Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка та дослідження методів аутентифікації користувачів в інформаційних системах
(тема)

Виконав:
студент 2 курсу, групи СШІм-21-2
Кутовий Д. О.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник проф. Філатов В.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Кутовому Данилу Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів аутентифікації користувачів в інформаційних системах

затверджена наказом університету від 31 березня 20 23 р. № 306Ст
2. Термін подання студентом роботи до екзаменаційної комісії 19 травня 20 23 р.
3. Вихідні дані до роботи

4. Перелік питань, що потрібно опрацювати в роботі
1) Огляд та аналіз сучасного стану проблеми
2) Постановка задачі
3) Архітектура проекту
4) Результати розробленої програми

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	14.04.2023 - 16.04.2023	виконано
2	Вимоги до системи	19.04.2023	виконано
3	Дослідження методів авторизації та аутентифікації	20.04.2023 - 25.04.2023	виконано
4	Розробка прототипів та підготовка до реалізації	26.04.2023	виконано
5	Реалізація проекту	26.04.2023 - 14.05.2023	виконано
6	Підведення підсумків проекту	14.05.2023 - 16.05.2023	виконано
7	Оформлення пояснювальної записки	28.04.2023 - 16.05.2023	виконано
8	Оформлення презентації	17.05.2023	виконано
9	Попередній захист	18.05.2023	виконано
10	Захист перед ЕК	19.05.2023	виконано

Дата видачі завдання 3 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Філатов В.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 73 с., 26 рис., 1 дод., 22 джерела.

АУТЕНТИФІКАЦІЯ, ПАРОЛЬ, 2FA, DOD, HMAC, IT, JS, LARAVEL, MFA, NIST, OTP, PHP, PIN, REACT, SHA, SIM, SMS, TOTP

Об’єктом дослідження є методи аутентифікації користувачів в інформаційних системах.

Предмет дослідження є ефективність та надійність методів аутентифікації користувачів в інформаційних системах, зокрема традиційних методів (наприклад, введення логіну та пароля) та біометричних методів (використання фізіологічних та/або поведінкових характеристик користувача).

Метою дослідження є визначення ефективності та надійності різних методів аутентифікації користувачів в інформаційних системах з метою поліпшення захисту конфіденційної інформації та забезпечення безпеки користувачів.

Методи дослідження:

– аналіз наукової літератури з питань аутентифікації користувачів в інформаційних системах;

– експериментальні дослідження ефективності різних методів аутентифікації користувачів в інформаційних системах, включаючи тестування на реальних користувачах та оцінку швидкості та точності роботи;

– застосування методів статистичного аналізу та інші методи обробки даних для оцінки результатів дослідження;

– вивчення сучасних тенденцій та розробка нових методів аутентифікації користувачів в інформаційних системах.

ABSTRACT

Explanatory note: 73 p., 26 fig., 1 ann., 22 sources.

AUTHENTICATION, 2FA, DOD, HMAC, IT, JS, LARAVEL, MFA, NIST, OTP, PASSWORD, PHP, PIN, REACT, SHA, SIM, SMS, TOTP

The object of research is user authentication methods in information systems.

The subject of the study is the effectiveness and reliability of user authentication methods in information systems, in particular, traditional methods (for example, entering a login and password) and biometric methods (using the user's physiological and/or behavioral characteristics).

The purpose of the study is to determine the effectiveness and reliability of various user authentication methods in information systems in order to improve the protection of confidential information and ensure the safety of users.

Research methods:

– analysis of scientific literature on user authentication in information systems;

– experimental studies of the effectiveness of various user authentication methods in information systems, including testing on real users and evaluating the speed and accuracy of work;

– application of statistical analysis methods and other methods of data processing to evaluate research results;

– study of modern trends and development of new user authentication methods in information systems.

ЗМІСТ

ВСТУП

Наразі паролі є стандартним методом аутентифікації користувачів. Оскільки апаратне забезпечення продовжує розвиватися, зламати ці паролі стає легше. Традиційним рішенням цієї проблеми є постійно зростаюча складність пароля та двофакторна аутентифікація. Однак користувачі стають напруженими через надто складні системи входу і часто обходять їх. Двофакторна аутентифікація також додає цю складність, і багато форм двофакторної аутентифікації за своєю суттю є небезпечними. У відповідь на ці проблеми наша кваліфікаційна робота пропонує безпарольну багатофакторну систему аутентифікації, яка використовує випробувані й перевірені існуючі технології, асиметричну криптографію, цифрові підписи та біометричну аутентифікацію.

Змодельоване тестування користувача показує багатообіцяючі результати, припускаючи, що реєстрацію можна завершити трохи більше ніж за тридцять секунд, а аутентифікацію трохи більше ніж за дві секунди. Також обговорюється аналіз можливих векторів атак нашої кваліфікаційної роботи, вжиті профілактичні кроки та їх вирішення в потенційних майбутніх дослідженнях.

1 ОГЛЯД ТА АНАЛІЗ СУЧАСНОГО СТАНУ ПРОБЛЕМИ

1.1 Постановка проблеми

Кібербезпека – це постійна боротьба між тими, хто володіє інформацією, яку варто захистити, і тими, хто хоче отримати цю інформацію. З розвитком технологій безпека зростає в експоненціальній швидкості. Ранні комп’ютери спричинили зміну дизайну безпеки: від незначного захисту шляхом обмеження фізичного доступу до безпеки для кількох користувачів і багаторівневого застосування. У епоху Інтернету будь-який підключений пристрій є потенційно вразливим. Один із методів захистити вразливості входу – це паролі.

У сфері технологій паролі реалізуються комп’ютерами, причому комп’ютер пропонує завдання, на яке потрібно правильно відповісти в текстовому форматі. Комп’ютер може лише проаналізувати введений текст і порівняти його з точною правильною відповіддю, тобто правильну текстову відповідь на виклик користувач має точно запам’ятати. Оскільки технологія продовжує розвиватися, паролі можна вгадувати, розшифровувати або використовувати грубим методом, що призводить до підвищення вимог до паролів, включаючи довжину, використання символів і відмінності в регістрі літер. Протягом кількох десятиліть політика Міністерства оборони (DoD) і Національного інституту стандартів і технологій (NIST) стверджувала, що безпечний пароль вимагає [1], [2]:

- щонайменше 9 символів;
- принаймні один спеціальний символ;
- принаймні одне число;
- принаймні одна велика літера;
- принаймні одна мала літера;
- містить принаймні 4 символи, ніж останній пароль;

івіііііііівофівоіфралвріваролвіарівлоарівоалрівоаріварівлоарівлоарівло

– змінювати кожні 90-150 днів.

Труднощі із запам'ятовуванням довгих рядків випадкового тексту не є новою проблемою. У психологічному дослідженні 1956 року Міллер [3] виявив, що середня людина може легко запам’ятати лише близько семи цифр або символів, плюс-мінус два. Переходячи до семінару безпеки USENIX, Клейн [4] стверджує, що користувачі, як правило, вибирають невеликий набір символів і цифр, які їм найбільше запам’ятовуються, як паролі. Ці паролі, як правило, менш безпечні, оскільки вони не є випадковими, і їх можна виявити та використати шаблони. У статті під назвою «Якщо ваш пароль 123456, просто зробіть його зламаним», Ванс [5] обговорює постійно зростаючий пул поширених розкритих паролів користувачів.

Чесвік [6] у дослідженні 2013 року виявив, що необхідність частої зміни пароля змушує користувачів вибирати менш безпечні паролі, які краще запам’ятовуються, просто змінюючи кілька символів, щоб пароль відповідав мінімальним вимогам. Адамс та інші [7] додають, що ці проблеми з паролями ускладняються тим фактом, що багато користувачів часто використовують кілька систем, які вимагають аутентифікації, багато з яких вимагають окремого набору облікових даних для аутентифікації. У відповідь на ці проблеми було запропоновано багато методів, однією з таких пропозицій є багатофакторна аутентифікація (MFA).

Двофакторна аутентифікація (2FA), підмножина MFA, була запропонована як можливе рішення для боротьби зі слабкими паролями користувачів, але вона породжує власний набір проблем. Джовер [8] виявив, що одноразові паролі (OTP) служби коротких повідомлень (SMS) вразливі до викрадення під час передачі незахищеними лініями стільникового зв’язку та заміною SIM-карт. Коган та інші [9] у своїй пропозиції щодо T/Key обговорюють, як одноразовий пароль на основі часу (TOTP)

вимагає, щоб секретне початкове число зберігалося у відкритому вигляді на сервері та може бути відкритим у разі атаки на всьому сервері.

Дрю [10] повідомляє про успішний злам мережі Lockheed Martin, підрядника національної оборони, після того, як секретні коди Lockheed Martin були викрадені з їхніх серверів.

Апаратні токени універсальної послідовної шини (USB), такі як YubiKey [11], не тільки вимагають необхідності купувати та переносити додаткове нестандартне обладнання, але також є вразливими до атаки із запитом на повторне відтворення. Нарешті, додавання ще одного рівня складності до вже складного рішення безпеки не гарантує, що користувачі дотримуватимуться додаткових заходів безпеки 2FA.

За словами Чесвіка [6], уявлення користувачів про потребу в безпеці значною мірою впливають на їхній вибір щодо безпеки. Адамс та інші [7] припускає, що користувачі, як правило, не повністю усвідомлюють ризики впливу безпеки та вилучення інформації, і в деяких випадках користувачі свідомо ухиляються від політики безпеки через те, що вони вважають непотрібною системою. Сассе та інші [13] пояснюють, що якщо користувачі вважають політику безпеки або пароль надто складною або не повністю розуміють політику, вони, швидше за все, обійдуть її. Ці проблеми проглядаються у зростаючій загрозі фішингу, браузерного підроблення запитів (CSRF), міжсайтового сценарію (XSS) і атак Man-In-The-Middle (MITM), що залишає користувача в замішанні як найкраще захистити себе. Система, яка забезпечує стандартний рівень безпеки та є простою у використанні та розумінні, може сприяти більшій сумісності користувачів.

1.2 Аналіз предметної галузі

1.2.1 Аутентифікація

Аутентифікація визначається як спосіб довести з достатньою впевненістю, що ви є тією особою, за яку себе видаєте. За межами сфери технологій це зазвичай робиться за допомогою ідентифікаційної картки, виданої державним органом. У штаті Вашингтон, наприклад, посвідчення водія, видане штатом. Департамент ліцензування видає ліцензії за умови, що вони можуть підтвердити вашу особу за переліком затверджених ідентифікаційних документів [14]. У сфері технологій аутентифікація може бути виконана кількома способами, зокрема: паролями, цифровими підписами та сертифікатами, а також біометрично. Кожен із цих методів має свої переваги та недоліки.

