та поведінки, включаючи спосіб утримання пристрою, навігацію по веб-сайту або використання мобільного додатку. Алгоритми машинного навчання аналізують ці дії для створення унікального профілю для кожного користувача.

Переваги динамічної біометричної автентифікації включають:

Постійний моніторинг: Динамічні біометричні методи надають постійну автентифікацію, зменшуючи ризик несанкціонованого доступу під час сесії.

Адаптивність: Ці методи можуть адаптуватися до змін у поведінці користувача з часом, таким як старіння або зміни фізичних характеристик.

Підвищена безпека: Постійний моніторинг поведінки ускладнює завдання зловмисникам, що намагаються видаїть себе за авторизованих користувачів.

Мала інтрузивність: Динамічні біометричні методи часто є незавадливими та не потребують додаткового обладнання, такого як сканери відбитків пальців або камери для сканування радужок.

Проте динамічна біометрична автентифікація також стикається з деякими викликами:

Помилкові позитиви та негативи: Постійний моніторинг може час від часу призводити до помилкових позитивів (неправильна автентифікація несанкціонованого користувача) або помилкових негативів (невдала автентифікація авторизованого користувача).

Питання конфіденційності: Постійний моніторинг поведінки може викликати питання конфіденційності, і користувачі можуть почувати себе незручно з постійним аналізом своєї діяльності.

Складність: Впровадження систем динамічної біометричної автентифікації може бути складним, вимагаючи вдосконалених алгоритмів машинного навчання та аналізу даних.

Динамічна біометрична автентифікація використовується в різних застосунках, включаючи онлайн-банкінг, кібербезпеку та системи контролю доступу, де постійний моніторинг та адаптивна безпека є важливими для захисту важливих даних і систем. Організації, які впроваджують динамічну біометрику, повинні враховувати питання конфіденційності та забезпечувати

безпеку обробки даних користувачів відповідно до відповідних правил та регулювань.

Вся ця інформація повинна збиратися та зберігатися під час реєстрації користувача. Автентифікацію можна виконати шляхом порівняння представлених біометричних даних з попередньо збереженою інформацією для цього користувача. Стверджується, що всі біометричні системи вимагають апаратного забезпечення для захоплення інформації користувача, але можуть забезпечити дуже високий рівень гарантії ідентичності, оскільки механізм автентифікації заснований на унікальній інформації користувача, яку важко дублювати або імітувати.

1.2.3 Багатофакторна автентифікація

Багатофакторна автентифікація підвищує безпеку завдяки резервуванню, що вимагає кількох ідентифікаційних елементів або кроків для автентифікації. Усі компоненти процесу ідентифікації мають бути успішно завершені; інакше автентифікація не вдається. Основна перевага цього методу полягає в зміцненні безпеки організації, оскільки він вимагає від користувачів підтверджувати свою особу за допомогою не лише імені користувача та пароля. Хоча ідентифікатори користувачів і паролі мають вирішальне значення, вони залишаються вразливими до атак грубої сили та крадіжки зловмисниками. Застосування кількох факторів автентифікації, таких як відбиток великого пальця або USB-ключ, вселяє більшу впевненість у стійкості системи проти потенційних загроз.

Поширеним повсякденним прикладом є банкомат, де користувач покладається на свою банківську картку як початковий метод автентифікації та особистий ідентифікаційний номер, відомий лише йому, як другий засіб автентифікації. Без обох цих елементів автентифікації доступ до банкомату обмежено.

Процес двофакторної автентифікації зазвичай поєднує пароль з іншим методом ідентифікації, який може включати PIN-код, одноразовий пароль, надісланий через SMS або електронну пошту, апаратний пристрій або мобільний додаток, сповіщення про схвалення, надіслане на надійний пристрій, або USB ключ. Цей дворівневий підхід покращує безпеку, вимагаючи від користувачів надання кількох дійсних факторів автентифікації для доступу.

1.3.4 OTP та TOTP

Тимчасові одноразові паролі, як зазначено в RFC-6238, працюють за допомогою попередньо визначеного псевдовипадкового секретного початкового коду. Цей початковий код надійно зберігається як на стороні служби автентифікації, так і на пристрої, керованому користувачем, який може мати форму виділеного апаратного маркера, наприклад RSA SecurID, або мобільної програми, як-от Google Authenticator. [4]

Шестизначний тимчасовий одноразовий пароль (TOTP) постійно генерується шляхом застосування цього секретного початкового значення до функції HMAC (код автентифікації на основі хешування повідомлень), використовуючи початкове значення та поточний час як вхідні дані. На практиці TOTP має тривалість тридцять секунд і повторно видається в кінці кожного інтервалу. Потім користувач вводить шестизначний TOTP, який відображається на його пристрої, в інтерфейс служби. Служба самостійно копіює TOTP зі збереженого вихідного коду та порівнює його з отриманим TOTP.

Однією з суттєвих переваг цього методу є те, що він не вимагає підключення до мережі для створення коду. Служба та пристрій користувача можуть незалежно обчислювати коди, використовуючи одне й те саме попередньо визначене секретне початкове значення. Це усуває вразливість

безпеки, пов’язану з підходом до SMS, коли коди можуть бути перехоплені під час передачі.

Однак помітним недоліком є те, що секретні початкові дані повинні зберігатися на стороні служби, щоб увімкнути перевірку коду. Якщо неавторизована сторона отримає доступ до служби, вона потенційно може викрасти секрети всіх зареєстрованих користувачів і створити дійсні коди без обмежень.

Незважаючи на цей недолік, метод TOTP, особливо коли він використовується в поєднанні з мобільним додатком, виділяється як один із найбезпечніших способів реалізації багатофакторної автентифікації без необхідності використання додаткового апаратного забезпечення. [5]

1.3.5 OAuth

OAuth — це стандарт відкритого протоколу, призначений головним чином для обробки делегованої авторизації. По суті, це дозволяє службам надавати доступ до ресурсів без безпосередньої автентифікації користувача. Натомість, якщо інша платформа вже автентифікувала користувача, вона може підтвердити його особу.

OAuth використовує систему на основі токенів для вирішення різноманітних проблем безпеки та зручності використання, які виникають під час роботи зі статичними паролями. Замість того, щоб покладатися на паролі, отримані від третьої сторони, OAuth надає користувачам маркери для автентифікації. Ця конструкція усуває необхідність для перевіряючої сторони (сервісу, який шукає авторизації) бути причетним до облікових даних користувача.

Наприклад, коли ви входите на веб-сайт, служба може запропонувати вам увійти за допомогою свого облікового запису Google. У цьому випадку користувачеві не потрібно передавати конфіденційні дані для входу; замість цього веб-сайт може отримати доступ до необхідної інформації через Google, залежно від угоди про взаємну довіру між веб-сайтом і Google. Важливо зазначити, що в OAuth відсутні власні можливості автентифікації; він суворо зосереджений на авторизації користувача. Отже, протокол не передбачає передачі особистої інформації користувача, за винятком принаймні маркера авторизації та унікального ідентифікатора користувача, які служать для ідентифікації, виключаючи передачу паролів облікових записів.

1.3 Аналіз основних загроз, вразливостей та атак на методи автентифікації

Сьогодні існує кілька більш складних методів атаки, ніж проста груба сила. Ці методи потребують особливої уваги при розробці надійної системи автентифікації. Проблема полягає не лише в самому паролі, але і в засобах збереження та передачі паролю. До найпоширеніших видів таких атак відносяться: атака повторного відтворення, атака посередника та фішинг.

1.3.1 Атака повторного відтворення

Атака відтворення — це зловмисна дія, під час якої неавторизована сторона перехоплює зв’язок між користувачем і довіреною організацією, згодом використовуючи отримані облікові дані для отримання неавторизованого доступу до служби. Ці облікові дані можуть містити або пароль користувача, або хешований пароль, наданий користувачем для

встановлення своєї особи. Отже, при розробці системи автентифікації необхідно врахувати цю вразливість.

У сценарії повторної атаки хакер захоплює передані дані та повторно передає той самий веб-запит на сервер, фактично маскуючись під браузер законного користувача. Отримавши відповідь від сервера, хакер отримує до нього доступ. Основні типи даних, які привабливі для зловмисників у таких ситуаціях, зазвичай охоплюють:

- Ідентифікатор сеансу.
- Облікові дані для входу та хеш пароля.

Уявіть, що ви намагаєтесь увійти на різні онлайн-платформи, такі як обліковий запис у соціальних мережах або інтернет-форум. Коли ви вводите дані для входу, ваш браузер надсилає хеш імені користувача та пароля на відповідний сервер. Якщо зловмиснику вдасться перехопити ваш хеш пароля та ідентифікатор сеансу, він може ініціювати новий сеанс і видати себе за вас, залишаючи сервер не підозрюючи про злом.

Оскільки хакери можуть повторно надсилати повідомлення через мережу, не розшифровуючи їх, вони можуть легко ввести в оману одержувача, змусивши повірити, що повідомлення є законним. Щоб ефективно вирішити цю ситуацію, одним із настійно рекомендованих підходів є використання унікального ідентифікатора для кожної транзакції чи сеансу. У цьому сценарії, якщо хакер спробує відтворити викрадену інформацію, він встановить власне підключення до сервера або іншого клієнта. Потім служба створить свіжий унікальний ідентифікатор для цього сеансу, на який зловмисник відповість вкраденим ідентифікатором у користувача. Коли сервер порівнює ці два ідентифікатори, він виявить невідповідність і, отже, відхилить спробу входу зловмисника. [6]

1.3.2 Атака «Людина посередині»

Атака “Людина посередині” (Man-in-the-Middle або MitM) є різнобічною та хитрою формою кібератаки, яка може приймати різні форми. У такій атаці, як правило, задіяні три ключові сторони: користувач, зловмисник і довірена сторона. Суть атаки "Чоловік посередині" полягає в тому, що зловмисник перехоплює чутливу інформацію, яка передається між користувачем і службою або іншим користувачем.

Одним з поширених способів виконання цієї атаки є прослуховування комунікації між користувачами в системі. Ця підступна дія відбувається тоді, коли кіберзлочинець розташовується в одній мережі з неозброєним користувачем і надійшовши таємно моніторить всі поточні передачі між користувачем і службою.

Основною метою атаки “Людина посередині” є крадіжка особистої інформації, до якої може входити інформація для входу, реквізити рахунків та номери кредитних карт. Це робить осіб, які використовують фінансові додатки, компанії, які покладаються на веб-додатки-як-сервіс, електронні комерційні платформи та інші веб-сайти, які вимагають аутентифікації користувачів, основними цілями.

Для операторів веб-сайтів та організацій, які дбають про безпеку, важливо застосовувати надійні протоколи безпеки обміну даними. Протоколи, такі як Transport Layer Security (TLS) і HyperText Transfer Protocol Secure (HTTPS), відіграють ключову роль у запобіганні атакам із підміною. Вони досягають цього, безпечно шифруючи і автентифікуючи дані під час передачі, тим самим заважаючи спробам перехоплення трафіку на сайті і унеможливлюючи розшифрування чутливої інформації, такої як токени аутентифікації. [7]

1.3.3 Фішингова атака

Фішинг — це мистецтво обману користувачів шляхом приховування шкідливого веб-сайту або спонукання їх розкрити конфіденційну інформацію. Ілюстративний сценарій включає зловмисника, який створює підроблену копію вашого банківського порталу, а потім маскується під вас, маніпулюючи вашими обліковими даними для входу. Озброївшись цією викраденою інформацією, зловмисник може проникнути на ваш банківський рахунок, отримуючи доступ до скарбниці персональних даних.

Як окремі особи, так і підприємства зобов’язані зміцнити свій захист від небезпеки фішингових атак. Для окремих осіб пильність стає першорядною. Ці атаки часто мають ледве помітні ознаки, такі як орфографічні помилки або зміни в доменних іменах. Важливо, щоб користувачі зупинилися та подумали про автентичність небажаних електронних листів.

Підприємства, з іншого боку, можуть прийняти ряд заходів для зменшення ризиків, пов’язаних як з фішинговими, так і з фішинговими атаками:

1. Запровадження двофакторної автентифікації (2FA): 2FA є одним із найнадійніших засобів захисту від фішингу. Він підвищує безпеку, використовуючи два фактори автентифікації: те, що користувач знає (наприклад, пароль і ім’я користувача), і те, що він має (наприклад, смартфон). Навіть якщо облікові дані співробітника зламано, 2FA гарантує, що викрадена інформація не може бути використана для доступу до конфіденційних програм.
2. Застосування суворої політики керування паролями: організації повинні підтримувати суворі протоколи керування паролями. Співробітників слід заохочувати періодично змінювати свої паролі та утримуватися від

використання однакових паролів для кількох програм. Це мінімізує ризик використання скомпрометованих облікових даних на різних платформах.

3. Проводьте освітні кампанії: зменшення загрози фішингових атак – це не лише технічне завдання. Просвітницькі кампанії можуть відіграти важливу роль у просуванні безпечних практик серед працівників. Співробітники повинні бути проінформовані про небезпеку натискання зовнішніх посилань електронної пошти та інші найкращі методи виявлення та уникнення спроб фішингу.[8]

1.4 Висновки до першого розділу

З розвитком технологій та переходом багатьох сервісів в мережу Інтернет, все частіше постає питання інформаційної безпеки. Однією з найстрашніших загроз безпеці є несанкціонований доступ, і саме автентифікація забезпечує захист від нього. Ця система дозволяє лише перевіреним особам мати доступ до конфіденційних даних. Різні методи автентифікації розрізняються за рівнем захищеності від атак, складністю реалізації та зручністю їх використання. Завдяки комбінації цих методів в дво-чи багатофакторній автентифікації ризики зламу значно знижуються.

У першому розділі були розглянуті основні концепції безпеки в інформаційному середовищі та роль автентифікації в цьому контексті. Аналіз різних методів та їх потенційних загроз підкреслює важливість використання сучасних технологій для забезпечення ефективної автентифікації. У результаті був вибраний найкращий метод автентифікації, і враховані можливі вразливості системи.

РОЗДІЛ 2. СТВОРЕННЯ СИСТЕМИ ДВОФАКТОРНОЇ АВТЕНТИФІКАЦІЇ

2.1 Аналіз сучасних технологій автентифікації

2.1.1 Програмний інтерфейс управління обліковими записами

API для керування обліковими даними: розширення можливостей безперервної автентифікації користувачів

API керування обліковими даними — це потужний інструмент, який дозволяє веб-сайтам безпечно керувати обліковими даними користувача, об’єднаними та відкритими ключами та отримувати доступ до них. Ця функція забезпечує зручну роботу користувача, дозволяючи окремим особам входити в систему, не вводячи паролі. Користувачі можуть легко ідентифікувати об’єднаний обліковий запис, який вони використовували для автентифікації, і відновити сеанси без необхідності явного входу, навіть якщо їхні сеанси минули.

Однією з ключових переваг цього API є його здатність інтегруватися з системою паролів агента користувача. Завдяки цьому веб-сайти можуть оптимізувати процеси керування обліковими даними, забезпечуючи послідовний підхід. Крім того, агенти користувачів отримують вигоду від покращеної підтримки керування обліковими даними користувача, особливо при роботі зі складними об’єднаними постачальниками облікових даних і нетрадиційними механізмами входу, які вимагають більше, ніж просто ім’я користувача та пароль.

Щоб вирішити ці проблеми, API пропонує механізми для веб-сайтів для зберігання та отримання різних типів облікових даних, надаючи користувачам гнучкість перегляду федеративного облікового запису, який використовується для входу, або безперешкодного відновлення сеансів, навіть після завершення процесу входу.

У пізнішій версії специфікації стає можливим отримати облікові дані з іншого субдомену. Наприклад, пароль, збережений на login.webapp.com, можна використовувати для автентифікації на www.webapp.com. Для цього пароль потрібно зберегти за допомогою методу CredentialsContainer.store(). Цей підхід іноді називають зіставленням із загальнодоступним списком суфіксів (PSL), хоча специфікація рекомендує використовувати PSL переважно для визначення дійсної області облікових даних; це не є обов'язковим. Відповідно, реалізації API у браузері можуть відрізнятися.