1.2.1.1 Паролі

Пароль – це форма симетричної криптографії, у якій дві сторони узгоджують секретне слово чи фразу (її також називають ключем). Якщо припустити повну таємницю ключа між двома сторонами, коли особистість піддається сумніву, просте знання цього ключа забезпечить розумну впевненість в ідентифікації. Процес обміну паролями можна коротко описати таким чином. Коли користувач намагається ввійти в службу, служба намагається отримати гарантії щодо особи користувача. Сервіс надсилає користувачеві запит на його пароль. Коли користувач вводить свій пароль у відповідь на запит виклику, служба перевіряє, чи збігається отриманий пароль зі збереженим паролем користувача. Оскільки передбачається, що пароль відомий лише користувачеві та службі, пароль забезпечує достатню впевненість щодо особи користувача, і служба може аутентифікувати користувача.

Оскільки служба зазвичай ініціює виклик, їй потрібно зберегти пароль, щоб порівняти його з отриманим введенням. Оскільки паролі можуть бути розкриті або викрадені під час атаки, їх потрібно зберігати в таємниці. Стівен та інші [16] «Open Web Application Security Project»

цитують найкращі практики щодо зберігання паролів. Рекомендації вимагають зберігання не самого фактичного пароля, а хешованої версії з використанням одностороннього криптографічного алгоритму хешування разом із випадковим рядком, який називається сіль.

1.2.1.2 Криптографічне хешування

Функція хешування – це спосіб перетворення даних у вихідні дані фіксованої довжини, які називаються дайджестами повідомлень, які суттєво відрізняються від вхідних даних, так що повторні хеші того самого вхідного елемента завжди створюватимуть той самий дайджест, але навіть невеликі зміни в введення призведе до суттєво відмінного дайджесту. Хешування забезпечує цілісність, що дані не були змінені з моменту їх надсилання. Це хешування відмови є важливим під час перевірки підписів, інакше інформація може бути підписана та змінена кимось іншим, ніж оригінальним підписувачем.

Оскільки той самий вхід завжди створюватиме той самий дайджест, сторона може надсилати дані разом із хешем даних. Сторона-одержувач може повторно обчислити хеш отриманих даних і порівняти його з надісланим хешем.

Ілюстрування хеш-функції, застосовану до кількох прикладів слів, де навіть зміна однієї літери створює значно інший дайджест зображена на рисунку 1.1.

Алгоритми хешування відрізняються криптографічною міцністю, розміром блоку та довжиною результуючого дайджесту. Хоча на рисунку не вказано використовуваний алгоритм хешування, важливо зазначити, що довжина кожного дайджесту не змінюється. При застосуванні до того самого алгоритму хешування, незалежно від вхідних даних, довжина дайджесту завжди залишатиметься постійною.

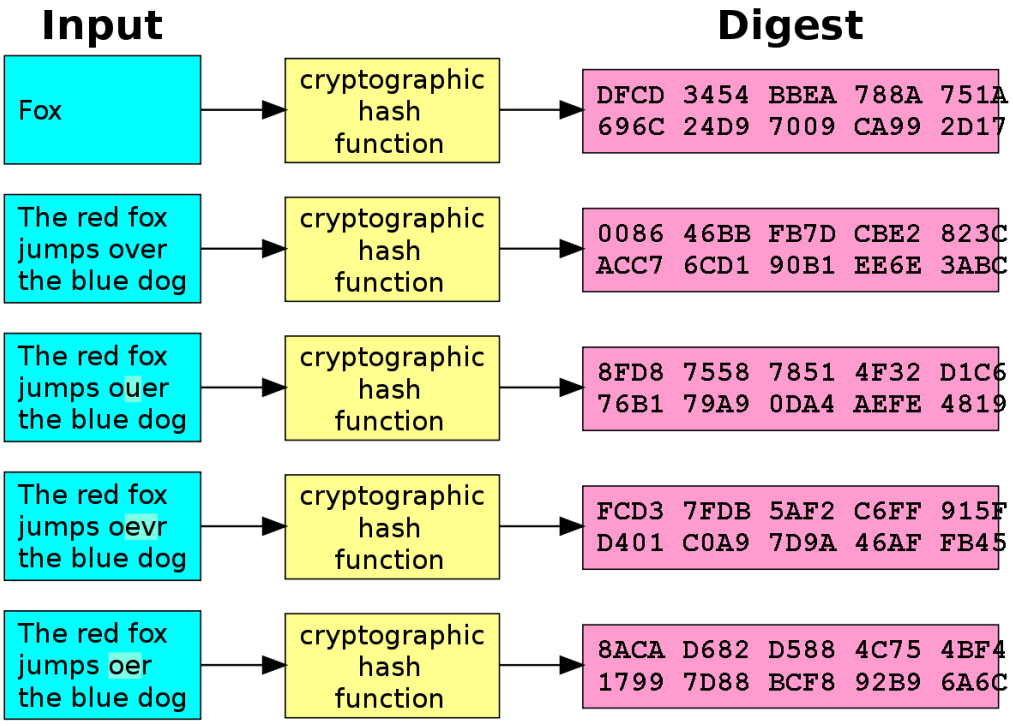


Рисунок 1.1 – Приклад хешування даних

Хоча існує кілька варіантів безпечного хеш-алгоритму (SHA), два часто використовувані варіанти SHA-2, SHA-256 і SHA-512, відрізняються розміром блоку та довжиною дайджесту. SHA-256 має 512-бітні блоки з 256-бітною довжиною дайджесту, тоді як SHA512 має 1024-бітні блоки та довжину дайджесту 512 біт. Хеш-функції схильні до явища, яке називається зіткненням, коли теоретично можливо, що два різні вхідні дані виводять однаковий точний дайджест повідомлення. Андрес [18] стверджує, що більші розміри блоків і довші дайджест-повідомлення забезпечують підвищений захист від зіткнень.

1.2.1.3 Асиметрична криптографія

На відміну від паролів, які є формою симетричної криптографії та покладаються на те, що обидві сторони мають спільний секрет, асиметрична криптографія покладається на те, що кожна сторона має закритий секретний

ключ, який не є спільним. Натомість відкритий ключ створюється математично із секретного ключа, так що закритий ключ неможливо виявити із зашифрованих даних або відкритого ключа без необґрунтованих витрат часу та зусиль. Це ще відомо як пара ключів.

Відкритий ключ можна використовувати для шифрування даних і відкритий, тоді як закритий ключ залишається секретним і є єдиним засобом для розшифровки інформації, зашифрованої за допомогою відповідного відкритого ключа. Хоча асиметричну криптографію можна використовувати для шифрування та дешифрування даних, її також можна використовувати для підпису та перевірки даних за допомогою цифрових підписів. Цю схему зображено на рисунку 1.2.

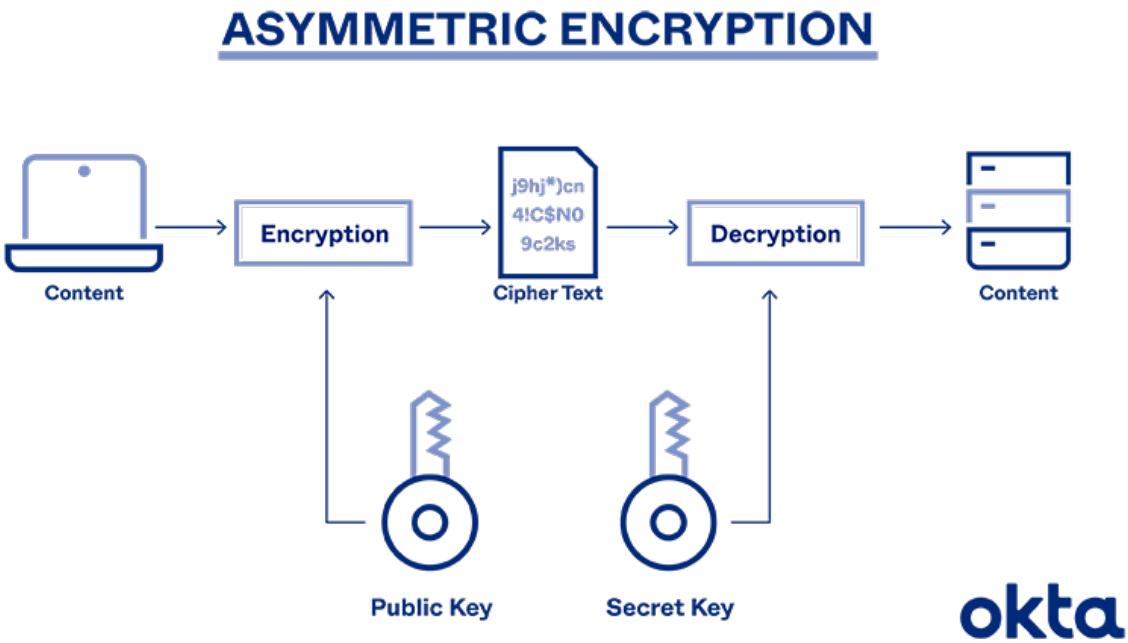


Рисунок 1.2 – Шифрування з асиметричним ключем

1.2.1.4 Сертифікати та підписи

Асиметрична криптографія також може бути використана для

перевірки відправника інформації та того, що дані не були підроблені. Замість використання відкритого ключа для шифрування дані спочатку хешуються, а потім шифруються за допомогою закритого ключа. Це гарантує, що будь-хто, хто має відкритий ключ, може не тільки розшифрувати дані, але також може повторно обчислити дайджест зашифрованих даних, перевіряючи, що дані не були підроблені та були надіслані власником закритого ключа. Припускаючи досконалу секретність закритого ключа, можна з розумною впевненістю визначити особу відправника. Це відомо як цифровий підпис.

Ключі можуть міститися всередині документа, який називається цифровим сертифікатом. Цифровий сертифікат містить інформацію про ключ і термін його дії. Найчастіше ці сертифікати використовуються на захищених веб-сайтах (наприклад, на банківських сайтах). Сертифікат для сайту містить його відкритий ключ, а сертифікат підписується його закритим ключем. Кожен, хто відвідує сайт, може перевірити справжність сертифіката за допомогою відкритого ключа, який міститься в сертифікаті.