2.1.2 API WebAuthn

WebAuthn API є розширенням попереднього API, що пропонує розширені можливості для веб-автентифікації. У 2019 році Консорціум всесвітньої павутини (W3C) офіційно визнав стандартом WebAuthn, розроблений для підвищення безпеки методів онлайн-автентифікації шляхом зменшення використання паролів, які легко піддаються фішингу.

Цей веб-API дає змогу веб-сайтам застосовувати автентифікацію на основі FIDO у підтримуваних браузерах і платформах. Це досягається завдяки спрощенню зв’язку з різними автентифікаторами FIDO, охоплюючи як старіші, так і новіші рішення, такі як U2F і UAF. Замість того, щоб покладатися на традиційну автентифікацію на основі пароля, WebAuthn використовує потужність криптографії з відкритим ключем для реєстрації та автентифікації користувачів.

Діапазон методів автентифікації, які підтримує WebAuthn, широкий і включає біометричні дані, Bluetooth, NFC і USB-ключі, такі як YubiKey. Розширюючи набір параметрів автентифікації, WebAuthn підвищує безпеку та зменшує сприйнятливість до фішингових атак.

Ядро WebAuthn API складається з двох основних викликів JavaScript: navigator.credentials.create() і navigator.credentials.get(). Метод create служить для реєстрації нового автентифікатора та збереження облікових даних користувача разом із ним на сервері. Цей процес можна використовувати для реєстрації нових користувачів або додавання додаткових методів автентифікації, включаючи нові асиметричні пари ключів.

Коли вже зареєстрований користувач бажає увійти, у гру вступає виклик navigator.credentials.get(). Цей виклик використовує попередньо зареєстровані облікові дані для проведення процесу автентифікації, забезпечуючи безпечний і плавний вхід. Після успішної автентифікації доступ надається за допомогою ще одного виклику navigator.credentials.get().

Підсумовуючи, WebAuthn API є ключовою розробкою, яка дає змогу веб-додаткам покращувати безпеку автентифікації користувачів шляхом розширення спектру доступних методів автентифікації та зменшення залежності від вразливих паролів. Він встановлює безпечний канал зв’язку між сервером і браузером, підвищуючи загальну безпеку в Інтернеті. [12]

Рис. 2.1 – діаграма взаємодії між сервером і браузером протоколу WebAuthn.

Важливо мати на увазі, що технологія WebAuthn є відносно новою, і, як наслідок, вона може бути несумісною з усіма версіями популярних веб-браузерів. Повний список браузерів, які підтримують програмний інтерфейс WebAuthn, можна знайти в таблиці 2.1.

Таблиця 2.1.

Підтримка WebAuthn на різних версіях браузерів

Браузер	**Версія**	**Підтримка Api**	**Відсоток користувачів**
Chrome	67- Поточна версія	Так	24%
	4-66	Ні	0.7%
Firefox	60- Поточна версія	Так	2.8%
	2-59	Ні	0.6%
Edge	18- Поточна версія	Так	4%
	13-17	Ні[1]	0.1%
Opera	54- Поточна версія	Так	1%
	10-53	Ні	0.2%
Safari	10- Поточна версія	Так	2.6%
	3.1-12.1	Ні[2]	0.2%
Chrome для Android	Поточна версія	Так	40.2%
Safari на iOS	14.5- Поточна версія	Так	13%
	13.3-14.4	Часткова підтримка[3 4]	0.95%
	3.2-13.2	Ні[5]	1%

З різних причин кілька інших браузерів були виключені з цього списку через їх застарілість, відсутність підтримки програмного інтерфейсу або обмежену базу користувачів.

Окремі розділи таблиці містять важливі зауваження:

1. Деякі браузери не пропонують підтримку WebAuthn за замовчуванням, але її можна ввімкнути. Щоб він працював, браузеру потрібен префікс «ms». Починаючи з версії 14 Edge, реалізація має префікс і базується на другій версії FIDO.
2. У деяких браузерах підтримку WebAuthn не ввімкнено за замовчуванням, але її можна активувати в меню експериментальних функцій. Зараз ці браузери підтримують пристрої CTAP і CTAP2 HID на базі USB.
3. Часткова підтримка в деяких браузерах означає, що пристрої FIDO2 можуть не працювати, якщо встановлено PIN-код.
4. Зверніть увагу, що експериментальна функція ще не ввімкнена в браузерах на основі WKWebView.
5. Хоча це не є налаштуванням за замовчуванням, підтримку WebAuthn можна також ввімкнути в Safari в меню експериментальних функцій. [13]

2.1.3 Стандарт FIDO2

FIDO2, відкритий стандарт автентифікації, спільно розроблений FIDO Alliance і Web Consortium, був задуманий для вирішення поширених глобальних проблем з паролями та оновлення звичайних методів автентифікації. По суті, FIDO2 можна розглядати як злиття U2F і UAF, де він будується на їхніх спільних функціях і значно покращує їх.

Альянс FIDO було створено з головною метою розробки надійних рішень автентифікації для заміни паролів. Згодом вони створили кілька методів для досягнення цього, включаючи UAF і згаданий раніше протокол U2F. Завдання

масової заміни стандартних паролів було багатогранним і охоплювало такі проблеми, як неоптимальний досвід роботи з користувачем і дорогі впровадження. Місія FIDO Alliance полягає в тому, щоб підтримувати відкриті стандарти, які пропонують підвищену безпеку порівняно з традиційними паролями, є більш зручними для користувача та легшими для розгортання та ефективного керування службами.

Консорціум був створений у 1994 році і мав на меті впорядкувати та структурувати ландшафт Інтернету, що розвивається. У той час існували обмежені стандарти, які регулювали дизайн веб-браузерів. W3C розпочала подорож, щоб створити веб-стандарти, які б забезпечили однакове відображення веб-сайтів у всіх браузерах. Головною метою W3C є розкриття повного потенціалу Всесвітньої павутини шляхом формулювання стійких стандартів, протоколів і вказівок. Яскраві приклади включають HTML, CSS і, зокрема, WebAuthn. Отже, той факт, що W3C є рушійною силою WebAuthn, підкреслює його потенціал як надійне рішення для автентифікації в різних браузерах, що робить його привабливим вибором для розробників з довгостроковими прагненнями. [14]

FIDO2 містить специфікацію автентифікації від World Wide Web Consortium і відповідний протокол CTAP, розроблений FIDO Alliance. WebAuthn, JavaScript API на основі браузера, оптимізує інтеграцію надійних методів автентифікації для веб-сервісів, враховуючи як безпарольну, так і вторинну автентифікацію. Разом із CTAP2, відповідальним за організацію взаємодії із зовнішніми автентифікаторами, вони разом складають FIDO2. WebAuthn і CTAP керують зв’язком між перевіряючою стороною, клієнтом або браузером і автентифікатором, як зазначено в цьому стандарті. Примітно, що одна з відмінностей між U2F і FIDO2 полягає в тому, що в той час як U2F функціонував виключно як метод аутентифікації за другим фактором, FIDO2 розширює його, щоб включити використання зовнішніх ключів як для аутентифікації за першим, так і за другим фактором. По суті, FIDO2 поєднує

функції присутності користувача ключа U2F з такими методами автентифікації, як PIN-коди, відбитки пальців тощо, розвиваючи концепцію. Важливо, що FIDO2 підтримує сумісність як з U2F, так і з UAF.

Необхідно визнати, що FIDO2 підтримує дві моделі довіри: двосторонню та тристоронню. Перший передбачає довіру між користувачем і сервером, тоді як останній представляє третю сторону, яку зазвичай називають службою метаданих, яка перевіряє пари ключів атестації на автентифікаторах. Ця служба метаданих містить інформацію про різні типи автентифікаторів і призначає їм унікальні коди.

Крім того, вкрай важливо визначити компанії, які вже прийняли цей стандарт для впровадження систем на основі FIDO. Кілька постачальників пропонують ключі безпеки FIDO2 у різних форм-факторах, які, як відомо, сумісні з двофакторною автентифікацією. [15]

Таблиця 2.2.

Таблиця постачальників, які сумісні з технологією FIDO2

Постачальники	**Біометрія**	**USB**	**NFC**	**Bluetooth**	**Сертифікація FIPS**
AuthenTrend	+	+	+	+	-
Ensurity	+	+	-	-	-
Excelsecu	+	+	+	+	-
Feitian	+	+	+	+	+
Giesecke Devrient	+	+	+	+	-
GoTrustID	-	+	+	+	-
HID	-	+	+	-	-
IDmelon Technologies	+	+	+	+	-
KONA I	+	-	+	+	-
NeoWave	-	+	+	-	-
Thetis	+	+	+	+	-
Token2 Switzerland	+	+	+	-	-
Yubico	+	+	+	-	+

2.1.4 технології U2F та UAF

Універсальний другий фактор (U2F) представляє стандарт, який поєднує звичайний статичний метод автентифікації, який залежить від імені користувача та пароля, з додатковим рівнем безпеки. Це збільшення зазвичай досягається шляхом використання спеціалізованих пристроїв, які використовують технології USB, NFC або Bluetooth на U2F-сумісних веб-сайтах і в програмах. Ці гаджети були ретельно розроблені, щоб віддати перевагу простоті, доступності та сумісності. Останнім часом було досягнуто помітного прогресу в розробці «ключів», які пропонують безконтактну автентифікацію, аутентифікацію без маркерів і пароля, що обслуговує як персональні комп'ютери, так і мобільні телефони. Мабуть, найвідомішим із цих ключів є YubiKey, створений шанованою компанією Yubico.

U2F виконує подвійну мету: генерацію ключів і перевірку підпису. Цей стандарт дозволяє створити унікальний ключ для кожного сайту, де розгорнуто пристрій. Важливо зазначити, що U2F не займається перевіркою особи користувача, а лише підтверджує присутність користувача. Отже, обов'язок перевірки користувача лежить на звичайній комбінації імені користувача та пароля, яка є важливою частиною процесу автентифікації. На рисунку 2.1 показано, що протокол U2F працює виключно між зовнішнім автентифікатором і пристроєм. Якщо зв'язок на основі сервера залишається обов'язковим, розробникам слід уважно розглянути цей фактор.

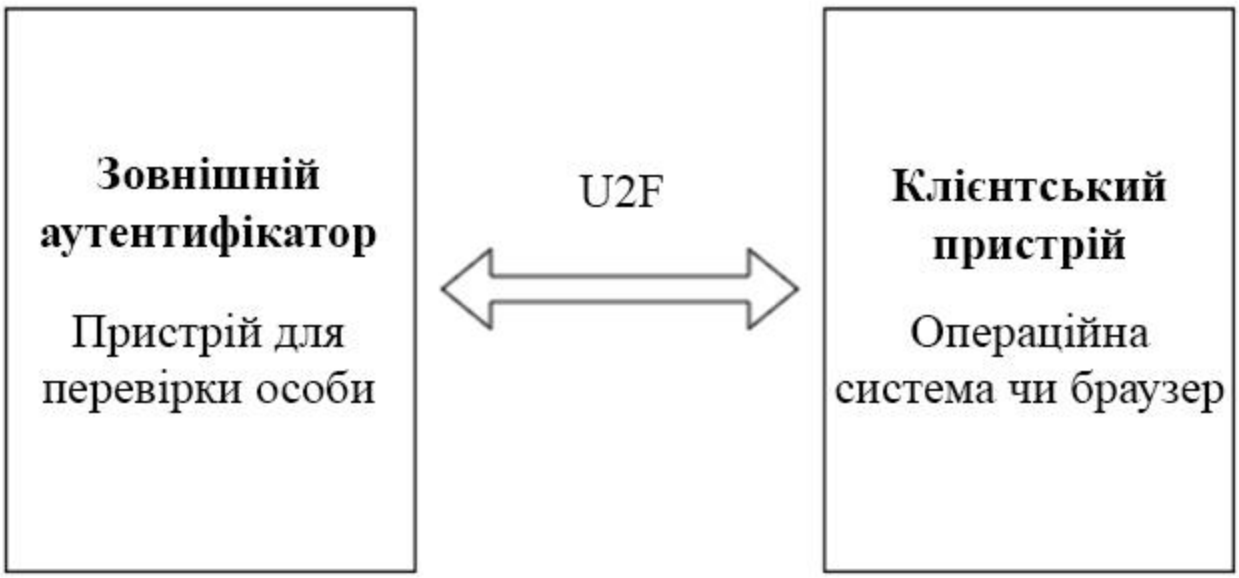

Рис. 2.2 - Схема функціонування U2F

З іншого боку, Universal Authentication Framework (UAF) виступає як стандарт без пароля. Підхід, прийнятий UAF, передбачає, що користувачі реєструють бажаний метод автентифікації в онлайн-сервісах, які пропонують підтримку UAF. UAF плавно використовує властиві можливості використовуваного пристрою, такі як сканування відбитків пальців, PIN-коди, біометрія обличчя тощо. Крім того, ця служба має гнучкість для об’єднання кількох методів автентифікації, створюючи кілька рівнів безпеки. Після того, як користувач зареєстрував ці механізми, він може без особливих зусиль повторити вибраний метод щоразу, коли потрібна автентифікація. Таким чином, користувач звільняється від необхідності вводити ім’я користувача та пароль кожного разу, коли він автентифікується за допомогою зареєстрованого пристрою.

На відміну від стандарту Universal Second Factor, UAF покладається на вбудовані функції пристрою для перевірки ідентичності користувача, тоді як U2F використовує спеціалізовані апаратні ключі, такі як YubiKey, для тієї ж мети. Використовуючи властивість пристрою, його можна застосувати для виклику до перевіряючого можуть здійснювати безпосередньо з веб додатку.

2.2 Проектування архітектури

Для ефективного впровадження універсального методу подвійної аутентифікації слід звернутися до всіх ключових складових системи, які взаємодіють у цьому процесі. Як ілюструє схема, існує ряд елементів, які повинні взаємодіяти та взаємодіяти між собою. Кожен компонент володіє своїми власними конфігураціями та специфікаціями, такими як критерії для методу комунікації. У наступному описі буде висвітлено, які аспекти важливо враховувати під час взаємодії з кожним з них.

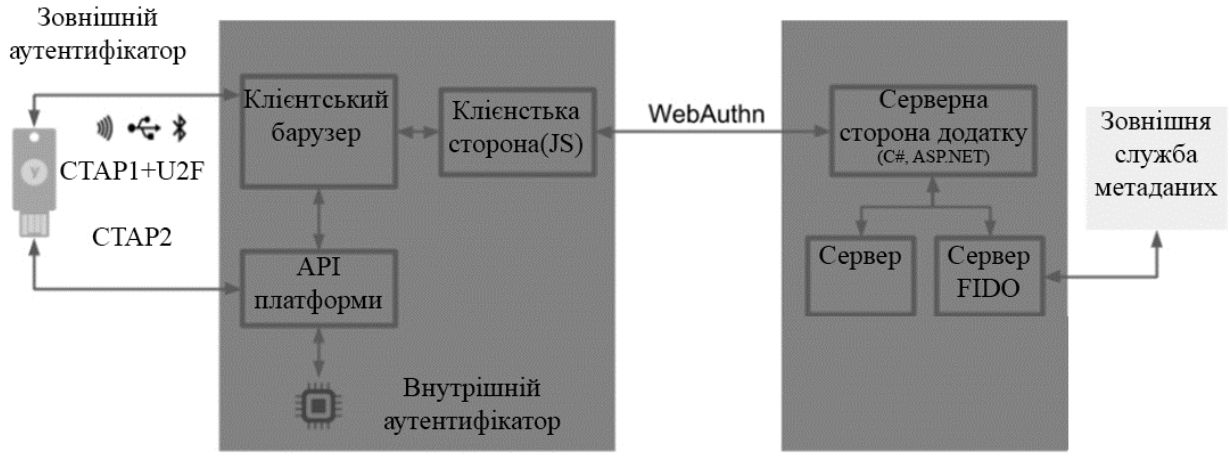

Рис. 2.3 – Архітектура системи

2.2.1 Пристрій клієнта

Користувач використовує клієнтський пристрій, щоб отримати доступ до програми. Як показано на ілюстрації, клієнтський пристрій служить важливим посередником, що з’єднує засоби автентифікації з клієнтськими функціями програми. У процесі розробки комплексної системи аутентифікації стає обов’язковим розуміти можливості, закладені в пристрої користувача. Це важливо, оскільки клієнтський пристрій зазвичай керує діапазоном рішень автентифікації, які можуть бути виконані через нього. Ретельний аналіз різних

користувацьких пристроїв показує широкий спектр підтримки різноманітних стандартів і методологій автентифікації.