1.2.1.5 Біометрія

Біометрична аутентифікація може включати будь-який біологічний атрибут, який унікально ідентифікує користувача. Найпоширенішим використанням цього є відбитки пальців, які поліцейські організації використовують десятиліттями. Інші приклади можуть включати сканування сітківки ока, атрибути обличчя, ідентифікацію голосу та серцевий ритм. Ця інформація повинна збиратися та зберігатися під час реєстрації користувача. Аутентифікацію можна виконати шляхом порівняння наданих біометричних даних користувача з попередньо збереженою інформацією про цього користувача. Всі біометричні системи потребують апаратного забезпечення для захоплення інформації користувача, але можуть забезпечити дуже високий рівень гарантії

ідентифікації, оскільки механізм аутентифікації базується на унікальній інформації користувача, яку важко скопіювати або імітувати.

1.2.2 Багатофакторна аутентифікація (MFA)

MFA – це модель безпеки за допомогою резервування, де для аутентифікації потрібні кілька елементів ідентифікації або фаз. Процес вимагає проходження всіх частин ідентифікації або всіх фаз, інакше аутентифікація не вдається. Джовер [8] наводить звичайний повсякденний приклад банкомату, де користувач має свою банківську картку як перший засіб аутентифікації, а персональний ідентифікаційний номер (PIN), який знає лише користувач, як другий засіб аутентифікації. Без обох цих аутентифікаторів використання банкомату обмежено. Він також включає найпоширенішу реалізацію MFA, 2FA, яка вимагає рівно двох аутентифікаторів.

Процес 2FA зазвичай поєднує пароль з іншою формою ідентифікації, як PIN-код, а саме – одноразовий пароль, надісланий через SMS, електронну пошту, пристрій або мобільний додаток, сповіщення про схвалення, надіслане на надійний пристрій, або USB-токен. Джейком [12] стверджує, що додаткові кроки аутентифікації можуть запобігти зловмиснику отримати доступ до скомпрометованого облікового запису, коли зловмисник уже знає пароль користувача або володіє іншим аутентифікатором.

1.2.2.1 Служба коротких повідомлень

Процес SMS MFA відрізняється залежно від послуги, яка надається, але зазвичай процес полягає в тому, що користувач спочатку успішно входить до служби за допомогою свого імені користувача та пароля, потім служба генерує та надсилає користувачеві на мобільний телефон SMS-повідомлення, що містить одноразовий пароль. Після цього користувач

вводить одноразовий пароль у повідомленні назад у службу. Сервіс порівнює отриманий одноразовий пароль з надісланим. Якщо вони однакові, то другий процес завершується успішно, користувач успішно аутентифікований і отримує доступ до служби. Цей процес також можна виконати електронною поштою, і підхід дуже схожий, за винятком того, що одноразовий пароль надсилається на адресу електронної пошти користувачів, а не на мобільний телефон користувача.

Основна перевага одноразового паролю над підходом SMS або електронною поштою полягає в його простоті налаштування та адміністрування з невеликими витратами. Оскільки багато сервісів використовують цей підхід, він також знайомий більшості користувачів.

Джовер [8] зазначає, що основним недоліком підходу SMS MFA є те, що він вимагає активного підключення до стільникової мережі та вважається загалом небезпечним через свою чутливість до багатьох векторів атак. У той час як електронна пошта потенційно може бути ціллю більшого ризику, оскільки зламаний обліковий запис може розкрити всю онлайн-ідентичність користувача, у дослідженні 2019 року Міріан та інші [20] виявили, що хакерам важко, якщо не неможливо, зламати облікові записи електронної пошти, захищені 2-етапною 2FA від Google, без соціальної інженерії.

В деяких випадках одноразовий пароль, надісланий електронною поштою, наприклад облікові записи, захищені 2-етапною двофакторною перевіркою Google, не підпадає під такі самі зобов’язання щодо безпеки, як одноразовий пароль, надісланий через SMS.

Однак електронна пошта, захищена за допомогою SMS MFA, потенційно може мати ті ж проблеми безпеки, що й інші служби, які використовують SMS MFA, а зламаний обліковий запис електронної пошти дозволить зловмиснику отримати доступ до одноразового паролю, надісланого електронною поштою.

1.2.2.2 Одноразовий пароль на основі часу

TOTP, як визначено Рахи та інші [21] у RFC-6238, працюють за допомогою попередньо визначеного псевдовипадково згенерованого секретного початкового числа, яке зберігається як у службі аутентифікації, так і на пристрої, керованому користувачем. Пристроєм може бути спеціалізований апаратний пристрій, наприклад Rivest-Shamir-Adleman (RSA) SecurID [22], або програма на мобільному пристрої, наприклад Google Authenticator. Шестизначний TOTP постійно генерується шляхом застосування секретного початкового числа до функції коду аутентифікації повідомлення на основі хешування (HMAC), використовуючи початкове число та час як параметри. На практиці тривалість часу дії TOTP становить тридцять секунд і відновлюється в кінці цього періоду. Користувач вводить шестизначний TOTP, який відображається на його або її пристрої, у службу, яка запитує. Потім служба, яка надсилає запит, окремо відтворює TOTP із початкового числа, яке зберігається в службі, і порівнює його з отриманим TOTP.

Коган та інші [9] стверджують, що основна перевага цього підходу полягає в тому, що для генерації цих кодів не потрібне підключення до мережі, оскільки сервіс і пристрій можуть обчислювати їх окремо, використовуючи одне й те саме попередньо визначене секретне число. Це означає, що підхід TOTP не має тих самих проблем, що й підхід SMS, коли код можна вкрасти під час передачі.

Коган та інші [9] пояснюють, що головним недоліком є те, що секретні початкові значення мають зберігатися у відкритому вигляді на службі, яка запитує, щоб служба могла перевірити TOTP користувачів. Якби зловмисник отримав неавторизований доступ до служби, зловмисник міг би викрасти секретне початкове число кожного зареєстрованого користувача та генерувати дійсні коди без обмежень. Дуже публічний приклад такого типу атаки став відомий після того, як мережа Lockheed Martin була

скомпрометована, як повідомив Дрю [10]. Незважаючи на це, Джовер [8] стверджує, що метод TOTP з мобільним додатком є найбезпечнішим способом виконання MFA без необхідності додаткового нестандартного обладнання.

1.2.3 Авторизація

Замість забезпечення підтвердження особи, як це робить аутентифікація, авторизація є способом забезпечення дозволу або прав доступу до системи чи служби. Можливо, користувач пройшов аутентифікацію, але не мав авторизації для доступу до ресурсу або навпаки. Повсякденним прикладом є різниця між ідентифікаційним бейджем і ключем. Ідентифікаційний бейдж може засвідчити особу, але він не відчинить замкнені двері. І навпаки, наявність ключа, який відкриває замкнені двері, не підтверджує особу, оскільки ключ міг бути втраченим або вкраденим. Більшість систем мають протоколи як для аутентифікації, так і для авторизації користувачів.

У системі з одним користувачем аутентифікований власник або основний користувач системи матиме адміністратор або кореневий доступ до всіх частин системи. Однак більшість сучасних систем надають багато послуг багатьом користувачам. Правильно налаштовані служби повинні дозволяти аутентифікованим користувачам отримувати доступ і налаштовувати власні дані в службі, одночасно запобігаючи будь-якому доступу користувачів без дозволу. Точний метод перевірки та забезпечення авторизації значною мірою залежить від системи, але, згідно з Харрісон та інші, авторизація, як правило, може бути зведена до реляційної матриці або списку контролю доступу (ACL), який відображає логічні значення (істина або хибність) користувачам і областям, до яких кожному користувачеві дозволено доступ.

1.2.3.1 Токени

JWT – це стандарт для зберігання закодованих заяв JavaScript Object Notation (JSON) у маркері, який широко використовується для надання та підтвердження авторизації.

JWT зберігає інформацію про службу чи організацію, яка випустила токен, користувача, для якого було видано токен, час видачі токена та час закінчення терміну дії токена. JWT також можуть мати цифровий підпис або шифрування їх емітентом, що дозволяє перевірити, чи токен не було підроблено з моменту його випуску.

Токени зазвичай зберігаються в браузері в локальному або локальному сховищі, але також можуть зберігатися в файлі cookie. Джонс та інші визначають процес обміну токенами для авторизації в RFC-7519. У разі використання для авторизації токени зазвичай надсилаються в заголовку запиту протоколу передачі гіпертексту (HTTP) у форматі: «Authorization: Bearer <token string>».

1.2.3.2 OAuth

OAuth – це відкритий стандарт протоколів, призначений для авторизації делегатів. Це означає, що служба може не мати змоги безпосередньо аутентифікувати користувача, але якщо інша служба вже аутентифікувала цього користувача, служба може поручитися за ідентифікацію користувачів.

Типовим прикладом цього є використання облікового запису Facebook для входу на веб-сайт «А». Користувач не має облікового запису, створеного на веб-сайті «А», але має обліковий запис на Facebook. Користувач може натиснути кнопку «Підключитися до Facebook», яка надсилає інформацію про користувача на веб-сайт «А».

Залежно від запиту на інформацію веб-сайту «А», інформація містить

принаймні токен авторизації та унікальний ідентифікатор користувача, який можна використовувати для ідентифікації користувача на веб-сайті «А», але ніколи не пароль користувача Facebook.

Можна зробити висновок, що OAuth – це простий стандарт авторизації, заснований на базових принципах інтернету, що уможливлює застосування авторизації практично на будь-якій платформі. Стандарт має підтримку найбільших майданчиків і очевидно, що його популярність лише зростатиме. Якщо ви задумалися про API для вашого сервісу, то авторизація з використанням OAuth гарний вибір.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Ціль

Запропонувати новий підхід до вирішення цих дилем, що дозволяє використовувати систему входу без пароля з використанням асиметричних ключів, цифрових підписів і біометрії для надання MFA.

Ціль – це створення системи, яка є більш безпечною або принаймні такою ж безпечною, як традиційна система аутентифікації за паролем користувача з використанням 2FA. Цілі цього проекту наступні:

– розробити мобільний додаток для iOS, який буде створювати та безпечно зберігати приватні ключі користувача;

– програма повинна відповідати за зв’язок із сервером для забезпечення аутентифікації;

– налаштувати веб-сервер Apache для демонстрації аутентифікації та авторизації за допомогою PHP та фреймворку Laravel, щоб забезпечити функціональність серверної частини;

– створити інтерфейс Single Page Application (SPA), який буде доступний для перегляду користувача, використовуючи React.js.