Розглядаючи мобільний клієнт, наприклад, що працює на системах Android або iOS, він може містити вже існуючі функції, що підтримують біометричні рішення, такі як сканування відбитків пальців і розпізнавання обличчя. Подібним чином Windows 10 може похвалитися подібною функцією, відомою як Windows Hello, яка полегшує автентифікацію за допомогою розпізнавання обличчя, сканерів відбитків пальців і персональних ідентифікаційних номерів (PIN). Враховуючи те, що ці можливості інтегровані у функціональні можливості клієнта, програми на стороні клієнта можуть легко використовувати їх. Крім того, Windows 10 вдається до зовнішніх пристроїв для реалізації певних методів автентифікації, які вона підтримує. Варті уваги приклади включають YubiKeys, USB-пристрої, оснащені сканерами відбитків пальців, і веб-камери, призначені для цілей розпізнавання обличчя.

Під час підключення зовнішніх пристроїв автентифікації слід уважно розглянути протоколи CTAP2 і CTAP1. Отже, надзвичайно важливо, щоб клієнтські пристрої розширювали підтримку цих протоколів. Наразі CTAP має різний рівень підтримки майже в усіх браузерах і платформах. Браузери мають необмежений доступ до всіх функцій, у той час як платформи, що працюють на системі iOS, мають більш обмежені можливості. Однак сфера програмного забезпечення, що підтримує ці протоколи, динамічно розширюється, оскільки розробники постійно впроваджують інновації.

Для кожного клієнтського пристрою створено різні типи програм. Примітні приклади включають програми Android або iOS для мобільних пристроїв і програми Windows, Mac і Linux для настільних систем. Крім того, веб-програми пропонують третій шлях, безперебійно функціонуючи на клієнтських пристроях із підключенням до Інтернету та сумісним браузером. Проте, коли веб-додаток намагається використовувати вбудовані функції, він

може зіткнутися з перешкодами без необхідної підтримки. Таким чином, стає обов’язковим розпізнати відповідний курс дій, якщо веб-додаток має на меті використовувати методи автентифікації на основі пристрою.

У цьому ландшафті кожен параметр спілкується з серверною частиною програми за допомогою унікальної мови, властивої клієнтській програмі. Процес розробки поділяється на зовнішню та бек-енд розробку, також відому як розробка на стороні клієнта та розробка на стороні сервера. Клієнтська сторона програми представляє частину, з якою клієнтський пристрій активно взаємодіє.

2.2.2 Застосунок клієнта

Видимий аспект програми, з якою користувачі взаємодіють у своїх браузерах, становить клієнтську сторону. Його основна мета — забезпечити інтерфейс для функціональності сайту, що полегшує взаємодію з користувачем. Усі дані, необхідні програмному забезпеченню на стороні сервера, такі як імена користувачів, паролі та різні ключі, передаються через цю клієнтську програму.

Традиційне програмне забезпечення розробляється на конкретних мовах, адаптованих для конкретних операційних систем; наприклад, Kotlin для Android, Swift для iOS і C# для Windows. І навпаки, клієнтська сторона веб-додатків часто створюється за допомогою мови JavaScript.

Що стосується доступу до програми, рідні програми встановлюють зв’язок із клієнтським пристроєм, тоді як веб-програми спілкуються через браузер. WebAuthn API покладається на JavaScript, щоб служити мостом між браузером і серверною частиною. Отже, взаємодія між призначеною службою та клієнтом може використовувати оманливий інтерфейс під час реалізації автентифікації FIDO2.

2.2.3 Аутентифікатори

Користувачі можуть пройти автентифікацію через платформу, внутрішній скануючий пристрій або зовнішній скануючий пристрій. Роль автентифікатора полягає в тому, щоб надати доказ заявленої особи користувача разом із конфіденційним елементом. У контексті використання статичних паролів самі користувачі можуть бути призначені автентифікаторами. Вкрай важливо захистити інформацію, що зберігається автентифікаторами, оскільки вона служить для підтвердження особи та/або присутності користувача.

Уразливість статичних паролів полягає переважно в їх простоті. Навпаки, динамічні паролі або криптосистеми з відкритим ключем не мають цієї слабкості. Ці альтернативи використовують секретні ключі або динамічні значення замість звичайних паролів. Оскільки користувачам складно запам’ятати ці секрети, існує потреба в безпечному місці зберігання. Отже, значення автентифікатора або методів для зберігання інформації автентифікації зростає в сфері безпеки.

Розглядаючи зовнішні ключі, такі як YubiKey, важливо визнати використання в цих пристроях фірмової технології. Будь-який клієнтський пристрій, що взаємодіє з ними, повинен мати сумісну мову спілкування. На щастя, два протоколи, а саме U2F/CTAP1 і CTAP2, були розроблені для полегшення цієї взаємодії. Це об’єднання підвищує ефективність зовнішніх ключів як безпечного рішення в криптосистемах. Сучасні версії YubiKey включають вбудовану перевірку користувача та реєстрацію присутності користувача, що робить його чудовим варіантом для автономної автентифікації.

Альтернативний підхід передбачає використання функцій, властивих пристрою, відомих як аутентифікатори платформи. Багато з цих методів автентифікації сумісні з криптосистемами з відкритим ключем і часто включають біометричні методи, такі як сканування відбитків пальців і розпізнавання обличчя. На рисунку 2.3 показано, що вбудовані методи

автентифікації не вимагають використання клієнтом CTAP1 або CTAP2, оскільки вони зазвичай підтримуються клієнтським пристроєм.

На малюнку показано візуальне представлення того, як працюють автентифікатори та їх послідовний потік. Наступні події після цієї взаємодії з'ясовують зв'язок між автентифікаторами та клієнтським пристроєм.

2.2.4 Застосунок сервера

Він функціонує як програма на стороні сервера, полегшуючи обмін запитами та відповідями між сервером і клієнтом. Цей компонент, званий «back end», залишається непомітним для користувачів, але відіграє вирішальну роль у посередництві зв'язку між клієнтом і сервером. Як правило, коли користувач взаємодіє з такими елементами, як кнопки в інтерфейсі клієнтської програми, відповідний метод виконується на серверній програмі.

Серверна сторона служить місцем розташування структури та функціональності програми, методів розміщення, які полегшують зв'язок між клієнтом і сервером. Це програмне забезпечення взаємодіє з клієнтською програмою за допомогою методів, здатних запитувати інформацію або пропонувати послуги клієнту. Згодом, залежно від характеру дії, сервер може передати цю інформацію назад на сервер. У сценаріях, що включають автентифікацію, цей обмін може відбуватися під час реєстрації або під час перевірки відповідей, отриманих клієнтською програмою.

При розробці серверної програми, адаптованої для різних клієнтських пристроїв, застосовуються різні мови програмування. Windows, наприклад, висуває певні мовні вимоги під час розробки програми. Подібні обмеження виникають під час створення мобільних додатків для таких платформ, як iOS

або Android. Отже, вибір підтримуваних мов стає критичним питанням, що залежить від конкретної програми, що розробляється.

2.2.5 Серверний пристрій

Сервер служить сховищем для всіх даних, які програма повинна зберігати. Щодо автентифікації, то зазвичай вона включає дані, пов’язані з користувачем, і спеціальні облікові дані автентифікації, які використовуються для перевірки спроби входу. Це може включати таку інформацію, як електронна адреса виборця, номер мобільного телефону, хеш пароля тощо.

Різні методи автентифікації вимагають різних рівнів безпеки на відповідних серверах. Наприклад, статичний пароль вимагатиме найкращих заходів безпеки, якщо його зберігати на сервері, оскільки лише володіння цим паролем дозволить зловмиснику отримати доступ до облікового запису.

Навпаки, якщо використовується криптосистема, інформація, що зберігається на сервері, не накладає таких же суворих вимог. Це пояснюється тим, що закритий ключ, який використовує користувач для входу, ніколи не передається на сервер для зберігання. Дані, отримані сервером, служать виключно для аутентифікації закритого ключа користувача. Отже, вирішуючи зберігати інформацію на сервері, ретельний розгляд вимог безпеки має вирішальне значення.

Цього можна досягти шляхом впровадження шифрування, маркерів або маскування на рівні бази даних. Крім того, додаткові заходи, такі як списки контролю доступу та дозволи, можуть бути включені на основі конкретних потреб.

2.2.6 Довіряюча сторона

Довіряюча сторона або Relying party (RP) складається принаймні з веб-сервера та сторони сервера веб-додатку. Це об’єкт, який запитує автентифікацію користувача. Це можна зробити за допомогою IdP або безпосередньо за допомогою FIDO2. У випадку FIDO2 він зберігає всю необхідну інформацію про користувачів, включаючи інформацію про метод автентифікації. Тут важливо зауважити, що закритий ключ не включено до цієї інформації. RP також може мати зовнішні служби для перевірки даних, такі як Attestation Trust Store. Це довірче сховище необхідне, лише якщо RP піклується про метадані атестації. Довіряюча сторона спілкується лише з клієнтом або IdP, таким чином залишаючи роботу зв’язку з автентифікатором клієнту.

2.2.7 Постачальник ідентифікаційних даних

Постачальник ідентифікаційної інформації (Identity Provider або IdP) – це об’єкт, який використовується в протоколах об’єднання для зберігання інформації про користувача. Користувач зберігатиме свою інформацію тут у безпеці, а RP знатиме, що автентифікації через IdP можна довіряти. Тому він вважається важливою сутністю як для користувачів, так і для RP.

2.2.8 Клієнт

Клієнт можна вважати мостом між RP і автентифікаторами. Він реалізує способи зв'язку з різними методами аутентифікації та встановлення з'єднання з RP. Найкращими прикладами клієнтів є апаратні пристрої, такі як ПК і телефони. ПК зазвичай запускає веб-програму, яка працює в браузері. З’єднання з веб-програмою/браузером — це те, як ПК спілкується з RP. Щоб

RP міг отримувати інформацію від автентифікатора, ПК повинен мати спосіб автентифікації. Для ПК найпоширеніший варіант - вставити зовнішній ключ у USB-вхід. Для мобільних клієнтів, таких як android та iOS, методи автентифікації також включені безпосередньо в клієнт. Це означає, що мобільні пристрої зазвичай одночасно є клієнтом і автентифікатором.

2.2.9 Зовнішній автентифікатор

Загалом, зовнішній автентифікатор – це те, що використовується для підтвердження особи та присутності користувача. Оскільки цей автентифікатор зазвичай є фізичним пристроєм, який користувач має на собі або поблизу, він вважається хорошим доповненням до безпеки користувача. Це пов’язано з тим, що багато зловмисних атак включають видавання себе за іншу особу. З таким пристроєм зловмисник мав би заволодіти пристроєм, щоб підключитися до облікового запису користувача, навіть якщо він знає іншу інформацію користувача. Коли говориться конкретно про FIDO2, зовнішній автентифікатор зазвичай є якимось ключем, який можна використовувати з USB, NFC або Bluetooth. Хорошим прикладом є YubiKey, який пропонує одноразовий пароль, криптографію з відкритим ключем і автентифікацію, а також підтримується протоколами U2F і FIDO2, розробленими альянсом FIDO. Це дозволяє користувачам автентифікувати себе за допомогою OTP або пари відкритий/приватний ключ. Приватний ключ ніколи не залишить ключ безпеки, лише відкритий ключ буде надіслано на сервер для зберігання. Для YubiKey потрібно виконати жест авторизації у вигляді торкання, щоб підтвердити присутність користувача.

У підтримуваних FIDO2 автентифікаторах є щось, що називається закритим ключем атестації. Цей ключ записується в пристрій автентифікації під час створення для використання для підпису облікових даних, включаючи відкритий ключ, який пізніше використовується в процесі FIDO2. Атестація

залежить від типу та моделі використовуваного пристрою, тобто YubiKey 5 та мобільний телефон з відбитком пальця матимуть власний унікальний ключ. Він використовується для підтвердження того, що користувач має певну модель пристрою під час реєстрації. Метою атестації є забезпечення цілісності повідомлення, яке надсилає користувач. Це тому, що зловмисник не зможе змінити відкритий ключ, не змінивши підпису. Це тому, що відкритий ключ було підписано закритим ключем, який зберігається на пристрої. Це дозволяє службі бути впевненим у тому, що все, що надсилається від цього користувача, справді безпечне.

2.2.10 Аутентифікатор платформи

Автентифікатори платформи – це методи автентифікації, включені в пристрій користувача, будь то ПК чи мобільний пристрій. Одним із головних недоліків зовнішнього автентифікатора є його універсальність. Це означає, що автентифікатор платформи можна використовувати лише на пристрої, на якому він реалізований. До пристрою також встановлюються деякі вимоги, щоб він міг використовувати автентифікатори платформи. Одним із них є мікросхема безпеки Trusted Platform Module (TPM) для обробки відкритих і закритих ключів, які використовуються в процесі автентифікації. Ще однією вимогою є камера або біометричний зчитувач, який підтримує тип введення, необхідний для методу автентифікації. Наприклад, якщо ви хочете використовувати відбиток пальця як спосіб автентифікації, пристрій повинен мати якийсь датчик, щоб це стало можливим. Це також стосується інших біометричних рішень. Зауважте, що це не обов’язкова вимога для автентифікатора платформи, але у випадку, коли потрібно використовувати сканування відбитків пальців, для цього має бути реалізовано відповідне рішення.

2.3 Принцип роботи протоколу

Зв’язок між клієнтом і автентифікатором в API WebAuthn управляється двома ключовими методами: .create() і .get(). Ці методи відіграють вирішальну роль у розробці платформи, зумовлюючи необхідність створення конкретних об’єктів для ініціації. У посібнику зі специфікації W3C ці об’єкти називаються CreationOptions і RequestOptions. Перший містить деталі для створення облікових даних або реєстрації, тоді як другий містить інформацію, що стосується автентифікації та входу.

Слід зазначити, що ці об’єкти мають як обов’язкові, так і дискреційні параметри. Необхідно вказати обов’язкові параметри, тоді як необов’язкові пропонують додатковий рівень безпеки для потоку FIDO2. При передачі через API WebAuthn ці об’єкти досягають клієнтського пристрою або автентифікатора. Таким чином, підтримання узгодженості у створенні цих об’єктів є життєво важливим для того, щоб система автентифікації відповідала функціональності API. Отже, основною метою програми є максимізація підтримки API. [12]

Об’єкти, надіслані з сервера, викликають відповіді від автентифікатора, і друга роль системи включає перевірку та ефективну обробку цих відповідей. Вміст відповіді частково залежить від параметрів, інтегрованих в об’єкти CreationOptions і RequestOptions. Наприклад, якщо відповідь стосується першого, вона повинна містити інформацію про створення облікових даних, тоді як у випадку RequestOptions ключ полягає в підписанні відповіді закритим ключем для підтвердження особи користувача. Отже, система виконує дві основні функції:

1. Створення сутності, що включає вимоги та параметри щодо того, як облікові дані мають створюватися або запитуватися.

2. Перевірка відповіді автентифікатора для забезпечення дотримання вимог, встановлених на першому кроці. Згодом відбувається створення облікових даних, готових для зберігання в базі даних.

2.3.1 Реєстрація

Перш ніж розпочати фактичну процедуру реєстрації FIDO2, користувачі зазвичай вводять певні дані про себе, наприклад ім’я користувача або пароль. Ця введена інформація служить основою для подальшої реєстрації облікових даних FIDO2. Після надання цих даних серверу не потрібна додаткова інформація від користувача, щоб ініціювати створення облікових даних FIDO2.

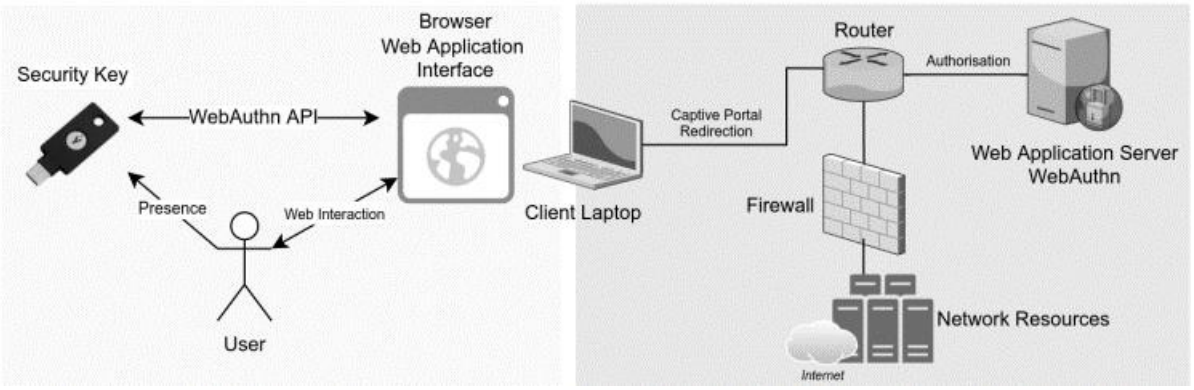

Рис. 2.4 –схематичне зображення процесу реєстрації з використанням WebAuth/FIDO.

Після того, як система прийме дані користувача, вона переходить до заповнення різних параметрів в об’єкті, готуючи його до виклику WebAuthn. Це заповнення параметрів в об’єкті є одним із важливих завдань на стороні сервера.

Після виклику WebAuthn вибірково використовує параметри в об’єкті, щоб ініціювати створення облікових даних. Характер створених облікових даних залежить від параметрів, надісланих сервером, і типу клієнтського пристрою, який використовується. Згодом зовнішній автентифікатор або

автентифікатор платформи отримує ці специфікації, генеруючи об’єкт Credential і повертаючи його перевіряючій стороні.

Потім служба ретельно перевіряє умови, викладені в CredentialOptions. Після перевірки відповіді автентифікатора об’єкт Credential готовий до збереження на сервері.

У CredentialObject більшість полів є необов’язковими. Тим не менш, три змінні — параметри «Виклик», «Сервер» і «Користувач» — вимагають заповнення з сервера. Критичним аспектом є генерація мінімум 16 криптографічно рандомізованих байтів на сервері, стратегічно реалізованих для запобігання атакам відтворення.

Параметр сервера передбачає обов’язкове поле ID, яке зазвичай представляє вихідну URL-адресу служби. FIDO2 диктує, що цей ідентифікатор має бути HTTPS, і автентифікатор покладається на цей ідентифікатор сервера, щоб встановити область для свого закритого ключа.

Як остаточний обов’язковий параметр має бути вставлена інформація про користувача. Ім’я та ідентифікатор є обов’язковими для цього параметра, а найкращі практики рекомендують виключати ідентифікатор особистої інформації. Його призначення полягає в тому, щоб пов’язати користувача з пов’язаними обліковими даними.

Мета присвоєння імені ідентифікатору полягає в тому, щоб надати мітку, яку можна прочитати людиною, і полегшити диференціацію серед користувачів, які мають спільне відображуване ім’я. Сервер зобов’язаний заповнити ці три параметри, але окрім них, стандарт FIDO2 містить численні додаткові параметри, які можна включити у структуру. Інтеграція цих додаткових параметрів служить для зміцнення безпеки, спонукаючи розробників включити підтримку для різноманітних компонентів.

На відміну від використання одноразових паролів, стандарт FIDO2 стикається з проблемою розміщення різних типів пристроїв у процесі зв’язку.

Це пов’язано з тим, що не всі методи автентифікації, які використовують криптографію з відкритим ключем, схвалені FIDO2, є загальнодоступними на всіх пристроях. Залежно від уподобань розробника, не кожен пристрій може бути визнано придатним для використання. Отже, існує великий інтерес до впровадження підтримки різноманітних клієнтських пристроїв для максимального підвищення зручності використання. Система повинна включати методи або об’єкти, адаптовані для таких платформ, як Android, iOS, ПК тощо. Розглянемо, наприклад, пристрій Android; коли користувач отримує доступ до сайту за допомогою такого пристрою, послуга повинна включати підтримку методів автентифікації, сумісних із цим конкретним пристроєм.

Переважним підходом у цьому контексті є використання форматів атестації. Кожен пристрій із підтримкою FIDO2 оснащено форматом атестації, властивим процесу його виробництва. Приклади включають формат Android для пристроїв Android і формат FIDO-U2F для U2F. Ці формати інкапсулюють всю інформацію, яку може надати певний тип пристрою. Додатковий параметр, відомий як "attestation", дозволяє включити значення для визначення використання цих форматів. Якщо сервер вирішує використовувати формат атестації, закритий ключ атестації в пристрої підпише відкритий ключ, згенерований під час реєстрації, перед передачею. І навпаки, якщо цей параметр не вибрано, відкритий ключ буде передано без підпису, без участі закритого ключа атестації.

У додаткових параметрах, що передаються в CredentialObject, існує можливість явного визначення прийнятних алгоритмів відкритих ключів для сервера. Отже, структура повинна володіти можливістю пристосовувати ці визначені типи для ефективної обробки відповіді з використанням будь-якого з алгоритмів, описаних в об’єкті. Серед цих алгоритмів автентифікатор виконає перший, сумісний із пристроєм, підкреслюючи необхідність визначення пріоритетності списку прийнятних алгоритмів.

Сервер повинен адаптувати свою пропозицію алгоритмів на основі використовуваного пристрою, забезпечуючи повне покриття більшості пристроїв. Цей підхід передбачає перерахування прийнятних алгоритмів у порядку пріоритету, що сприяє бездоганній сумісності з різними пристроями.

Інший елемент у межах об’єкта забезпечує гнучкість визначення переваги платформи або зовнішніх автентифікаторів. Ця функція дає розробникам змогу встановлювати обмеження щодо типів автентифікаторів, допустимих у рамках розробки.

Ці 3 параметри є основними засобами, за допомогою яких стандарт FIDO2 може визначати та налаштовувати криптографічні пристрої та рішення з відкритим ключем, доступні як для користувачів, так і для автентифікаторів. Вони значно сприяють адаптивності фреймворку, дозволяючи приймати деталізовані рішення щодо параметрів алгоритму та типів автентифікаторів, інтегрованих у систему.

Численні додаткові параметри можуть бути включені в CredentialObject. Хоча розробники прагнуть включити якомога більше цих параметрів, основний акцент має залишатися на двох сценаріях, виділених раніше. Форматування JSON є обов’язковим під час надання CredentialObject виклику .create().

Під час процесу перевірки ретельна перевірка всіх параметрів у CredentialObject є важливою для забезпечення відповідності інформації у відповіді. Отже, кожен параметр, вставлений у CredentialObject, повинен мати відповідні методи перевірки.

Відповідь автентифікатора складається з двох ключових об’єктів: clientDataJSON і attestationObject. Ці сутності несуть інформацію, що стосується специфікацій, викладених у створюваному об’єкті. Об’єкт clientDataJSON охоплює дані, пов’язані з взаємодією з браузером або сервером, потенційно розкриваючи інформацію у разі зловмисних дій під час спілкування.

З іншого боку, attestationObject надає детальну інформацію про дії автентифікатора та інформацію про пристрій. Він охоплює відкритий ключ і різні докази, що підтверджують особу автентифікатора. Цей об’єкт зведений у стисле двійкове представлення, широко відоме як CBOR. Отже, система повинна мати можливість розпакувати його. Вимоги до сервера мають полегшити порівняння між цими двома об’єктами.

Якщо сервер запитує формати атестації, він повинен мати можливість інтерпретувати відповідь на основі типу пристрою, з якого вона надійшла. Для цього потрібен доступ до служби метаданих, здатної перевірити певний тип пристрою.

Крім того, автентифікація FIDO2 може бути виконана без використання форматів атестації, хоча за рахунок того, що сервер не отримує інформації про тип використовуваного пристрою. У таких випадках стороння служба метаданих не вимагає визначення типу пристрою.

Користь використання форматів атестації полягає в його здатності розрізняти тип пристрою та полегшувати визначення можливостей пристрою. Крім того, це слугує запобіжним заходом, гарантуючи, що пристрої без зазначеного формату не зможуть автентифікуватися на сервері. [12]

Після перевірки всієї відповідної інформації серверу доручено створити облікові дані, які містять важливі дані для входу користувача. Ці облікові дані повинні, як мінімум, містити ідентифікатор облікових даних, ідентифікатор користувача та відкритий ключ.

2.3.2 Аутентифікація

Розробка системи вимагає ретельного розгляду процесу автентифікації, який має спільні риси з іншими функціями системи, але також демонструє

відмінні характеристики. Основний принцип роботи відповідає малюнку 2.3, як показано в процесі реєстрації. Однак фазу автентифікації відрізняють нюанси у вмісті та параметрах.

Починаючи із запиту користувача на вхід через клієнтську програму, автентифікація розгортається у відповідь на це прохання. Вхідні дані, надіслані користувачем на сервер, можуть бути використані для створення RequestObject. Після отримання запиту на вхід повіряюча сторона формує RequestObject, що містить специфічні для системи параметри, і передає їх до WebAuthn API. API WebAuthn, у свою чергу, розгортає об’єкт для виклику .get(), запускаючи процес автентифікації та запитуючи відповідь від автентифікатора. Ця відповідь згодом передається назад на сервер. Подібно до процесу реєстрації, потік автентифікації інкапсулює окрему інформацію та реквізити в своїх об’єктах.

Об’єкт RequestObject передбачає виклик як єдину вимогу, надаючи менше опцій порівняно з реєстрацією, але зберігаючи структуру, подібну до неї. Хоча більшість варіантів є дискреційними, бажано включити якомога більше. Важливим параметром, унікальним для потоку автентифікації, є «allowCredential». Тут сервер має облікові дані для автентифікації користувача, використовуючи ідентифікатор облікових даних, отриманий під час реєстрації, для запиту тих самих облікових даних. Крім того, цей параметр дозволяє вказати бажаний пристрій автентифікації, будь то USB, NFC або Bluetooth.

Подібно до процесу реєстрації, необхідно розробити підходи для перевірки відповідей автентифікатора. Отже, методи перевірки є важливими для додаткових параметрів, включених до RequestObject.

У відповіді на аутентифікацію виявляється помітне відхилення від процесу реєстрації – відсутність відкритого ключа. Натомість відповідь у цьому контексті включає виключно підпис, прикріплений закритим ключем. Згодом

перевірка цього підпису відбувається за допомогою попереднього відкритого ключа на сервері.

Це підкреслює важливість використання підтримуваних алгоритмів. Сервер, використовуючи попередньо зареєстрований відкритий ключ і криптографічний алгоритм, вибраний для облікових даних, розшифровує виклик, підтверджений закритим ключем. Цей складний процес забезпечує цілісність і безпеку механізму автентифікації. [12]

2.4 Висновки до другого розділу

У висновку до другого розділу створення двофакторної системи аутентифікації були розглянуті різні важливі аспекти, що надає повне розуміння сучасного ландшафту технологій аутентифікації. Аналіз сучасних технологій аутентифікації висвітлив змінюючийся характер заходів безпеки в веб-застосунках. Складна архітектурна концепція підкреслила важливість надійної структури для забезпечення ефективності двофакторної системи аутентифікації.

Більше того, обговорення принципу роботи протоколу FIDO2 пролило світло на передовий стандарт, наголошуючи на його ролі у покращенні безпеки та користувацького досвіду. Розгляд робочого алгоритму протоколу FIDO2, разом зі вглибленням у програмний інтерфейс WebAuthn, розкрив його особливі переваги перед іншими варіантами аутентифікації. Рекомендації та найкращі практики для впровадження цього стандарту були ретельно розглянуті, надаючи практичне керівництво для безшовної інтеграції в різноманітні системи.

Аналіз новизни представленого програмного інтерфейсу включав демонстрацію обсягу сумісних браузерів і відповідного користувацького

базису. Цей аналіз підкреслив помітну тенденцію до збільшення кількості додатків, що отримали підтримку API, що свідчить про колективний рух галузі до більш безпечних та зручних практик аутентифікації. Водночас спостерігався помітний занепад використання застарілих чи непопулярних версій браузерів, що свідчить про зростаючу усвідомленість та прийняття сучасних стандартів аутентифікації.

РОЗДІЛ 3. ПРОГРАМНИЙ МОДУЛЬ ПОДВІЙНОЇ АУТЕНТИФІКАЦІЇ ДЛЯ ВЕБЗАСТОСУНКІВ

3.1 Огляд використаних технологій

У цьому розділі обговорюються вимоги до універсальної системи автентифікації, яку планується розробити. У двох попередніх розділах було обговорено різні технології, які є доречними, коли справа доходить до створення структури автентифікації. Тепер необхідно дати огляд вимог, які було встановлено для розроблювального фреймворку. Дизайн всієї системи повинен бути інтуїтивно зрозумілим, щоб ним міг ефективно користуватися будь-який користувач. Він також повинен пропонувати рішення, які не мають очевидних чи багатьох слабких місць. Пропоноване рішення має бути безпечним і практичним, щоб користувачам було комфортно ним користуватися. Для цього в цій роботі розділив вимоги на дві категорії: функціональні та нефункціональні.

- Підтримка мов веб-програмування

Фреймворк, який буде розроблено, повинен підтримувати загальні/стандартні мови веб-програмування, щоб надати розробнику менше обмежень, коли справа доходить до того, які мови вони хочуть використовувати для своєї програми.

- Підтримка зовнішнього ключа.

Структура повинна підтримувати використання зовнішніх ключів як засобу автентифікації. Це ключі, які використовують USB, NFC або Bluetooth для зв’язку з клієнтським пристроєм. Деякі ключі також пропонують декілька таких рішень в одному ключі.

- Внутрішня підтримка автентифікації

Структура повинна підтримувати різні типи методів автентифікації, які пропонуються нативними для мобільних пристроїв. Прикладом цього є сканер відбитків пальців і PIN-коди.

- Рішення динамічної автентифікації

Структура повинна підтримувати рішення динамічного пароля як метод автентифікації. Це має на меті запропонувати додаткову можливість автентифікації для розробника.

- Автентифікація на основі ризиків

Необхідно створити базове рішення автентифікації на основі ризиків. Це допоможе програмі оцінити загрозу спроби входу.

- Зробити фреймворк зручним для розробників

Розробнику має бути легко використовувати функціональні можливості, реалізовані у фреймворку. Тому важливо, щоб структура включала більшість функціональних можливостей, необхідних для належного проходження автентифікації.

- Запобігання атакам на методи автентифікації

Необхідно вжити заходів щодо запобігання типам атак, які були обговорені в попередньому розділі

- Можливі атаки на методи автентифікації.

Це зроблено для того, щоб зменшити ймовірність будь-яких зловмисних атак на використовувані методи автентифікації.

Програмний модуль для подвійної автентифікації був розроблений з використанням комбінації інструментів, включаючи мову програмування C#, фреймворк ASP.NET Core, бібліотеку fido2 net lib і платформу Microsoft Identity. Ці ресурси відіграли вирішальну роль у створенні серверної складової проекту. Платформа Microsoft Identity, відома своїми надійними можливостями, полегшує створення облікових записів, автентифікацію, авторизацію тощо.