2.2 Вимоги до системи

Для створення такої системи потрібно, щоб вона відповідала наступним вимогам:

– зручність і зрозумілість;

– легка для користування;

– коректний зв’язок із сервером для забезпечення аутентифікації.

3 АРХІТЕКТУРА

3.1 Огляд системи

Система представлена логічно розділеними компонентами. Процес аутентифікації, який використовується для цього проекту, працює наступним чином.

3.1.1 Реєстрація

Ця кваліфікаційна робота припускає, що на пристрої користувача вже встановлена програма.

На рисунку 3.1 зображено процес реєстрації користувача, а нижче описані пункти, які зображені на цьому рисунку.

Рисунок 3.1 – Процес реєстрації користувача

Описано такі пункти:

1) користувач відкриває програму на своєму мобільному пристрої;

2) користувач заповнює реєстраційну форму на мобільному пристрої;

3) під час реєстрації Secure Enclave генерує пару ключів;

4) на сервер надсилається відкритий ключ, ім’я користувача, номер телефону та адреса електронної пошти;

5) сервер створює нового користувача та зберігає інформацію про нього в базі даних;

6) сервер надсилає відповідь JSON із створеним ідентифікатором користувача та статусом HTTP 201 CREATED на телефон після успішного створення користувача або код помилки в іншому випадку.

3.1.2 Аутентифікація

Якщо користувач уже зареєстрований, він просто авторизується в програмі. Після реєстрації та входу користувач пристрій буде перенаправлено на екран перегляду домашньої камери. Згідно цьому у користувача є два шляхи:

1) користувач натискає кнопку «Вхід» на сайті SPA. SPA надсилає запит Axios API до кінцевої точки /random. Створюється псевдовипадковий криптографічно захищений 64-байтовий одноразовий номер, кодується рядок base64 і повертається до SPA. QR-код генерується та відображається в SPA з рядком nonce;

2) SPA починає прослуховування на каналі, ідентифікованому за допомогою nonce для трансляції «підтвердження».

Користувач сканує QR-код на SPA за допомогою домашньої камери на мобільному пристрої. На пристрої створюється пакет у кодуванні JSON, який містить ідентифікатор користувача та nonce, що міститься у відсканованих даних із QR-коду. Хеш SHA-512 створюється з пакета. Потім

пристрій надсилає запит до Secure Enclave, щоб підписати комплект за допомогою закритого ключа на пристрої. Користувачеві пропонується виконати біометричну аутентифікацію. Якщо біометрична автентифікація не вдається, процес зупиняється. В іншому випадку комплект підписується за допомогою «American National Standards Institute» (ANSI) X9.62 з хешем SHA-512. Потім пакет, хеш і підпис вкладаються в корисне навантаження. Корисне навантаження закодовано base64 і надсилається на сервер у запиті API до кінцевої точки /login у тілі запиту.

Щойно сервер отримує запит автентифікації з корисним навантаженням, він розгортає корисне навантаження та декодує base64, а JSON декодує пакет. Сервер шукає користувача, який відповідає ідентифікатору користувача, що міститься в пакеті. Якщо користувача знайдено, об’єкт користувача витягується за допомогою Eloquent ORM, а відкритий ключ користувача витягується з бази даних. Сервер повторно обчислює хеш SHA-512 на пакеті та порівнює його з хешем, що міститься в корисному навантаженні. Якщо значення збігаються, сервер перевіряє наявність підпису в пакеті, base64 декодує його та використовує PHP OpenSSL verify для перевірки підпису на відповідність відкритому ключу користувача. Якщо користувача не знайдено, обчислений хеш не відповідає надісланому, підпис не знайдено або підпис не перевірено, запитувачу повертається неавторизований код HTTP 401.

Після перевірки підпису створюється персональний носій JWT, унікальний для користувача з об’єкта користувача. Термін дії JWT становить один тиждень із дати створення. JWT розміщується в кеш-пам’яті сервера, використовуючи рядок nonce як ключ і JWT як значення, із часом життя (TTL) у дві секунди. Якщо JWT не буде отримано протягом цього періоду, його буде видалено з кешу. Потім сервер надсилає подію LoginAuthorized, яка транслює «approval-granted» на ідентифікаторі каналу nonce і повертає статус JSON HTTP 200 OK разом із створеним JWT на мобільний пристрій.

Після того, як SPA, що прослуховує, отримує трансляцію «надано схвалення», він надсилає запит Axios API до кінцевої точки /login/confirm разом із nonce у тілі запиту. Сервер перевіряє кеш на наявність ключа, який відповідає отриманому одноразовому ключу, якщо його не знайдено, запитувачу повертається неавторизований статус HTTP 401. В іншому випадку JWT витягується з кешу, а запис кешу видаляється, щоб запобігти атакам із відтворенням у часі. Отриманий JWT надсилається назад запитуючому клієнту разом із статусом HTTP 200 OK. Клієнт Axios отримує JWT і зберігає його в локальному сховищі браузера. Клієнт Axios отримує JWT, якщо він присутній, і надсилає його в заголовку кожного запиту до сервера. Нарешті, користувач перенаправляється на головну сторінку. Тепер клієнт браузера користувача має право робити будь-які запити ресурсів, оскільки JWT зберігається в localStorage, і Axios автоматично надсилатиме JWT у заголовку кожного запиту.

На рисунку 3.2 зображено процес аутентифікації користувача, який описано вище.

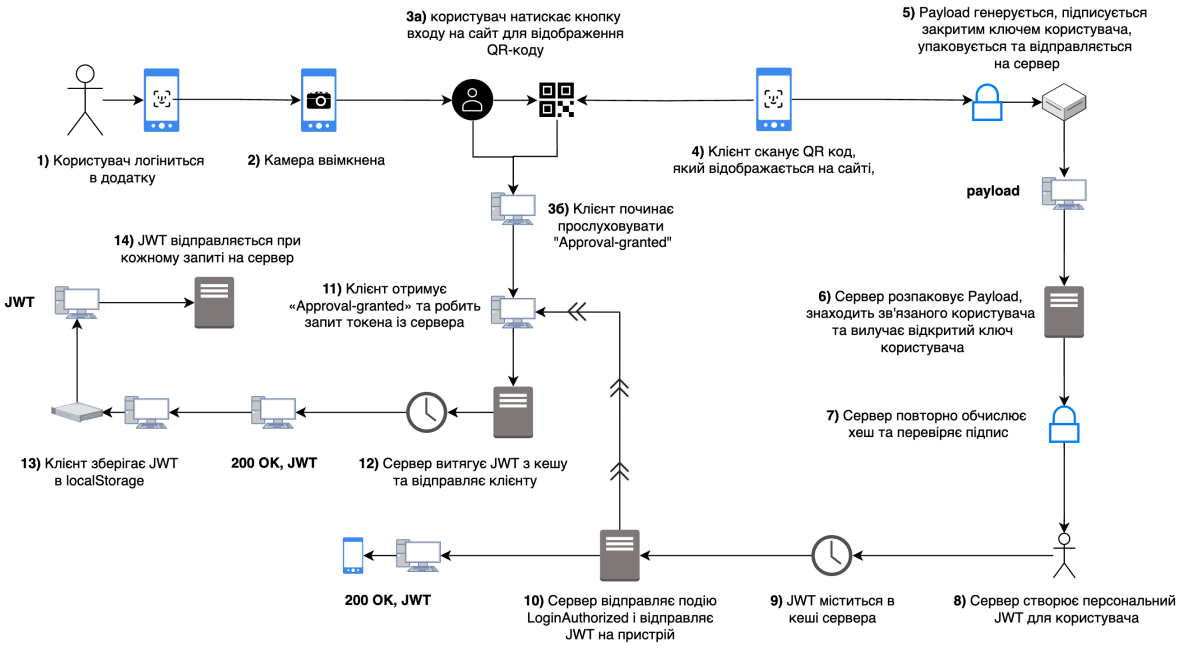


Рисунок 3.2 – Процес аутентифікації користувача

3.1.3 Вихід із системи

Оскільки весь метод аутентифікації не має стану, користувач вийде з системи лише після закінчення терміну дії JWT або після виходу користувача вручну. LocalStorage браузера зберігається на вкладках, вікнах і коли браузер закрито. Визначено кінцеву точку API /logout, доступ до якої можна отримати з мобільного пристрою користувача або безпосередньо в браузері. Браузер і мобільний пристрій не отримують сповіщення автоматично, коли інший пристрій виходить із системи, хоча процес виходу майже ідентичний для обох пристроїв.

Під час виходу з системи через будь-який пристрій викликається /logout API, який скасовує та видаляє JWT користувача із сервера, роблячи будь-які подальші запити несанкціонованими. Під час виходу з веб-переглядача JWT видаляється з localStrage браузера, і користувач перенаправляється назад на сторінку входу. Під час виходу з мобільного пристрою користувачеві нагадують закрити браузер, оскільки браузер залишатиметься на сторінці, до якої він входив востаннє до закриття. Після повторного відкриття браузера SPA переспрямовує на сторінку входу та видаляє будь-які JWT, які були присутні в локальному сховищі браузера. Цей процес зображено на рисунку 3.3.

Рисунок 3.3 – Процес виходу із системи

3.2 Технології реалізації

3.2.1 HTML

HTML – скорочення від «HyperText Mark-up Language» – перекладається як «Мова розмітка гіпертексту». Гіпертекст – це текст, що не послідовно зв'язаний з іншими документами, тобто у вас є змога з першої сторінки документу перейти на останню. Іншими словами HTML – це мова розмітки, або ще один спосіб зберігання інформації. За допомогою HTML ти позначаєш текст, вказуючи своєму веб-переглядачу, як він має розуміти позначений текст, так само як і на жорсткому диску інформація зберігається в блоках, кластерах, секторах, доріжках і тільки за допомогою, такої, визначеної структури твій комп'ютер розуміє, що треба, а що не треба зчитувати.

У HTML текст позначається за допомогою теґів. Кожен HTML документ буде складатися з деякої групи елементів, де кожен елемент буде визначатися (починатися та закінчуватися) певним теґом (для деяких елементів кінцевий теґ не є обов'язковим). Теґ – це назва елементу, записана у кутових дужках (< >).

Кожен HTML теґ має свою унікальну назву з визначеним синтаксисом, яка записується латинськими літерами і не чутливий до регістру.

3.2.2 CSS

CSS (абревіатура від Cascading Style Sheets, що в перекладі означає каскадні таблиці стилів) – це спеціальна мова (мова стилів), за допомогою якої описують вигляду документів (як і де відображати елементи веб-сторінки), написаних мовами розмітки даних. Найчастіше CSS використовується для документів, котрі розмічені мовою HTML, XHTML та

XML.

Зараз HTML в чистому вигляді має дуже обмежений набір інструментів, що не дозволяє вирішувати ті чи інші дизайнерські та функціональні замисли веб-ремісників. Ну ось хоч би, до прикладу, взяти початкове запитання всіх веб- ремісників «Як прибрати підкреслення у посиланні?» або «Як змінити стиль посилання, при наведенні на нього курсора?». За допомогою лише одного HTML такого зробити не вдасться!. А таких запитань безліч. Тут й приходить на допомогу CSS, який вирішує більшість завдань, що відносяться до стильового оформлення сторінки.

Одна з головних переваг використання CSS – це можливість розділити зміст сторінки від її оформлення. Таке розділення дозволило покращити сприйняття та доступність змісту, забезпечити більшу гнучкість та контроль за відображенням змісту в різних умовах, зробити зміст більш структурованим та простим, прибрати повторення та інші Власне це ж і була основна мета створення цієї технології.

Що дає використання CSS:

– відображати один і той же документ в різних стилях;

– декілька дизайнів сторінки для різних пристроїв. Наприклад, на екрані дизайн буде розрахований на велику ширину, під час друку меню не виводитиметься, а на смартфоні меню буде внизу, під вмістом;

– зменшення часу завантаження сторінок сайту за рахунок перенесення правил відображення в окремий CSS-файл. В цьому випадку браузер завантажує тільки структуру документа і дані, що зберігаються на сторінці, а стильові правила цих даних завантажуються браузером тільки один раз і кешуються;

– простота подальшої зміни дизайну. Не потрібно правити кожну сторінку, а лише змінити CSS-файл;

– додаткові можливості оформлення. Наприклад, за допомогою CSS-розмітки можна зробити так, щоб меню було завжди видно при

скролінгу сторінки, або прибрати підкреслення у посилань;

– дозволяє створювати складну і пропрацьовану техніку дизайну.

3.2.3 JavaScript

JavaScript – це повноцінна динамічна мова програмування, яка застосовується до HTML документу, і може забезпечити динамічну інтерактивність на веб-сайтах. JavaScript неймовірно універсальний і доброзичливий до новачків. Маючи великий досвід, за допомогою цієї мови програмування можна створювати ігри, анімовану 2D і 3D графіку, повномасштабні програми з базами даних і багато іншого! JavaScript сам по собі досить компактний, але дуже гнучкий. Розробниками написано велику кількість інструментів поверх основного мови JavaScript, які розблокують величезна кількість додаткових функцій з дуже невеликим зусиллям. До них відносяться:

– програмні інтерфейси додатка (API), вбудовані в браузери, що забезпечують різні функціональні можливості, такі як динамічне створення HTML і установку CSS стилів, захоплення і маніпуляція відео потоків, робота з веб-камерою користувача або генерація 3D графіки і аудіо семплів;

– сторонні API дозволяють розробникам впроваджувати функціональність в свої сайти від інших розробників, таких як Twitter або Facebook;

– також можна застосовувати до HTML сторонні фреймворки і бібліотеки, що дозволить прискорити створення сайтів і додатків.

3.2.4 PHP

PHP (абревіатура від Hypertext Preprocessor) – це мова програмування, яка використовується для створення динамічних веб-сторінок і веб-додатків. PHP є безкоштовним програмним забезпеченням з відкритим

вихідним кодом, що забезпечує зручну і просту розробку веб-сайтів.

PHP виконується на сервері, тому він може генерувати HTML-код, який відправляється клієнту, що запитує сторінку. Це дозволяє вбудовувати веб-сторінки з динамічним вмістом, таким як форми, калькулятори, системи управління вмістом (CMS) і багато іншого.

Одна з головних переваг PHP полягає у тому, що він має велику спільноту розробників, які створюють безліч доповнень і бібліотек для полегшення розробки веб-додатків. Крім того, PHP дозволяє легко взаємодіяти з базами даних, такими як MySQL і PostgreSQL, що робить його улюбленим вибором для розробників веб-додатків.

Однак, PHP не є ідеальним інструментом для всіх веб-проектів. Наприклад, він може бути повільнішим за інші мови програмування, такі як Node.js, у випадку з великими веб-додатками. Також, зі зростанням кількості мов програмування та технологій, PHP може бути менш популярним серед розробників веб-додатків. У будь-якому випадку, PHP є потужним інструментом для розробки веб-додатків і є варіантом для тих, хто шукає зручний спосіб створення динамічних веб-сторінок і веб-додатків.

За допомогою PHP можна розробляти різноманітні веб-додатки, включаючи соціальні мережі, онлайн-магазини, блоги, форуми та багато іншого. PHP підтримує такі функції, як робота з файлами і директоріями, керування cookies, робота з формами і валідація даних, а також можливість роботи з XML-документами і веб-сервісами.

PHP також має вбудовану функціональність для роботи з об'єктами, що дозволяє писати більш структурований і модульний код. Крім того, PHP дозволяє взаємодіяти з іншими сервісами, такими як HTTP-запити і API, що робить його потужним інструментом для розробки веб-додатків. PHP також має велику кількість фреймворків, які спрощують розробку веб-додатків, таких як Laravel, Symfony і CodeIgniter. Ці фреймворки мають безліч корисних функцій і бібліотек, що дозволяє розробникам більш швидко і

ефективно створювати веб-додатки.

Отже, PHP - це мова програмування з великою кількістю функцій і фреймворків, що дозволяє розробникам створювати динамічні веб-сторінки і веб-додатки з великою кількістю різноманітних функцій і можливостей.

3.2.5 Laravel

Laravel – це фреймворк PHP, який дає змогу використовувати стратегію програмування MVC, створюючи чисті, гнучкі та масштабовані серверні програми. Laravel містить ORM API, Eloquent, який дозволяє розділяти дані та бізнес-логіку. Laravel також не залежить від інтерфейсу, що дозволяє розробникам обирати включений Blade Engine Laravel для традиційних сторінок, написаних на PHP, і відтворюваних на мові гіпертекстової розмітки (HTML) або використовувати маршрутизацію API, щоб надсилати запити від окремого клієнта інтерфейсу. Цей проект використовував останній підхід із Laravel версії 8.7.1. Laravel також включає Axios, сторонню бібліотеку запитів, яка використовувалася для створення XMLHttpRequests у браузері.

3.2.6 React.js

React.js – це бібліотека JavaScript, яка дозволяє створювати інтерактивні і ефективні інтерфейси користувача для веб-додатків. Розроблена компанією Facebook, React.js стала однією з найбільш популярних бібліотек для фронтенд-розробки.

Основними принципами React.js є компонентний підхід та використання віртуального DOM. За допомогою компонентного підходу розробник може створювати малий і самодостатній код, який може бути повторно використаний. Компоненти можуть бути простими або складними і взаємодіяти між собою, щоб створити складний інтерфейс користувача.

Віртуальний DOM є іншою ключовою особливістю React.js. Він дозволяє ефективно оновлювати сторінки веб-додатків, що зменшує навантаження на сервер та забезпечує швидке оновлення інтерфейсу користувача. React.js забезпечує швидку і ефективну роботу зі структурою DOM, що зменшує час оновлення сторінок та забезпечує плавну роботу інтерфейсу користувача.

Окрім цього, React.js може використовуватись з різними іншими бібліотеками та фреймворками, такими як Redux, Angular, Vue.js, що робить його дуже гнучким і зручним для використання. React.js також має багато різноманітних додатків та інструментів, таких як React Developer Tools, що полегшують розробку веб-додатків з React.js.

Отже, React.js – це потужна бібліотека JavaScript, що дозволяє створювати швидкі, ефективні та інтерактивні інтерфейси користувача для веб-додатків за допомогою компонентного підходу та віртуального DOM.

Оскільки React.js використовувався для створення SPA в цьому проекті, кожен компонент використовувався як представлення, яке можна розглядати як сторінку Uniform Resource Locator (URL). Під час доступу до певної URL-адреси React Router завантажує шаблон компонента та будь-який код і стилі, визначені в розділах сценарію та стилю.

3.2.7 React Router

Незважаючи на те, що Laravel надає функції маршрутизації, JavaScript SPA, що використовує серверний API, вимагає використання маршрутизації JavaScript. Це було зроблено за допомогою основної бібліотеки аддонів React, React Router. React Router також містить можливість додавати навігаційні захисники, які використовувалися для захисту захищених маршрутів від неавторизованого та неаутентифікованого доступу. Це було досягнуто шляхом спочатку визначення списку слагів або іменованих URL-адрес. Потім ці слаги були зіставлені з конкретними компонентами React.

Наприклад, під час доступу до слага /login повертається компонент Login. Під час визначення цих зіставлень кожному захищеному маршруту було призначено асинхронну функцію beforeEnter. Коли слаг отримує доступ через браузер, маршрутизатор спочатку перевіряє, чи є на маршруті функція beforeEnter. Якщо маршрутизатор виявляє функцію, вона викликається.

Цей проект визначив функцію isAuthenticated, яка повертає логічне значення true, якщо веб-переглядач аутентифіковано, і false в іншому випадку. Функція isAuthenticated викликає визначений проектом маршрут API /isAuthenticated. Цей маршрут надсилається через драйвер Laravel Passport API, який перевіряє, чи було надіслано дійсний маркер у запиті. Маршрут повертає код статусу HTTP 200 OK, якщо маркер дійсний, або код статусу HTTP 401 неаутентифікований в іншому випадку. Якщо отримано код статусу 200, функція isAuthenticated повертає true, інакше повертає false. Функція beforeEnter викликає функцію isAuthenticated кожного разу, коли здійснюється доступ до захищеного маршруту. Якщо функція повертає true, тоді повертається запитуваний маршрут, інакше повертається маршрут входу, а будь-які JWT, присутні в локальному сховищі, видаляються.