Одночасно бібліотека FIDO2 дає змогу застосовувати універсальний другий фактор до раніше створених облікових записів користувачів.

Вибір фреймворка ASP.NET Core обумовлений його винятковою майстерністю в порівнянні з аналогічними веб-технологіями на альтернативних мовах. У рейтингу з понад 120 фреймворків ASP.NET Core займає 8 позицію, обігнавши багатьох конкурентів і поступаючись лише мовам низького рівня, таким як C++ або Rust. Примітно, що він перевершує такі динамічні мови, як Python із його Django та Flask, а також PHP із Symfony та Laravel, встановлюючи його перевагу з точки зору продуктивності. [17]

Рис. 3.1 – Блок схема: Реєстрація WebAuthn із доступними обліковими даними.

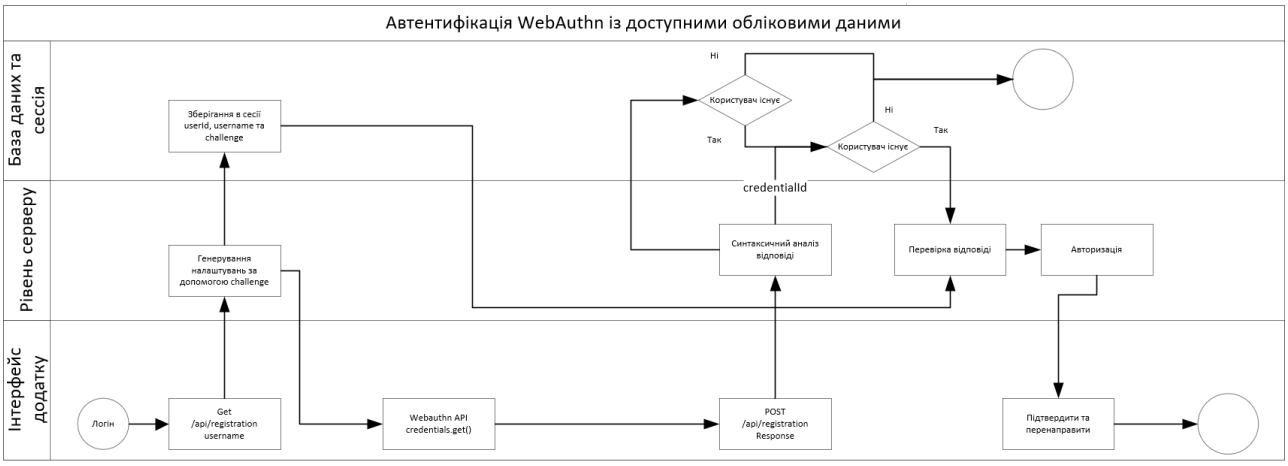

Рис. 3.2 – Блок схема: Автентифікація WebAuthn із доступними обліковими даними.

Веб-програма автентифікації WebAuthn відповідає класичному шаблону розробки програмного забезпечення Model-View-Controller (MVC). У цьому розділі детально описано збережені дані, необхідні для програми: модель архітектури програми. Дані, які зберігаються цією веб-програмою, безпосередньо відповідають мінімальній інформації користувача та його обліковим даним для автентифікації. Враховуючи реалізацію автентифікації, облікові дані автентифікації є відкритими ключами WebAuthn. Зокрема, веб-додаток обробляє два основні типи даних:

- Деталі веб-сеансу. Після автентифікації користувача починається веб-сеанс. У повній веб-програмі він дозволяє авторизувати веб-операції цьому автентифікованому користувачеві. За визначенням, ці деталі тісно пов’язані з веб-сеансом, тому вони не вважаються постійними.
- Користувач і його відповідні облікові дані. Для керування користувачами та автентифікації користувачі та їхні відповідні облікові дані зберігаються в постійній базі даних.

Деталі веб-сеансу спочатку є лише функціональними ідентифікаторами для контролерів для керування веб-сеансами.

Що стосується деталей користувача, на малюнку 3.3 показано діаграму класів з атрибутами конкретного користувача разом із моделлю облікових даних WebAuthn для зберігання пристроїв, зареєстрованих для автентифікації цього користувача

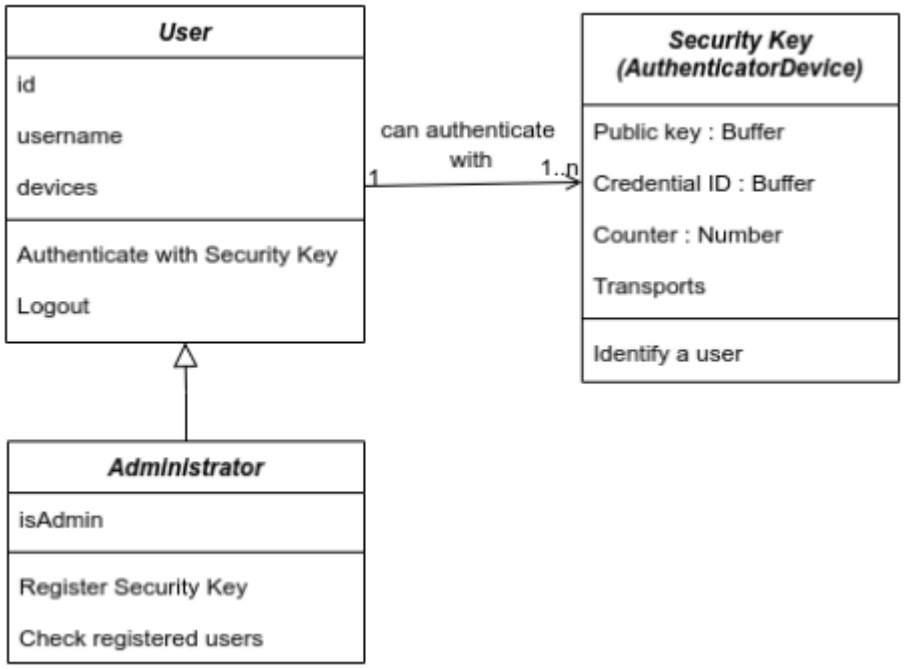

Рис 3.3: Діаграма класів користувача та облікових даних. Користувач має принаймні один пов’язаний ключ безпеки.

Розвиток цієї функції є основною характеристикою сервера аутентифікації WebAuthn. Це дозволяє використовувати ключі безпеки WebAuthn, сумісні з доступними обліковими даними. Зокрема, пристрої можуть бути зареєстровані, пов’язані з користувачем, а потім використані під час автентифікації, тобто під час перевірки особи користувача.

Є два типи облікових даних WebAuthn, пов’язаних із ключами безпеки: видимі та невидимі. Облікові дані, які можна знайти, фізично зберігаються в ключі безпеки. Навпаки, облікові дані, які не можна знайти, зберігаються на сервері в зашифрованому вигляді.

Як перша розробка, сервер був розроблений для підтримки видимих облікових даних. Під час автентифікації користувача сервер WebAuthn надсилає запит на ключ безпеки без попередньої ідентифікації користувача. Отримавши відповідь від ключа безпеки, сервер може ідентифікувати відповідного користувача за допомогою параметра userHandle.

На рисунку 3.1 показана міжфункціональна діаграма з потоком даних операції реєстрації. По-перше, при реєстрації вказується ім'я користувача. Якщо користувач не існує, створюється новий користувач. Потім пристрій (ключ безпеки) реєструється для цього користувача.

Під час автентифікації за допомогою ключа безпеки ім’я користувача не потрібне. Як показано на міжфункціональній діаграмі на малюнку 3.2, userHandle, включений у відповідь автентифікатора, використовується як ідентифікатор користувача для пошуку зареєстрованого пристрою для перевірки відповіді автентифікатора. Отже, використання доступних облікових даних WebAuthn означає, що користувач ідентифікується після автентифікації облікових даних. Таким чином, користувачеві не потрібно ідентифікувати себе за допомогою імені користувача під час процесу.

3.2 Реалізація фреймворку для програмного модулю

Незважаючи на те, що багато функціональних можливостей у існуючій структурі FIDO2 вже працює, необхідно дати пояснення до більш важливих реалізацій, які підтримуються. Це включає параметри об’єктів, які надсилаються до WebAuthn API, а також те, як відповіді обробляються фреймворком.

Під час реалізації фреймворку FIDO2 є чотири основні сценарії, які фреймворк повинен обробляти. По-перше, маємо два сценарії для процесу реєстрації у FIDO2; створення параметрів облікових даних, а також перевірка та реєстрація відповіді від автентифікатора. Під час реєстрації створюється відкритий ключ, необхідний для автентифікації користувача. Як уже згадувалося, є два сценарії, через які повинні пройти, щоб це зробити.

Перший — для RP створює об’єкт, включаючи параметри, що описують уподобання щодо того, як має бути створений об’єкт облікових даних користувача.

Другий метод включає обробку та перевірку інформації, отриманої від користувача, і створення об’єкта, який може зберігатися RP.

Інші два сценарії — коли вже наявний користувач хоче увійти в програму А саме, створення варіантів твердження та перевірка відповіді твердження. Параметри твердження такі ж, як і параметри запиту, які обговорювали в розділі про дизайн. Однак у реалізації будемо називати це параметрами твердження. Створення параметрів підтвердження можна вважати схожим на створення параметрів облікових даних. У цьому методі RP також надсилає об’єкт із набором параметрів, які встановлюють уподобання щодо відповіді. Перевірка відповіді є другим методом і є методом, який обробляє та перевіряє інформацію, отриману від користувача. Однак збережена інформація також буде використовуватися під час перевірки, включаючи відкритий ключ. Використовуючи ці чотири сценарії, розглянемо кожен із них і пояснимо, як це було реалізовано та як це вирішує нашу проблему.

Перший сценарій — це коли RP отримує запит на реєстрацію від користувача та потребує створення параметрів облікових даних. Нижче ви можете побачити малюнок, який показує структуру цього об’єкта з усіма параметрами, які він включає.

Усі ці параметри представляють те, що стандарт FIDO2 визнав корисним для реалізації безпечного рішення автентифікації. Як зазначали в розділі про дизайн, потрібні лише деякі з них, але в цьому рішенні більшість із них реалізовано. Тепер розглянемо, що пропонує фреймворк впровадження кожного з них.

Обов'язкові поля:

Завдання: створюється перевіряючою стороною, має довжину принаймні 16 байтів і рандомізовано, щоб запобігти атакам відтворення. У цій реалізації це зберігається в сеансі до завершення реєстрації. Це допоможе запобігти атакам повторного відтворення, оскільки відповідь має надіслати правильні байти, щоб відповідь була прийнята. У цій реалізації було використано метод

RandomNumberGenerator.Create, включений у C#. Після цього API WebAuthn вимагає, щоб запит був закодований за допомогою Base64URL. Тому реалізовано клас для обробки кодування та декодування Base64URL. Після того, як його було закодовано за допомогою Base64URL, його потрібно відформатувати в Uint8Array, щоб його прийняв API WebAuthn. Це також включає user.id, оскільки вони є двійковими параметрами.

Перевіряюча сторона: для параметра перевіряючої сторони в облікових даних у нас є три поля, які слід заповнити. Це ім’я, піктограма та ідентифікатор перевіряючої сторони. Це поле призначене для того, щоб автентифікатор дізнався, від кого надходить запит. Тут клієнт повинен вставити домен за допомогою HTTPS і додати його до об’єкта RP. RP.ID зберігається разом із закритим ключем на пристрої автентифікації. Це робиться для того, щоб приватний ключ, який використовується в цьому зв’язку, діяв лише для цього RP.

Користувач: останньою необхідною опцією є користувач, який має чотири поля. Це ім’я, значок, відображуване ім’я та ідентифікатор. З цих полів потрібні лише ім’я та ідентифікатор. Це поле призначене для реєстрації інформації про користувача, якому належать створювані облікові дані. Ідентифікатор, який надсилається в API WebAuthn, має бути послідовністю байтів із максимум 64 байтів. Клієнт має ввести ці значення. Цей ідентифікатор є тим, до чого буде прив’язано облікові дані відкритого ключа.

Необов’язкові поля: необов’язкові поля означають, що це поля, для яких клієнт має значення за замовчуванням, якщо RP не містить у них інформацію. Значення все ще настійно рекомендуються для додавання додаткової та індивідуальної безпеки до процесу реєстрації.

Вибір автентифікатора: за допомогою цього поля RP може вказати свої вимоги до автентифікатора. Він містить три різні поля, які мають різні параметри, що визначають вимоги, встановлені RP. Перше поле — це поле authenticatorAttachment, яке може бути порожнім, платформним або

міжплатформним. Значення, яке встановлюється, залежить від того, чи має RP певні вимоги до того, який тип автентифікатора слід використовувати. У випадку, коли встановлено значення «порожній», обмеження не встановлюються. Ця опція дозволяє веб-переглядачу знати, чи має він дозволити платформу чи зовнішній автентифікатор.

Друге поле – це поле requiredResidentKey, яке має значення true або false. Якщо це правда, автентифікатор створить резидентне джерело облікових даних закритого ключа під час створення відкритого ключа. Це означає, що приватний ключ і дескриптор користувача зберігаються на пристрої автентифікації користувача. Таким чином, у цьому сценарії RP має створити цей дескриптор користувача (User.ID) і надіслати його автентифікатору. Цей параметр використовується, коли ви хочете дозволити пароль і/або вхід без імені користувача [39].

У випадку, коли встановлено значення false, RP має зберігати облікові дані та інформацію про користувача на сервері. Нарешті, у нас є поле userVerification, яке містить значення; бажані, необхідні та небажані. Бажано означає, що перевірка користувача буде використана, якщо вона доступна. Обов’язковий дозволить лише автентифікатори з підтвердженням користувача, а не рекомендується використовувати його.

У цій структурі встановлено значення за замовчуванням для цих параметрів. Для authenticatorAttachment встановлено значення null, requiredResidentKey встановлено як false, а для userVerification — як preferred. Це означає, що у випадку, коли автентифікатор має можливість перевірити, він перевіряє.

PubKeyCredParams: визначає криптографічні алгоритми, які підтримуються RP. Це допомагає автентифікатору знати, які алгоритми можна використовувати під час шифрування відповіді. Порядок коду означає бажаний порядок, обраний сервером. Як показано на малюнку, цей параметр містить список різних підтримуваних алгоритмів, які підтримує RP. У цій структурі

було реалізовано алгоритми для підтримки всіх підтримуваних FIDO2 автентифікаторів. Це дає розробнику, який використовує фреймворк, свободу використовувати їх усі.

Тайм-аут: час, який має користувач, перш ніж вимагатиметься відповідь. Ця функція використовує мілісекунди, а рекомендований час становить 60 секунд. Тому за замовчуванням встановлено 60 секунд, але у випадку, коли цього вимагає безпека, час можна зменшити.

Список виключень облікових даних: цей параметр реалізовано, оскільки він дозволяє виключати певні облікові дані. Це список, надісланий RP, включаючи облікові дані, зареєстровані для конкретного користувача. У випадку, коли автентифікатор знаходить наявний тип облікових даних та ідентифікатор для RP, нові облікові дані не буде створено. Це запобігає багаторазовій реєстрації того самого автентифікатора на одному RP. Це дозволяє більш ефективно реєструвати кілька пристроїв автентифікації, знаючи, що всі зареєстровані є унікальними. Для цієї опції нам потрібно реалізувати спосіб створення об’єкта, який можна додати до цього списку. У цій структурі це називається дескриптором облікових даних відкритого ключа. Дескриптор містить ідентифікатор облікових даних, тип і, за бажанням, може містити метод транспортування.

Під час реєстрації користувача це буде додано як окреме поле на сервері. Якщо це поле зберігається на сервері, RP може включити їх у свій об’єкт createCredentialOptions, надісланий автентифікатору.

Атестація: у цьому полі є три варіанти на вибір; немає, непрямі та прямі. Якщо вибрано прямий доступ, сервер повинен отримати формат атестації від автентифікатора, щоб підтвердити тип автентифікатора. Щоб цей крок був можливим, RP має включити службу метаданих у свою реалізацію, як було обговорено в розділі про проектування. У цій реалізації включено службу метаданих, тому можливе використання атестації.

Служба метаданих — це сервер, що містить інформацію про всі різні формати атестації та типи пристроїв. І використовується в реалізації для пошуку та підтвердження інформації щодо типу пристрою, який використовується.

Якщо атестація непряма, це означає, що сервер дозволяє анонімні дані атестації. Це означає, що користувач може використовувати анонімізацію ЦС для створення заяв про атестацію, щоб захистити конфіденційність користувача. Це означає, що користувач може використовувати третю сторону для створення звітів, які надсилаються до RP.

Нарешті, маємо опцію «none», яка вказує на те, що сервер не зацікавлений у даних атестації. Значення за замовчуванням для поля атестації в цій реалізації - немає.

Як пояснювалося вище, ви бачите, що всі ці параметри, як обов’язкові, так і необов’язкові, є основою створення та додавання захисту об’єкта. Коли ці параметри реалізовано, вони включають багато функцій, які обговорювали в розділі про дизайн. Деякі з них вимагають передових методів, тоді як інші просто покладаються на вставку бажаного варіанту.

Після ознайомлення з цією інформацією стає зрозуміло, що клас, який містить ці параметри, має вирішальне значення для фреймворку. Це буде об’єкт, який буде надіслано як CreationOptions до методу .create() в API WebAutn.

Другий сценарій: перевірка та реєстрація облікових даних користувача. Другий важливий сценарій, звичайно, полягає в тому, як обробити відповідь, отриману після надсилання цього запиту користувачеві. Це сценарій, за якого сервер або приймає та створює облікові дані користувача, або відхиляє їх із повідомленням про помилку.

Коли користувач отримує запит із параметрами щодо створення облікових даних, користувачеві потрібно підтвердити його наявність,

натиснувши YubiKey, використовуючи відбиток пальця або будь-яке інше рішення. Після цього автентифікатор почне створювати облікові дані, перебираючи необхідні параметри, надіслані RP. Якщо все в порядку, автентифікатор надішле відповідь із усією інформацією, необхідною RP для створення облікових даних користувача.

У разі успішної відповіді RP повинен буде перевірити інформацію, яку надіслав користувач. Це включатиме перегляд усіх параметрів і перевірку того, що вони справді відповідають вимогам, надісланим спочатку RP. Це буде зроблено шляхом порівняння інформації, отриманої автентифікатором, з оригінальним об’єктом, надісланим RP. У цьому процесі також необхідно використовувати криптографічні методи для дешифрування та підтвердження правильності інформації. Нижче показано метод, який працює з цим сценарієм у рамках.

Перевіряє цю інформацію метод VerifyAsync. Тепер систематично переглянемо всі оператори if і пояснимо функціональність, що стоїть за ними.

Перше, що можемо побачити, це об’єкти, які використовуються в цьому методі. Це об’єкти, які використовуються для порівняння та перевірки відповіді. Про них буде згадано, коли будемо переглядати заяви. OriginalOptions — це об’єкт, який було надіслано в першому сценарії та використовується для порівняння.

По-друге, у нас є конфігураційний об’єкт, який містить типові значення для деяких значень.

По-третє, перевіряємо, чи ідентифікатор облікових даних не використовувався раніше. Після цього включаємо службу метаданих, яку використовуємо для атестації. Нарешті, у нас є ідентифікатор прив’язки маркера, який дає змогу переконатися, що зв’язок не було підроблено. Ці об’єкти є параметрами методу перевірки.

Перше, що включено до цього методу, це ще один виклик методу під назвою BaseVerify, який перевіряє, що запит, джерело, тип і прив'язка маркера, надіслані автентифікатором, містять ті самі значення, що й вихідні параметри. Як бачимо в заяві, це робиться шляхом прийняття виклику, отриманого від автентифікатора, і використання операторів для їх порівняння. Ці твердження мають на меті гарантувати, що спілкування між RP і користувачем не було маніпульовано.

Якщо знову поглянемо на структуру, перший перевірить, чи справді відповідь є викликом «webauthn.create», а не «webauthn.get». Якщо тип щось інший, буде видано помилку. Другий оператор перевіряє, чи Raw.Id є нульовим чи має довжину, що дорівнює нулю. Це має на меті перевірити, чи зареєстровано ідентифікатор для отриманої відповіді. Після цього твердження слідує твердження, що Raw.Type справді є відкритим ключем, що є скороченням від криптографічних алгоритмів із відкритим ключем. Наразі FIDO2 підтримує лише цей тип.

Нарешті, перевіряємо, чи відсутні authData. Це об'єкт, який містить дані автентифікатора, які мають використовуватися в створюваних облікових даних.

Після цього вони створюють нову змінну, включно з об'єктом authData, надісланим із автентифікатора. Вони також створюють дві змінні clientDataHash і rpIdHash. Ці змінні призначені для підготовки до перевірки об'єкта атестації.

Після цього алгоритм хешування SHA256 використовується для хешування двох нових змінних, які створили, clientDataHash і rpIdHash. clientDataHash приймає хеш-значення відповіді ClientDataJson, а rpIdHash приймає хешований оригінальний Rp.Id. Це робиться для того, щоб могли порівняти отримані значення з вихідними значеннями, надісланими RP.

Після цього виконується новий набір операторів if. По-перше, порівнюються rpIdHash, який щойно створили, і той, який отримано як відповідь. Якщо вони не збігаються, буде видано помилку. Це тому, що якщо

rpIdHash відрізняється, це означає, що запит, який отримав користувач, був з іншого джерела. Це означає, що могла бути якась зловмисна дія, яка намагалася змінити зв’язок між користувачем і RP.

Другий оператор if перевіряє, чи встановлено автентифікатором прапор User Present, а третій перевіряє, чи встановлено прапор Attestation. Присутній користувач має переконатися, що для ініціювання відповіді було використано певну дію, тоді як прапор Атестації підтверджує, що один із підтримуваних форматів FIDO2 використовувався автентифікатором. Це обов’язкові значення, які має заповнити автентифікатор, щоб мати змогу продовжити реєстрацію.

На цьому малюнку вони перевіряють формат атестації, який використовувався автентифікатором. Тут можемо побачити варіанти; none, tpm, androidkey, android-safetynet, fido-u2f та packed. Код переглядає різні можливості та встановлює об’єкт «перевірник» на об’єкт типу, який було вибрано.

Packed — це загальний формат атестації, що використовується в ключах, єдина мета яких — використовуватися як засіб автентифікації WebAuthn, наприклад ключі безпеки. Коли йдеться про настільні комп’ютери, зазвичай використовується формат TPM, тоді як формати Android використовуються, коли використовується пристрій Android. Нарешті, у нас є FIDO-U2F, старіші ключі, призначені для використання з U2F. [40]

У межах цього методу Verify(), який використовується в об’єкті верифікатора, міститься велика частина реалізації, яка стосується атестації. Залежно від використовуваного формату атестації, було реалізовано рішення для перевірки всіх підтримуваних форматів. Це включає підтримку алгоритмів, які підтримуються цим форматом. Серед них у нас є алгоритми хешування та PKC, щоб мати можливість перевірити відповідь. Алгоритми, реалізовані для обробки PKC у цій структурі, перераховані вище, тоді як SHA256-512 використовується для хешування.

Якщо тип атестації не включено в структуру, видається помилка про те, що надісланий тип не існує. У випадку, якщо формат не використовується, для нього потрібно встановити значення «немає».

Оператор if перевіряє, чи ідентифікатор облікових даних уже зареєстрований у користувача. Це робиться для того, щоб на RP не було дубльованих користувачів.

Нарешті, бачимо, що якщо всі вищезазначені вимоги виконуються, буде повернено об’єкт з усіма необхідними параметрами. Це параметри, які можна використовувати для підтвердження спроби входу, коли це відбувається.

Потім цей об’єкт можна додати до бази даних. Він містить CredentialId, який є унікальним ідентифікатором для цих облікових даних. Відкритий ключ, який використовується для розшифровки підпису закритого ключа. Користувач - це те, як RP підключатиме користувача до облікових даних. Після цього з’являється лічильник, який включений для запобігання атакам повтору, особливо атакам клонування. Потім у нас є CredType, який містить формат атестації, який має автентифікатор. Нарешті, у нас є Aaguid, який вказує марку та модель пристрою автентифікації.

3.3 Інтеграція систем

Хоча сам по собі це не етап проекту, цей розділ описує налаштування середовища для розробки двох вищезгаданих сценаріїв: розробки веб-додатку та інтеграції з адаптивним порталом. Незважаючи на їх незалежний характер, описане тут середовище базується на остаточному сценарії, оскільки воно є більш складним. Останній варіант системи, включає технологію адаптивного порталу. Вибрана технологія OpenNDS, встановлена в мікропрограмі маршрутизатора OpenWRT, яка входить до цієї топології, що показана на малюнку 3.4. Крім того, OpenNDS можна налаштувати за допомогою

віддаленого сервера за допомогою параметра Forwarding Authentication Service (FAS). У цій топології віддалений сервер також відомий як сервер FAS. Підсумовуючи, середовище має реалізовувати спрощену версію остаточного сценарію. Зокрема, з трьома основними компонентами:

- клієнтський пристрій;
- маршрутизатор із технологією адаптивного порталу;
- сервер веб-додатків.

Враховуючи потреби, зазначені вище, середовище базується на віртуальному середовищі. З цією метою було створено кілька віртуальних машин (VM) і мережевих інтерфейсів у VirtualBox, які увімкнули зв’язок між адаптерами, які використовують лише хост. Розроблена топологія розділяє корпоративну мережу на чотири різні локальні мережі, підключені до маршрутизатора OpenWRT

1. Інтернет. Налаштований з NAT у VirtualBox, представляє інтерфейс WAN для Інтернету.
2. Демілітаризована зона (DMZ). Вона також відома як локальна мережа сервера, де розміщуються корпоративні сервери.
3. Менеджмент (МНГ). Ця локальна мережа представляє мережу для цілей адміністрування. У топології віртуальної мережі він представляє головну машину.
4. Мережа клієнтів. Клієнтські віртуальні машини підключаються до цього інтерфейсу. Він представляє клієнтів порталу приєднання.

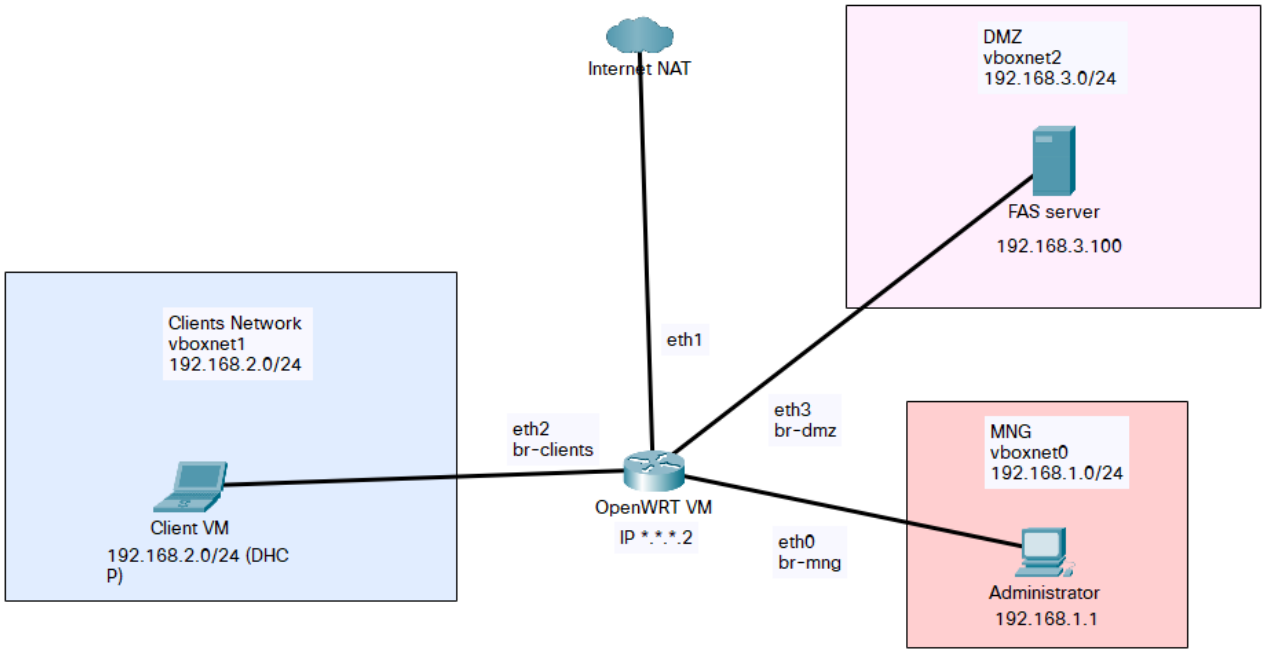

Рис 3.4 Топологія віртуальної мережі

Налаштування в VirtualBox. Маршрутизатор керує чотирма визначеними мережами. Інтерфейси OpenWRT представлені як eth0-3. Логічні мости OpenWRT представлені br-<тегом>.

Маршрутизатор OpenWRT має три інтерфейси з відповідним мостовим пристроєм і дротовим портом. Ці віртуальні порти представляють дротові з'єднання, які забезпечують підключення до мережі VirtualBox.

Всього є три мережі LAN, які будуть налаштовані як мережі VirtualBox лише для хостів. Вони представлені на малюнку 4.7 як vboxnet0-2 разом із відповідною адресою IPv4: 192.168.1-3.0/24. Для кожної з цих мереж VirtualBox хост-машина встановлює один логічний інтерфейс. Відповідна адреса IPv4 для хост-машини завжди буде x.x.x.1. Маршрутизатору OpenWRT буде призначено адресу IPv4 x.x.x.2 у кожній з локальних мереж. Адресація налаштовується вручну, тому DHCP-сервер VirtualBox для цих мереж має бути відключений. Нарешті, кожна з підключених локальних мереж налаштована в OpenWRT як пристрій мосту, вказуючи відповідний дротовий порт, до якого відповідна локальна мережа підключена через адаптер VirtualBox.

OpenNDS — це технологія Captive Portal, яка використовується в сценарії для інтеграції з автентифікацією WebAuthn. Ця технологія також випущена у формі пакета OpenWRT, тому її можна встановити за допомогою opennds для встановлення opkg. Після встановлення файл конфігурації можна знайти в /etc/config/opennds. Для тестування його основної функціональності ми можемо використати конфігурацію за замовчуванням, адаптуючи інтерфейс шлюзу до br-клієнтів, інтерфейс клієнтського мостового пристрою в OpenWRT. Основну перевірену функціональність можна підсумувати в три етапи:

1. Доступ до веб-сторінки HTTP з клієнта, підключеного до мережі.
2. OpenNDS перенаправить запит на сторінку адаптивного порталу за замовчуванням. Ця сторінка просить користувача прийняти Умови надання послуг.
3. Після того, як користувач погоджується, OpenNDS показує інформаційну сторінку та дозволяє переспрямування на оригінальну запитану сторінку.

3.4 Демонстрація роботи програмного модулю

Початковим екраном даного програмного модулю є форма реєстрації, де користувач повинен ввести свої данні. А саме для проходження першого етапу необхідно вказати логін та пароль. В даному випадку під логіном розуміється пошта користувача.