3.2.8 Мобільний пристрій

Для зчитування QR-кодів потрібен пристрій, оскільки потрібен принаймні один аутентифікатор. З цієї причини потрібен мобільний пристрій з камерою. О’Ді стверджує, що понад три мільярди людей є власниками смартфонів у всьому світі. Дослідницький центр Pew показує, що смартфони стали майже повсюдними у Сполучених Штатах, у 2021 році понад 85 відсотків американців володіли смартфонами. Попередні статистичні дані свідчать про те, що більшість користувачів мають смартфони, і оскільки смартфони вже використовуються для кількох типів МЗС, смартфон був би хорошим вибором для цього проекту. Доступно кілька моделей і марок смартфонів, однак існує кілька вимог до апаратного

забезпечення, які можуть забезпечити найкращу безпеку з найменшими незручностями для користувача, наприклад: біометрична автентифікація, камера з достатньою роздільною здатністю та мобільний пристрій із безпечний анклав, ізольований апаратний контейнер, який зберігатиме особистий ключ користувача.

Оскільки закритий ключ користувача зберігається на мобільному пристрої, необхідно захистити ключ у разі втрати або зламу пристрою. Захист ключа здійснюється за допомогою Secure Enclave. Більшість пристроїв смартфонів мають налаштування, які дозволяють користувачеві захистити їх паролем, хоча це оптимально, цей проект не передбачає, що користувач увімкнув це налаштування, тому до програми додано додатковий логін. Оскільки багато моделей смартфонів, які використовуються, мають доступний метод біометричної автентифікації, що полегшує роботу користувача, інтегрована біометрія використовується там, де це можливо.

Apple iPhone XR відповідає всім цим вимогам і був легко доступний. Тому для цього проекту було використано Apple iPhone XR з iOS 15.

3.2.9 Secure Enclave

Secure Enclave – це ізольований апаратний контейнер у підтримуваному головному центральному процесорі (CPU) Apple. Secure Enclave додає ще один рівень захисту до ключа користувача, оскільки пари ключів не можна імпортувати або вилучати з Secure Enclave. Контейнер має власний процесор, який суворо обмежений операціями в межах Secure Enclave і працює на нижчій тактовій частоті, щоб запобігти атакам з перебором годинника та потужності. Контейнер Secure Enclave не має власної пам’яті, але має механізм захисту пам’яті, який секвеструє та шифрує захищену область пам’яті під час завантаження з основного контролера пам’яті, доступ до якого доступний лише Secure Enclave.

Secure Enclave також зберігає біометричні дані користувача, наприклад дані про відбитки пальців або обличчя. Біометричні дані зберігаються та посилаються лише в Secure Enclave. Дані не можна експортувати, Apple не може отримати до них доступ, і, якщо пристрій зламано, їх неможливо легко витягти з пристрою.

Цей проект використовував Secure Enclave для створення пари відкритих і приватних ключів користувача, зберігання закритого ключа користувача та цифрового підпису корисного навантаження користувача під час входу. Пари ключів генеруються Secure Enclave, і хоча відкритий ключ можна експортувати, закритий ключ залишається в самому Secure Enclave. Програми, які створюють пари ключів за допомогою Secure Enclave, насправді ніколи не бачать приватний ключ, а лише отримують посилання на приватний ключ і отримують лише результат операцій, які виконує Secure Enclave. Це означає, що в разі зламу пристрою закритий ключ користувача не можна легко отримати з пристрою.

3.2.10 Ключі

Пари ключів, створені в Secure Enclave, являють собою 256-бітні ключі ECDSA з параметром первинного випадкового домену secp256r1, який можна порівняти з криптографічною міцністю 3072-бітного ключа RSA або цифрового підпису (DSA). Це означає, що вони набагато менші, а їх підпис і перевірка є швидшими, ніж їхні еквівалентні ключі RSA або DSA. Під час створення ключа було додано два додаткові параметри: cAttrAccessibleWhenUnlockedThisDeviceOnly, що дозволяє отримати доступ до ключа, лише коли пристрій розблоковано, а програма працює на передньому плані, і biometryAny, який робить ключ доступним, лише якщо пристрій може аутентифікувати користувач використовує доступний біометричний аутентифікатор.

Ключі ECDSA обчислюються за допомогою еліптичних кривих над

скінченним полем, визначеним параметром домену. Підпис, створений із 256-бітного ключа ECDSA, має 512 бітів і перевіряється шляхом повторного обчислення хешу за допомогою алгоритму, указаного в підписі, а потім відновлення точки на кривій за допомогою відкритого ключа та перевірки, що це та сама точка, яка була випадково генерується під час підписання.

На рисунку 3.4 зображено захищені компоненти Enclave на мікросхемі.

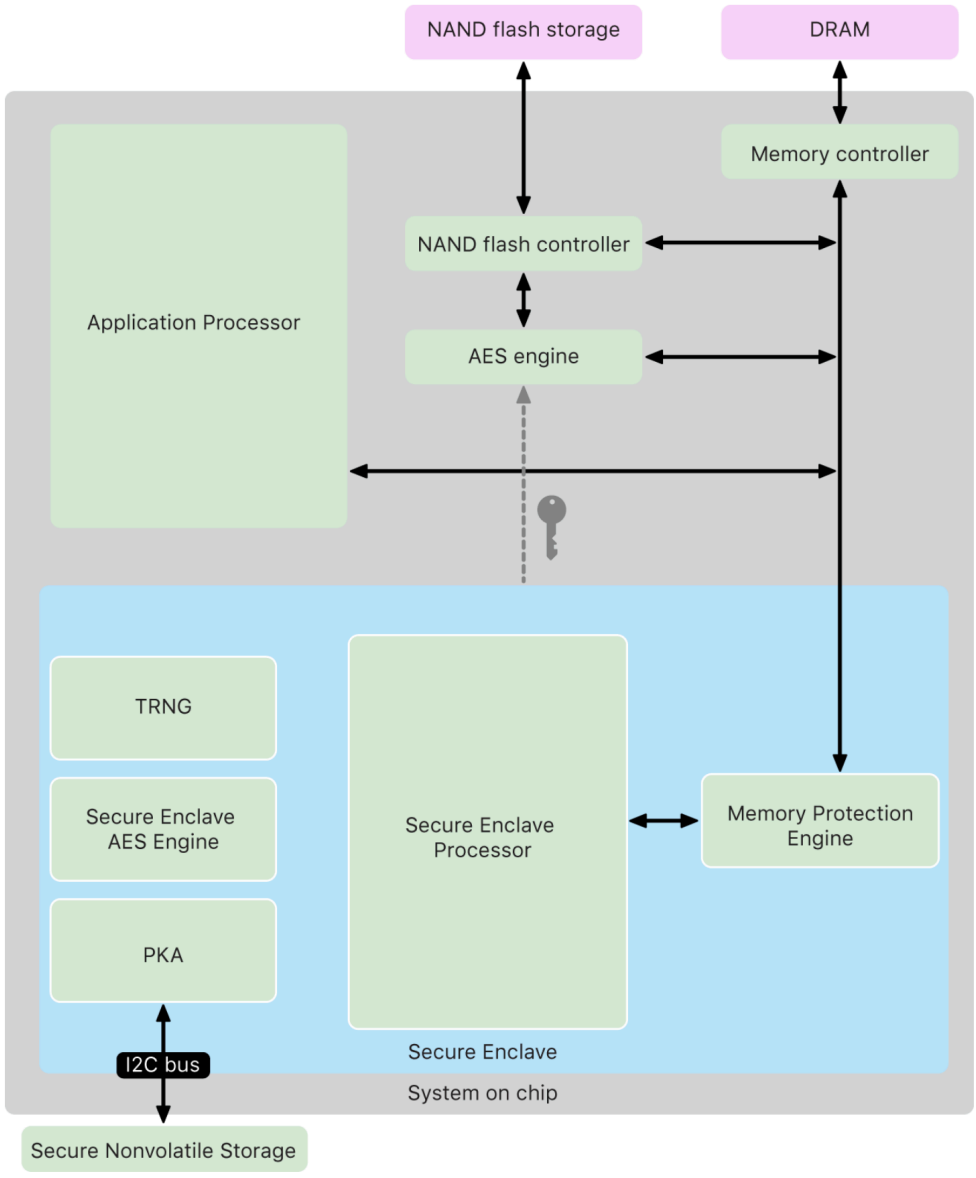


Рисунок 3.4 – Захищені компоненти Enclave на мікросхемі

Під час підготовки відкритого ключа до експорту Apple Software Development Kit (SDK) експортує публічні ключі ECDSA у формат ANSI X9.63 (04 ||X ||Y). Хоча OpenSSL підтримує ключі X9.63, бібліотека PHP OpenSSL вимагає, щоб ключі були відформатовані в Privacy Enhanced Mail (PEM). Щоб забезпечити перевірку ключа за допомогою PHP OpenSSL, ключ спочатку перетворюється у формат правил кодування (DER), а також додається заголовок ідентифікатора об'єкта Abstract Syntax Notation One (ASN.1) для параметра домену secp256r1. Потім ключ кодується за основою 64 і загортається в заголовок PEM, а саме в закриваючі теги – BEGIN PUBLIC KEY та END PUBLIC KE.

На рисунку 3.5 показано приклад ключа ECDSA sec256r1 у форматі PEM, експортованого з Secure Enclave.

```
-----BEGIN PUBLIC KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAArAAAABNlY2RzYS
1zaGEyLW5pc3RwNTIxAAAACG5pc3RwNTIxAAAAhQQAyf5OuwTDQ8gBiOuqYX39EyXhq84M
x2s0Zvz/dNXwU7KsIJENJ/6FYnS9Jr87o9nUOEj6QQOjwDjesRYPJ4AoqRYAHV6ii1HTlA
z3b32J6iduZeJsrsfF94YGVmm2ZbBmYg4roZ0xIjwZLLoX7u01OU83kSCcyUiDelG0WO
-----END PUBLIC KEY-----
```

Рисунок 3.5 – Експортний відкритий ключ, закодований PEM

3.2.11 Мобільний додаток

Щоб використовувати всі апаратні компоненти Apple iPhone, була потрібна рідна програма. Apple вимагає, щоб рідні програми використовували їх інтегроване середовище розробки Xcode (IDE), яке доступне лише через Apple App Store на системах macOS. Декілька апаратних засобів і частин інтерфейсу користувача Swift вимагали використання Apple iOS SDK версії 13 або новішої, яка доступна лише через Xcode 11. Однак, оскільки для цього проекту використовувалася iOS 14, для якого потрібен Xcode 12, використовувався Xcode 12 на Macbook Pro із

встановленою macOS 10.15.4 Catalina.

Мобільний додаток було створено з використанням мови Swift. Мобільний додаток потребувало використання кількох сторонніх бібліотек. CocoaPods використовувався як менеджер залежностей для отримання всіх сторонніх бібліотек. Для створення форми реєстрації користувача використовувався конструктор форм Eureka. BCryptSwift, реалізація bcrypt у Swift, використовувалася для безпечного зберігання кодів доступу користувачів у хешованій та доданій формі у в'язці ключів. OAuthSwift використовувався для обробки маркерів і підключень до сервера.

3.2.12 BCrypt

BCrypt – це звичайний алгоритм, який використовує переваги інтенсивного налаштування ключа в eksblowfish, щоб безпечно зберігати паролі користувачів у хешованій та засоленій формі. Його головна перевага полягає в тому, що його робоче значення регулюється.

Коефіцієнт роботи – це кількість часу, необхідна для хешування пароля та соління n раундів. Коефіцієнт роботи слід відрегулювати до найвищого допустимого значення, оскільки прогрес у апаратному забезпеченні значно зменшить коефіцієнт роботи. За Порніним значення робочого фактора повинно бути не менше 241 мс. Це значення часу гарантує, що користувацький досвід не постраждає, забезпечуючи достатній захист від атак грубої сили та словникових атак. Бібліотека BCryptSwift за замовчуванням встановлює n на десять раундів, але дозволяє регулювати коефіцієнт роботи, встановлюючи для n ціле число від чотирьох до шістнадцяти раундів. Кожного разу, коли n збільшується, коефіцієнт роботи зростає експоненціально.

Під час тестування бібліотеки BCryptSwift на Apple iPhone X виявилося, що найбільша кількість тестованих раундів, чотирнадцять, вимагала робочого фактора приблизно дев'ять хвилин. Чотири раунди, як

мінімум, вимагали приблизно 0,54 секунди, а десять раундів, за замовчуванням, вимагали приблизно 31,3 секунди. Було обрано шість раундів з найвищим допустимим коефіцієнтом роботи приблизно дві секунди (1,96 с). Це призвело до додаткових двох секунд затримки під час реєстрації та під час входу в програму за допомогою пароля.

3.2.13 Сервер

Фреймворк Laravel надав внутрішній API разом із кількома бібліотеками, включаючи основну бібліотеку OAuth 2.0, Passport версії 10.0.1, для обробки автентифікації та видачі токенів клієнтам. SPA було створено за допомогою Vue.js, мінімалістичного фреймворку JavaScript, який дозволив зберігати та отримувати маркери з локального сховища браузера та запити до внутрішнього API. Бібліотека PHP OpenSSL використовувалася разом із OpenSSL версії 1.1.1 для перевірки хешу та цифрового підпису під час автентифікації.

Сервер був розміщений за допомогою Apache 2.4.46 з PHP 7.4.9 і MySQL Community Server версії 8.0.21 як механізм зберігання даних на локальній машині Windows 10. Підключення до сервера було захищено за допомогою TLS 1.3 з 2048-бітним ключем RSA та сертифікатом від довіреного кореневого CA Let’s Encrypt.

3.2.14 QR-коди

QR-коди є формою універсального штрих-коду, який можна сканувати з будь-якого напрямку на 360 градусів практично будь-якою маркою та моделлю смартфона без спеціального апаратного чи програмного забезпечення. Хоча існує кілька типів штрих-кодів, в своєму дипломі я використовую QR-коди, які є майже усюдисущими та їх легко сканувати.

QR-коди можуть зберігати до трьох кілобайт практично будь-яких

типів даних і дуже стійкі до пошкоджень. Ця здатність зберігати дані відрізняється від звичайних двовимірних штрих-кодів (таких як штрих-код, що використовується в універсальних кодах продуктів), які здатні зберігати максимум двадцять цифр.

Оскільки ця кваліфікаційна робота потребуватиме візуалізації 64-байтового nonce у зручному для мобільних пристроїв форматі, QR-коди найкраще підходять для цього використання.

Для відтворення QR-кодів на сервері використовувалася бібліотека компонентів React, reactqrcode, оболонка для node-qrcode, яка базується на «QRCode для JavaScript» Кадзухіко Арасе.

Бібліотеку було встановлено через менеджер пакетів вузлів (npm), а компонент імпортовано в основну програму React. Нарешті, відтворення QR-коду на сторінці було здійснено шляхом додавання елемента HTML у розділ шаблону компонента Login.

3.2.15 Ngrok

Ngrok – це безкоштовний сервіс, який запускає встановлену програму, яка дозволяє розробникам відкривати свої локальні сервери за трансляцією мережевих адрес (NAT) і брандмауерами для загальнодоступного Інтернету, не вдаючись до переадресації портів.

Це робиться шляхом створення тунелю від локальної машини до загальнодоступного Інтернету та надання публічної URL-адреси для доступу до сервера. URL-адреса створюється як субдомен на ngrok.io у вигляді випадкового унікального шістнадцяткового рядка,

Наприклад d758984e9d54, за яким слідує ngrok.io, де повна URL-адреса буде http://d758984e9d54.ngrok.io.

На рисунку 3.6 показана консоль Ngrok з активним тунелем, де створено URL-адресу.

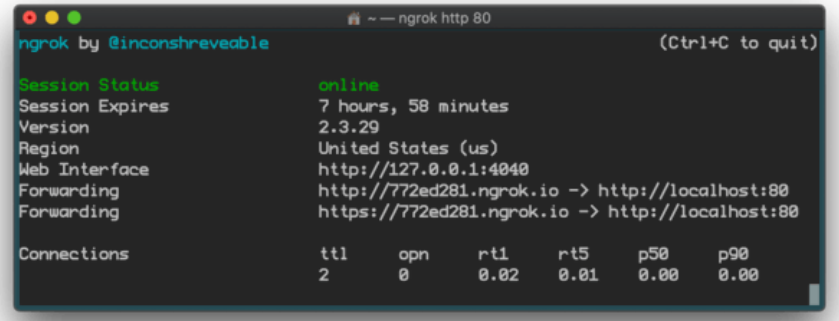


Рисунок 3.6 – Консоль Ngrok у командному рядку з прикладом створеної URL-адреси

Хоча основна послуга безкоштовна, є низка функцій, які доступні лише за платну місячну підписку, як-от можливість створити тунель у спеціальному домені або безпечний тунель TLS.

Для цього проекту потрібне дійсне доменне ім’я, захищене TLS, для якого зазвичай потрібна спеціальна адреса Інтернет-протоколу (IP). Ngrok пропонує можливість створити тунель TLS на власному домені, попередньо зарезервувавши користувацький домен на інформаційній панелі Ngrok.

На рисунку 3.7 показано приклад створеного запису CNAME, який вказує на власний субдомен для Ngrok. Потім це спеціальне доменне ім’я використовувалося для всіх запитів до сервера від мобільної програми.

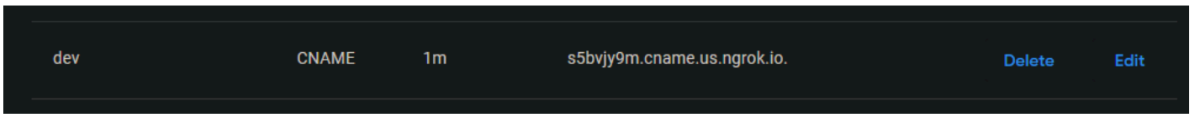


Рисунок 3.7 – Запис CNAME додано до реєстратора DNS

На рисунку 3.8 показано приклад резервування власного домену на інформаційній панелі Ngrok. Потім власне доменне ім’я вказується на Ngrok шляхом створення запису CNAME у реєстраторі DNS.

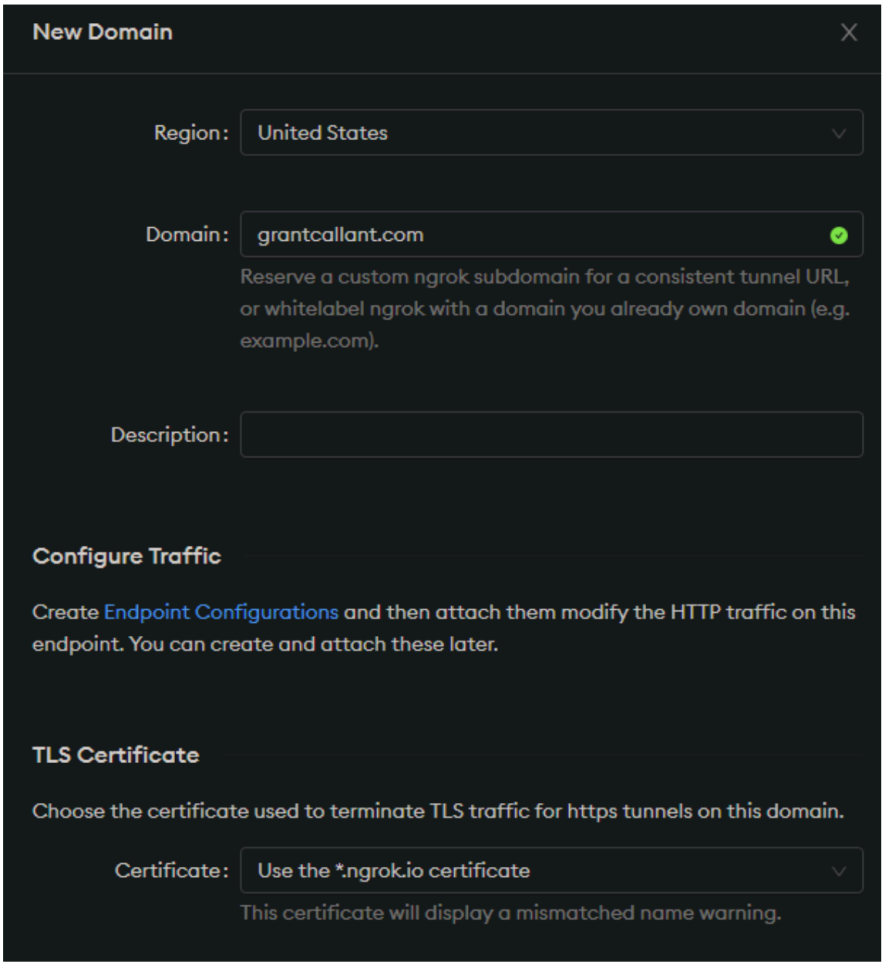


Рисунок 3.7 – Інформаційна панель Ngrok резервує домен

3.2.16 Сповіщення

Оскільки аутентифікація виконується з телефону, настільний клієнт має отримати сповіщення про успішну автентифікацію. WebSockets надають можливість надсилати клієнтам оновлення в реальному часі, що ефективніше, ніж постійне опитування, щоб перевірити, чи оновлено дані. Laravel містить функції трансляції та подій, які надсилатимуть події через WebSocket і дозволятимуть обробку подій від слухача. Laravel пропонує два драйвери для трансляції на стороні клієнта, Ably і Pusher Channels. Цей проект використовував Pusher Channels.