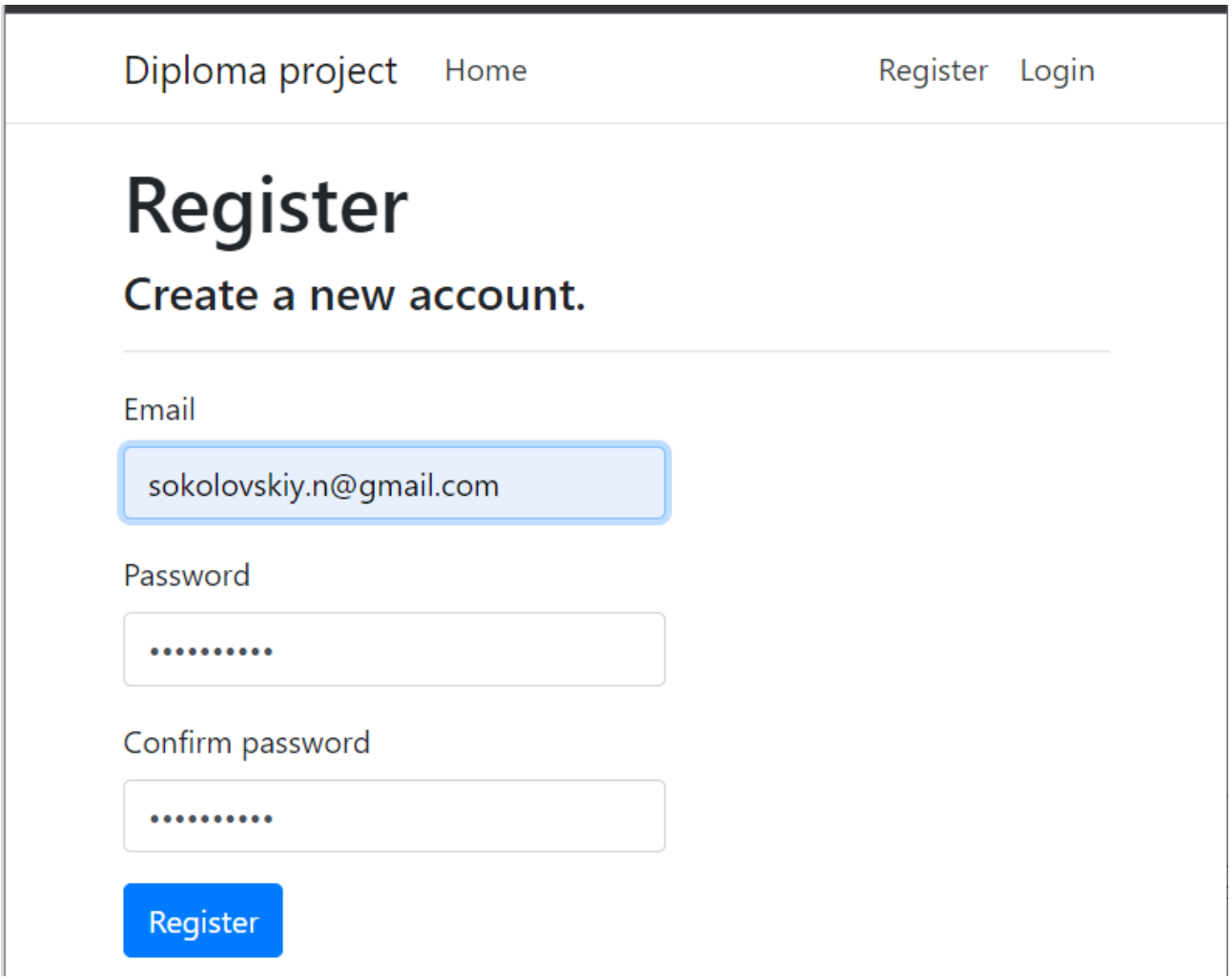

Рис. 3.5 – Форма реєстрації

Другим обов’язковим кроком реєстрації виступає етап додавання користувацького автентифікатора. Натиснувши на кнопку «Додати двофакторну автентифікацію» почнеться пошук всіх доступних зовнішніх пристроїв та вбудованих програмних засобів автентифікації. Відповідно на наступному малюнку показане підключення різних видів автентифікаторів.

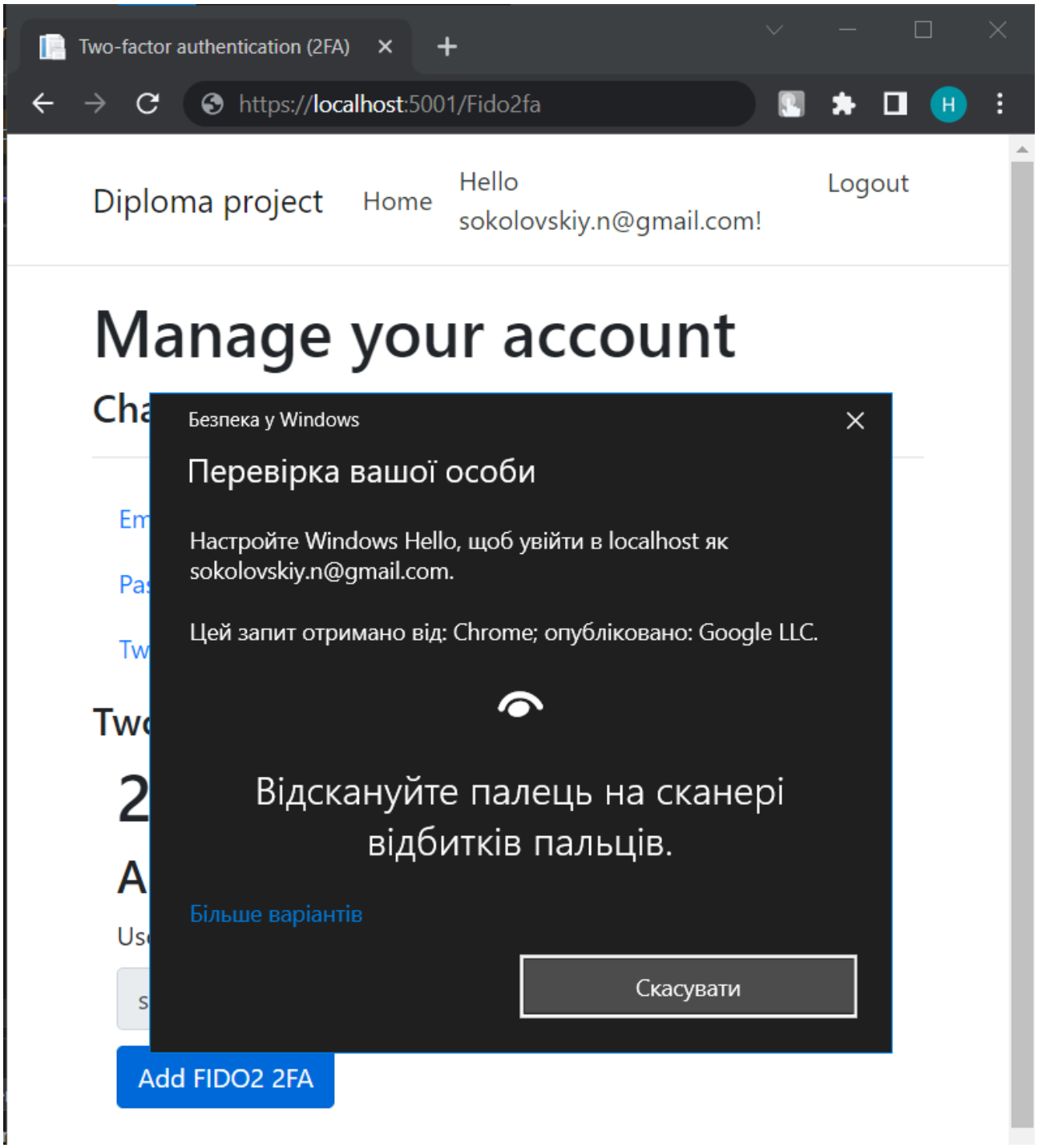

Рис. 3.6 – Використання програмного автентифікатора Windows Hello

Після успішного етапу реєстрації та при зміні аутентифікатора відбувається звернення до бази даних для збереження облікової інформації користувача. В таблицях збережено як інформація

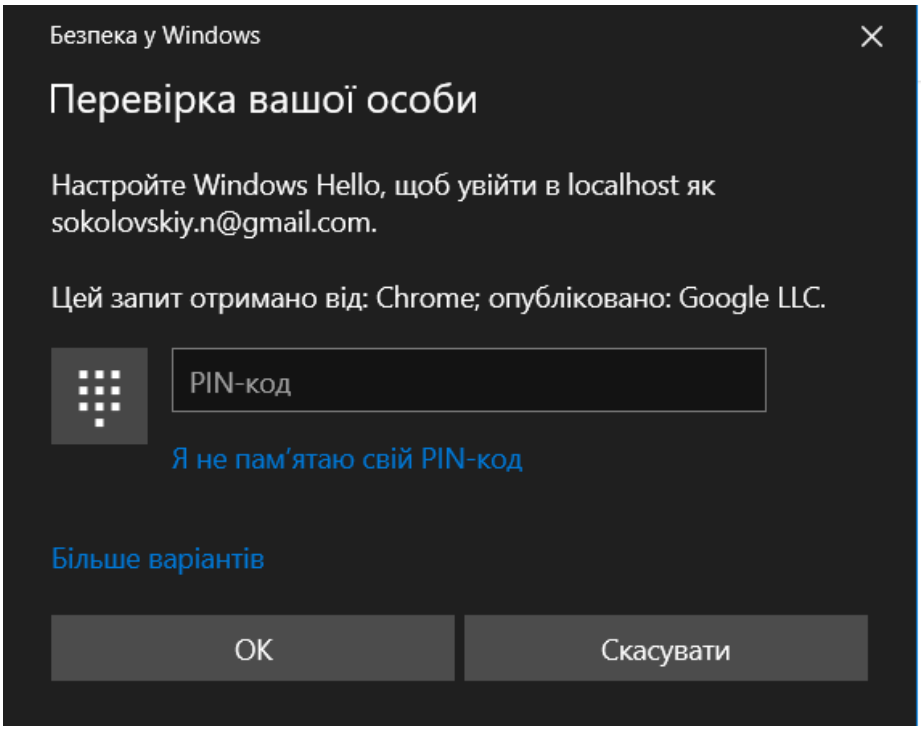

Рис. 3.7 –Програмний аутентифікатор Windows Hello.

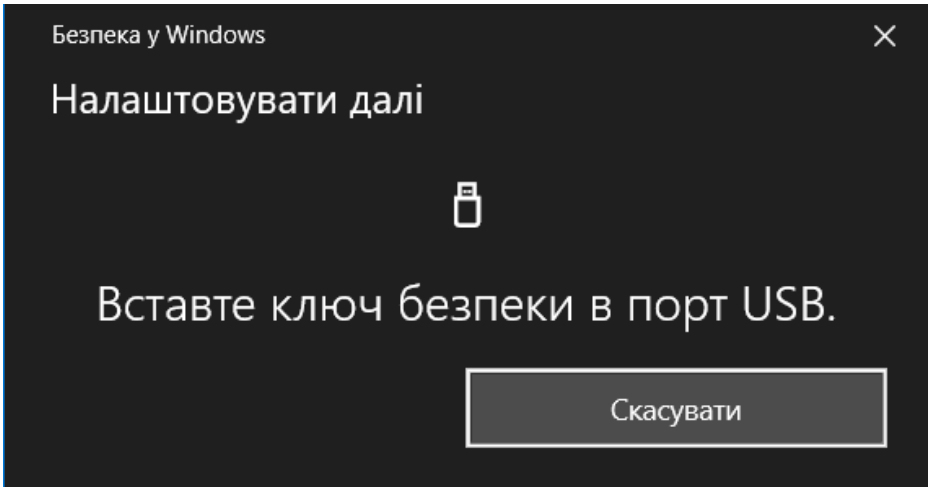

Рис. 3.8 –Додавання USB-ключа, як аутентифікатора.

Після реєстрації можна перевірити робото здатність програмного модулю, вийшовши зі свого аккаунту та перейшовши на сторінку Login. Як і на етапах реєстрації, у користувача з'являється форма входу. Після успішного першого кроку, веб додаток виводить на екран вимогу пройти другий етап автентифікації з використанням зареєстрованого аутентифікатора.

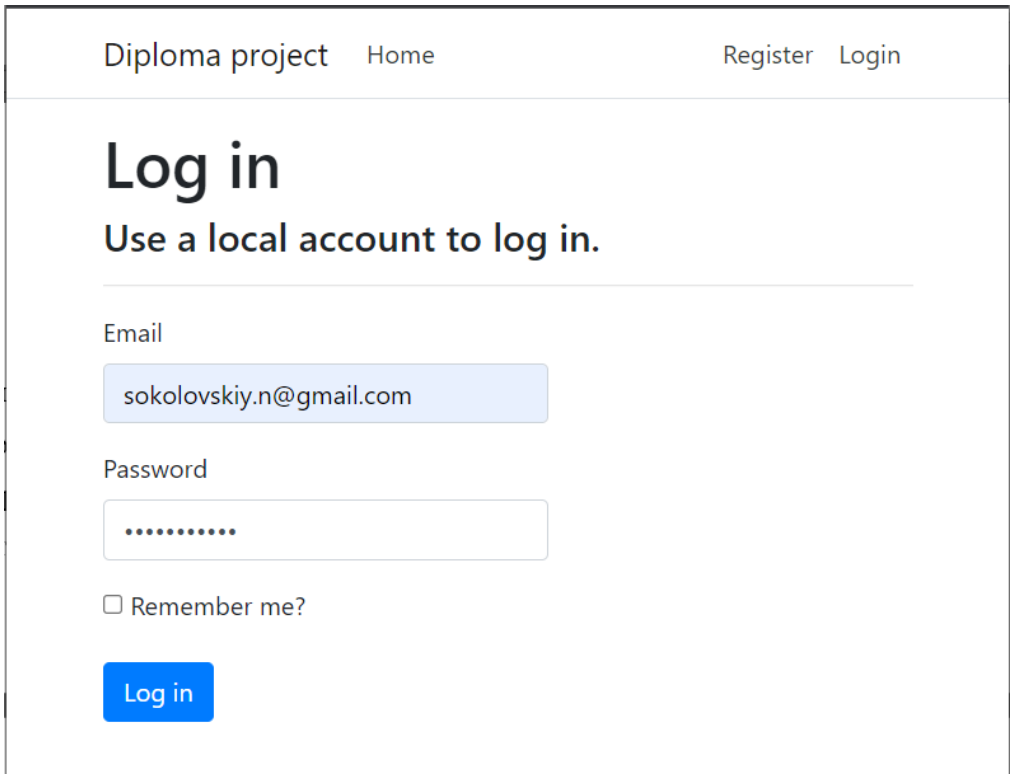

Рис. 3.9 –Форма входу в обліковий запис.

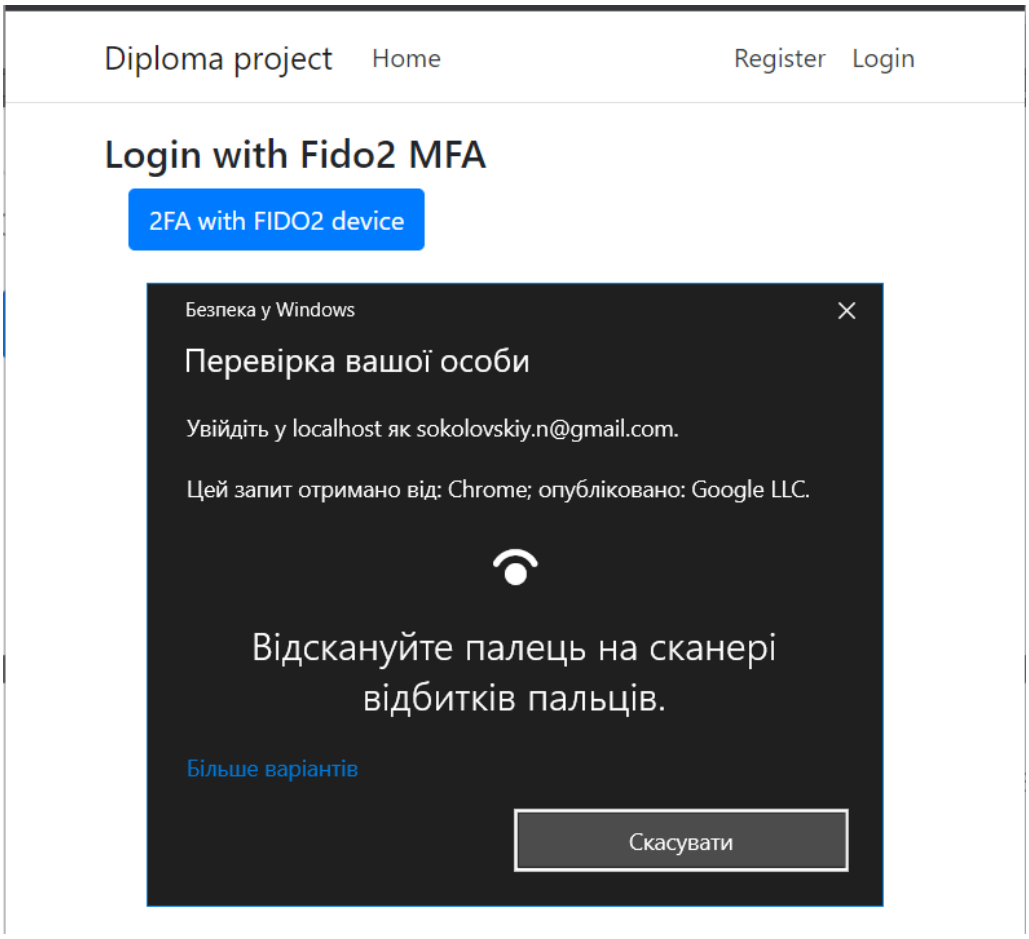

Рис. 3.10 – Вимога використання автентифікатора для входу в обліковий запис.

3.5 Висновок до третього розділу

В третьому розділі була описана архітектура впровадженої системи та опис її елементів. Були продемонстрована робота програмного модулю на основі стандарту FIDO2 з використанням програмного інтерфейсу WebAuthn.

Щоб запобігти атакам відтворення, спрямованим на FIDO2, одним із запобіжних заходів є включення поля «виклик». Крім того, існує лічильник, який збільшується з кожним використанням пристрою автентифікації.

У контексті захисту від фішингових атак або атак типу "людина посередині", спрямованих на FIDO2, спроба отримати інформацію від автентифікатора, змусивши його увійти на оманливий веб-сайт, виявляється нездійсненною. Ця стратегія ефективна лише на початковому етапі автентифікації. Причина цього полягає в зв’язку автентифікатора з сервером через інформацію, що зберігається в базі даних. Отже, коли автентифікатор отримує запит на вхід, він перевіряє ідентифікатор пристрою, порівнюючи його з зареєстрованими. Якщо збігів не знайдено, автентифікацію буде відмовлено.

Іншим засобом протидії фішингу та атакам типу "людина посередині" є використання HTTPS для зв’язку між службою та користувачем. Це забезпечує конфіденційність даних, що передаються між користувачем і сервером, оскільки прослуховування стає недоступним. Навіть якщо зловмиснику вдається отримати передану інформацію від аутентифікатора, розшифрувати критичні дані залишається неможливим.

РОЗДІЛ 4. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

4.1 Екологічні проблеми утилізації побутових відходів.

Після приєднання України до Європейської Ради та реалізації програми адаптації українського законодавства до норм і стандартів Європейського Союзу Україна зобов'язана вирішити проблеми у сфері поводження з відходами. Водночас, аналізуючи актуальні проблеми у сфері поводження з відходами, слід виділити ті, без вирішення яких неможливо забезпечити ефективність національної системи поводження з відходами, до яких належать:

1) юридичні:

- відсутній механізм регулювання відносин і забезпечення виконання зобов'язань у сфері поводження з відходами (транспортування, переробка, утилізація);–система відповідальності споживачів, виробників і компетентних органів є недосконалою та нерегламентованою;
- недосконаліть тендерної процедури у сфері переробки та утилізації відходів;
- питання про екологічну освіту перебуває на критично низькому рівні, а програми щодо її підвищення відсутні;–зв'язок між суб'єктами господарювання та держаним сектором в галузі є мінімальним;
- відсутність контролю державними органами влади над виконанням усіх нормативно-правових актів щодо поводження з відходами;
- відсутність закону про вторинні матеріальні ресурси;
- відсутність реального державного контролю за утворенням відходів і поводженням з ними, внаслідок чого понад 60% використовується в тіньовій економіці і лише 10% контролюється державою;

2) економічні:

- відсутність системи штрафів у сфері поводження з відходами і навпаки системи економічної мотивації за ефективне поводження з відходами;–відсутність реальних інструментів протидії незаконному захороненню відходів;
- відсутність сформованих внутрішніх ринків вторинної сировини (державного та приватного ринків);
- відсутність економічно структурованої, спрощеної та гармонізованої з нормами міжнародного законодавства методики визначення норм утилізації;
- невідповідність виробничих потужностей екологічно безпечному видаленню, утилізації, регенерації відходів;
- ліцензування псевдо-діяльності з переробки нафтопродуктів без належного контролю за її результатами;
- труднощі із залученням іноземних інвестицій, зумовлені великим ризиком інноваційних процесів в Україні та відсутністю детальної діяльності органів державної виконавчої влади щодо системи поводження з відходами;

3) соціальні:

- нерозвиненість системи екологічної освіти, екологічної свідомості виробників і споживачів;–низький рівень екологічної культури та екологічної свідомості громадян;
- небажання в кожному домогосподарстві докладати праці та зусиль для сортування сміття;
- технічні:
- невирішені питання логістики;
- невирішені питання взаємозв’язків між стейкхолдерами та процеси поводження один між одним.

Крім того, на сьогодні чинна нормативно-правова баз у сфері поводження з відходами не відповідає Директивам ЄС про відходи, у тому числі й ті, які Україна зобов'язана імплементувати згідно з Угодою про асоціацію. Основні принципи ЄС щодо управління відходами в Україні ще не виконано. Основна лексика, яка використовується у відповідних актах України, не відповідає визначенням у нормативних документах ЄС щодо відходів. Проблеми в правовому регулюванні, зокрема, на етапі правозастосування та реалізації у сфері поводження з відходами, у затримці запровадження належних стандартів ЄС призводять до правової невизначеності, що дискредитує іноземних інвесторів. Тому варто зауважити, що запропонований економіко-правовий механізм управління системою поводження з відходами має бути орієнтований на вирішення зазначених проблем.

Узагальнюючи дослідження, економіко-правовий механізм управління сферою поводження з відходами пропонується визначати як систему взаємо-відносин між зацікавленими сторонами, засновану на принципах, елементах, економічних і правових засобах управлінського впливу, що використовуються для регулювання екологічних та економічних правовідносин для досягнення завдань і цілей екологічної політики держави. Слід зазначити, що механізм тісно пов'язаний і залежить від економічної, політичної, правової та культурної системи суспільства, специфіки організації та функціонування державного механізму, а також від еколого-правової культури суб'єктів господарювання.

Аналізуючи ж сучасний стан сектору переробки відходів в Україні, можна побачити значну невідповідність кількості відходів, що генеруються щорічно, з тією кількістю, що здатна пройти належну переробку в спеціально відведе-них для цього місцях. Усередньому Україна щороку виробляє близько 17 млн т побутового і промислового сміття. Проте відсоток, який відповідає кількості сміття, яке потім потрапляє на переробку, становить не більше 5%. На преве-ликий жаль, в Україні більшість сміття в кінцевому результатіпотрапляє на звалища, розмір яких вражає та лише продовжує збільшуватися.

У середньому українець викидає від 200 до 300 кг побутового сміття на рік. Проте майже усе це сміття просто прямує на звалища і в кращому разі спалюється. Відсоток переробки сміття в Україні порівняно з європейськими країнами просто мізерний. Адже Швеція, Німеччина, Австрія, Швейцарія та багато інших переробляють від 90 до 99% власних відходів. А деяким країнам навіть не вистачає власного сміття, що зумовлює його експорт. У той час як в Україні спостерігається катастрофічна відсутність необхідних потужностей для роботи зі своїм сміттям. Кількість сучасних сміттєпереробних заводів, які відповідають усім екологічним вимогам і нормам, в Україні незначна. Саме тому тих наявних потужностей недостатньо для того, щоб справитися з наявною екологічною проблемою в країні, не говорячи вже про ефективну переробку цих відходів.

Говорячи ж про розвиток інфраструктури, яка повинна забезпечувати процеси належної переробки і сортування відходів, то ситуація виглядає дуже критичною, адже в Україні існує всього-на-всього 4 сміттєспалювальних заводи, з яких працює лише один. Більше того, у XXI ст.в Україні взагалі відсутні заводи із сортування та комплексної переробки побутових відходів, що неприпустимо для європейських амбіцій держави та комфортного існування в майбутньому загалом.

Усі наведені факти свідчать про недосконаліть сфери переробки відходів в Україні, а також про відсутність необхідних механізмів і регуляторів задля створення єдиної, ефективної та взаємопов’язаної системи поводження з відходами для громадян, суб’єктів господарювання та держави.

ВИСНОВКИ

Внаслідок завершення дипломної роботи було розроблено передовий двоетапний механізм аутентифікації, використовуючи інтерфейс програмного забезпечення WebAuthn та відповідаючи стандартам FIDO, з урахуванням оптимальних практик для забезпечення максимальної безпеки системи.

У першому розділі були розглянуті основні аспекти забезпечення безпеки вебзастосунку та роль аутентифікації. Аналізуються методи аутентифікації, їх переваги та недоліки. Детально розглядаються основні загрози та атаки, які можуть надати злочинцям несанкційований доступ до облікового запису користувача.

Другим етапом був аналіз сучасних технологій для впровадження системи аутентифікації. Здійснювався аналіз програмного забезпечення, сумісного з цими технологіями. Розглядалися найкращі практики для впровадження протоколу WebAuthn для забезпечення максимального рівня безпеки системи. Також була розроблена архітектура майбутньої системи аутентифікації.

В третьому розділі булу розглянуті основні засоби та інструменти для реалізації системи. Також, детально описано побудову фрейворку, основних алгоритмів та інтеграцій в системі. Вкінці роботи була показана робоча версія створеного програмного модулю та викладений основний порядок роботи з ним.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Конфіденційність, цілісність та доступність [Електронний ресурс]. – Режим доступу: https://www.techtarget.com/whatis/definition/Confidentiality-integrity-and-availability-CIA
2. Різниця ідентифікації, аутентифікації та авторизації [Електронний ресурс]. – Режим доступу: https://imageware.io/identification-authentication-authorization-difference/
3. Біометрія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%91%D1%96%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D1%96%D1%8F
4. Time-Based One-Time Password [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc6238
5. The OAuth 2.0 Authorization Framework [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc6749
6. Що таке атака повторного відтворення [Електронний ресурс]. – Режим доступу: https://nordvpn.com/blog/replay-attack/
7. Атака «Людина посередині» [Електронний ресурс]. – Режим доступу: https://www.imperva.com/learn/application-security/man-in-the-middle-attack-mitm/
8. Огляд універсального другого фактору(U2F) [Електронний ресурс]. – Режим доступу: https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.html
9. Огляд Архітектури FIDO UAF [Електронний ресурс]. – Режим доступу: https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-uaf-overview-v1.1-id-20170202.html
10. API керування обліковими даними [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/Credential_Management_ API
11. Web Authentication: An API for accessing Public Key Credentials [Електронний ресурс]. – Режим доступу: https://www.w3.org/TR/webauthn/

12. Can I use webauthn? [Електронний ресурс]. – Режим доступу: https://caniuse.com/?search=webauthn
13. Альянс FIDO [Електронний ресурс]. – Режим доступу: https://fidoalliance.org/overview/
14. Постачальники ключів безпеки FIDO2 [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/azure/active-directory/ authentication/ concept-authentication-passwordless
15. Web Authentication API [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US /docs/Web/API/Web_Authentication_API
16. Web framework benchmarks [Електронний ресурс]. – Режим доступу: https://www.techempower.com/benchmarks/#section=data-r20&hw=ph& test=composite.
17. Увімкнення Captive Portal у бездротовій мережі Cisco [Електронний ресурс]. – https://www.cisco.com/c/en/us/support/docs/smb/wireless/cisco-small-business-300-series-wireless-access-points/smb4937-enable-a-captive-portal-on-your-cisco-wireless-network.html
18. CBOR — Стисле двійкове представлення об’єктів [Електронний ресурс]. – https://cbor.io/
19. Важливість стандартів W3C [Електронний ресурс]. – https://www.bopdesign .com/bop-blog/2013/06/the-importance-of-w3c-standards/
20. Інтеграція протоколів FIDO [Електронний ресурс]. – https://fidoalliance.org /integrating-fido-and-federation-protocols/
21. Керівництво з веб-автентифікації [Електронний ресурс]. – https://webauthn.guide/
22. Посібник OpenWrt на VirtualBox [Електронний ресурс]. – https://openwrt.org/docs/guide-user/virtualization/virtualbox-vm
23. OpenNDS. Служба автентифікації пересилання (FAS) [Електронний ресурс]. – https://opennds.readthedocs.io/en/stable/fas.html

24. Атестація та затвердження [Електронний ресурс]. – https://developer.mozilla.org/en-US/docs/Web/API/Web_Authentication_API/Attestation_and_Assertion
25. Yubiko. Резидентні ключі. [Електронний ресурс]. – https://developers.yubico.com/WebAuthn/WebAuthn_Developer_Guide/Resident_Keys.html

Додаток А

Лістинг коду моделі даних, які необхідні для аутентифікації

```
public class FidoStoredCredential
{
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public virtual int Id { get; set; }
    public virtual string UserName { get; set; }
    public virtual byte[] UserId { get; set; }
    public virtual byte[] PublicKey { get; set; }
    public virtual byte[] UserHandle { get; set; }
    public virtual uint SignatureCounter { get; set; }
    public virtual string CredType { get; set; }
    public virtual DateTime RegDate { get; set; }
    public virtual Guid AaGuid { get; set; }
    [NotMapped]
    public PublicKeyCredentialDescriptor Descriptor
    {
        get { return string.IsNullOrWhiteSpace(DescriptorJson) ? null
                : JsonConvert.DeserializeObject<PublicKeyCredentialDescriptor>(DescriptorJson); }
        set { DescriptorJson = JsonConvert.SerializeObject(value); }
    }
    public virtual string DescriptorJson { get; set; }
}
```

Головна функція для забезпечення двофакторної атунтифікації FIDO2

```
[HttpPost]
[ValidateAntiForgeryToken]
[Route("/mfamakeAssertion")]
public async Task<JsonResult> MakeAssertion([FromBody] AuthenticatorAssertionRawResponse clientResponse)
{
    var jsonOptions = HttpContext.Session.GetString("fido2.assertionOptions");
    var options = AssertionOptions.FromJson(jsonOptions);
    var creds = await _fido2Storage.GetCredentialByIdAsync(clientResponse.Id);
    if (creds == null)
    {
        throw new Exception("Unknown credentials");
    }
    var storedCounter = creds.SignatureCounter;
    IsUserHandleOwnerOfCredentialIdAsync callback = async (args) =>
    {
        var storedCreds = await _fido2Storage.GetCredentialsByUserHandleAsync(args.UserHandle);
        return storedCreds.Exists(c => c.Descriptor.Id.SequenceEqual(args.CredentialId));
    };
    var res = await _lib.MakeAssertionAsync(clientResponse, options, creds.PublicKey, storedCounter, callback);
    await _fido2Storage.UpdateCounterAsync(res.CredentialId, res.Counter);
    var user = await _signInManager.GetTwoFactorAuthenticationUserAsync();
    if (user == null)
    {
        throw new InvalidOperationException($"Unable to load two-factor authentication user.");
    }

    var result = await _signInManager.TwoFactorSignInAsync("FIDO2", string.Empty, false, false);
    return Json(res);
}
```