3.2.17 Laravel Echo

Laravel містить вбудовану функцію для прослуховування та обробки подій на стороні сервера, однак ці події не видно окремим інтерфейсним клієнтам. Laravel має власну бібліотеку JavaScript під назвою Laravel Echo, яка дозволяє клієнтам JavaScript прослуховувати канали трансляції для будь-яких надісланих подій. Це робиться, спочатку підписавшись на канал, який клієнт бажає слухати, і на яку подію слухати. Потім клієнт може обробити подію [42].

3.2.18 Pusher каналів

Pusher Channels – це послуга преміум-підписки з обмеженим безкоштовним планом Sandbox для розробників з менш ніж двома сотнями тисяч повідомлень на день і менш ніж сотнею одночасних підключень. Для цього проекту Pusher діє як міст між серверною частиною Laravel і інтерфейсом Laravel Echo. Laravel відправляє подію та транслює її в Pusher, який потім перенаправляє подію в Laravel Echo. Pusher містить веб-панель із консоллю налагодження. Підписки на канали та події відображаються на консолі в режимі реального часу

3.2.19 Інтерфейс користувача на робочому столі

Інтерфейс робочого столу включає в себе кілька частин розміщеного SPA, які користувачі не можуть контролювати. Простий прототип SPA був створений для зберігання колекції активів. Ці ресурси будуть доступні для завантаження будь-яким аутентифікованим користувачем. Немає інтерфейсу адміністратора, і аутентифікованим користувачам не потрібна додаткова авторизація. Рисунок 3.9 ілюструє приклад SPA для неаутентифікованого користувача. Зліва показано меню. Меню

відображається завжди, оскільки SPA не потрібно оновлювати сторінку, щоб оновити вміст сторінки. Жирним шрифтом відокремлюються розділи в меню, а текст під ним містить посилання на вміст. Неаутентифікований користувач може клацнути будь-яке посилання, але буде постійно перенаправлятися назад на сторінку входу без будь-яких відгуків. Єдине, що може зробити неаутентифікований користувач, це натиснути синю кнопку входу.

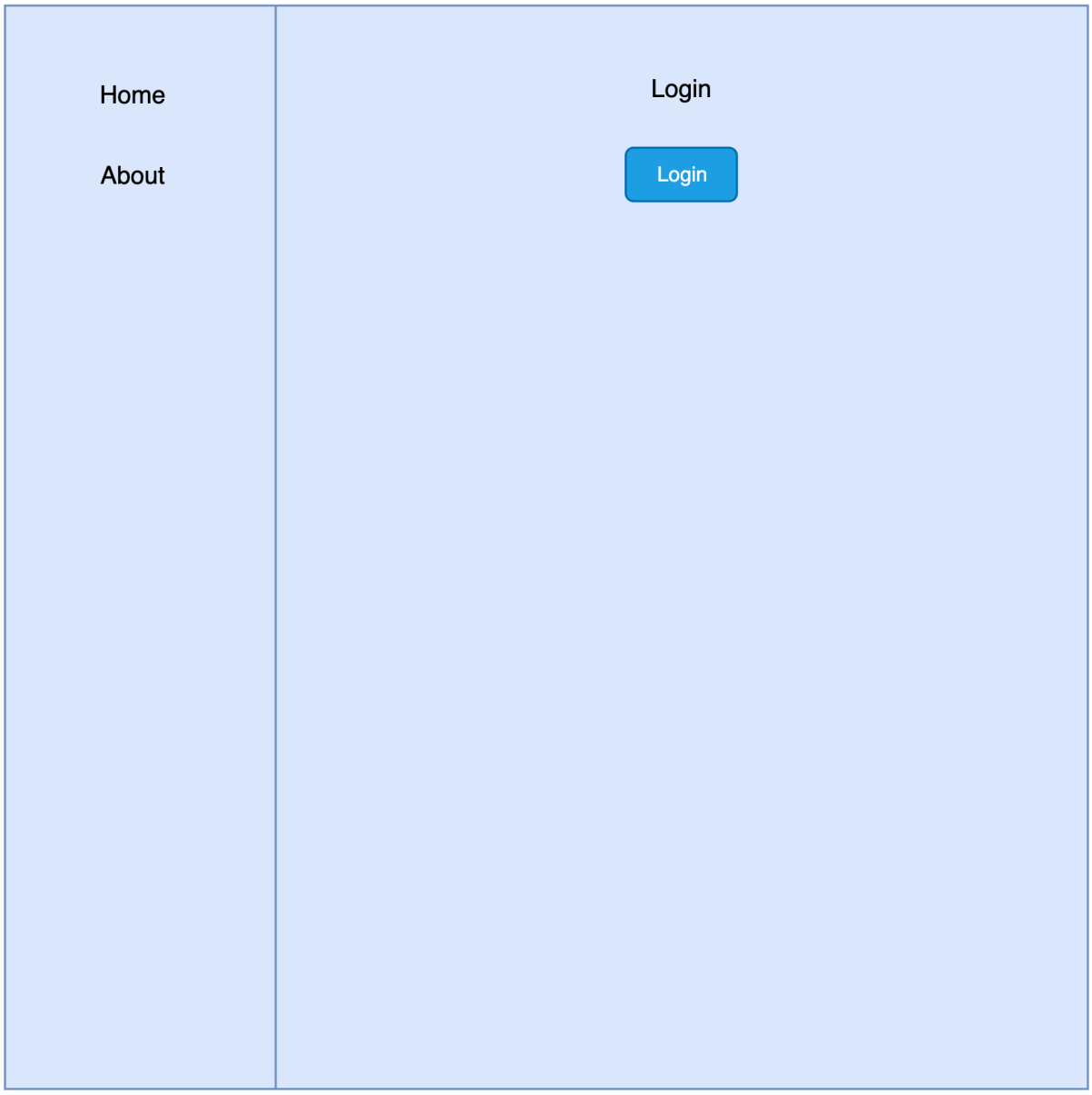


Рисунок 3.9 – SPA логін сторінка

Після того, як користувач натисне кнопку «Вхід», новий вміст буде представлено на тій же сторінці. На рисунку 3.10 показано QR-код, який відображається SPA після натискання кнопки входу.

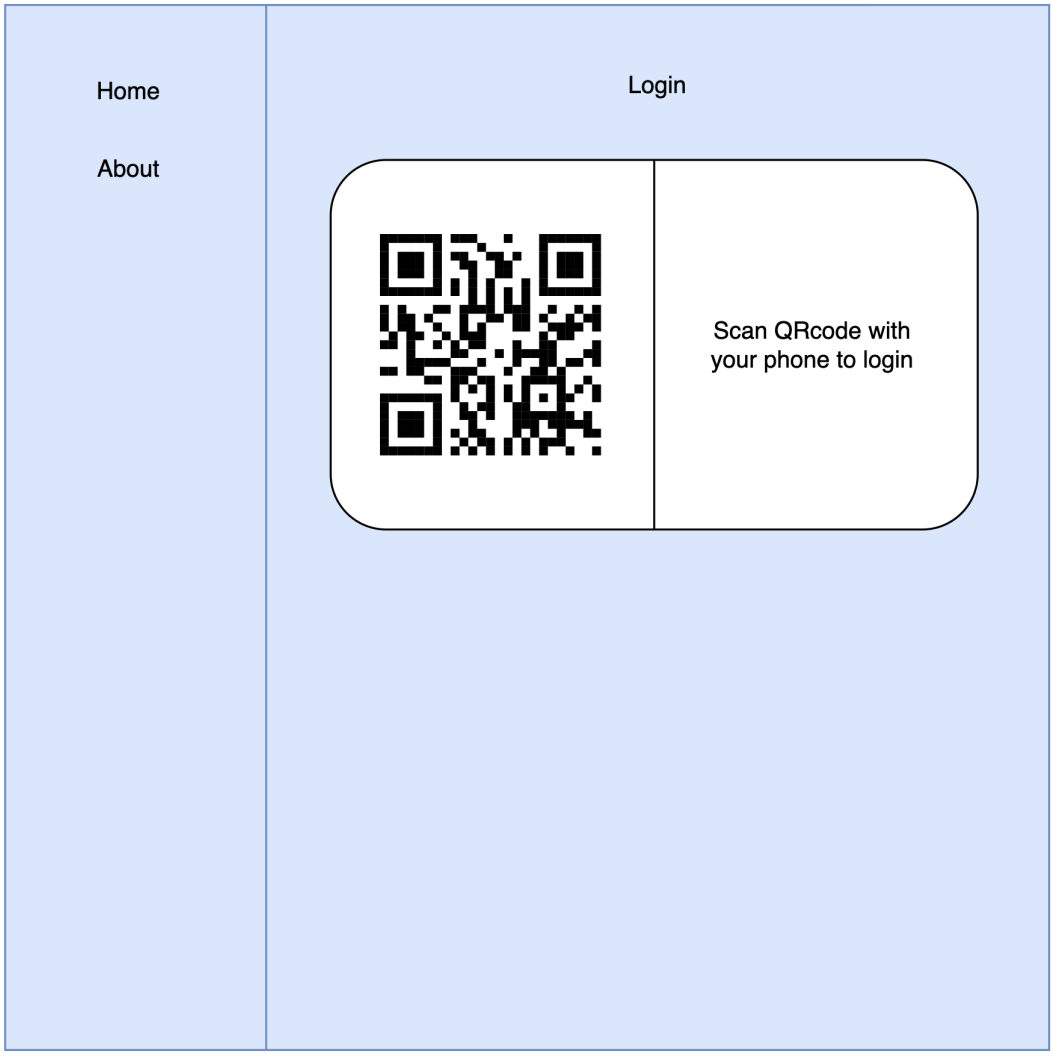


Рисунок 3.10 – Сторінка входу до SPA після того, як користувач натисне кнопку входу

Після успішної автентифікації користувача клієнту робочого столу видається JWT для авторизації, і користувач автоматично перенаправляється на домашню сторінку. На рисунку 3.11 зображено домашню сторінку SPA.

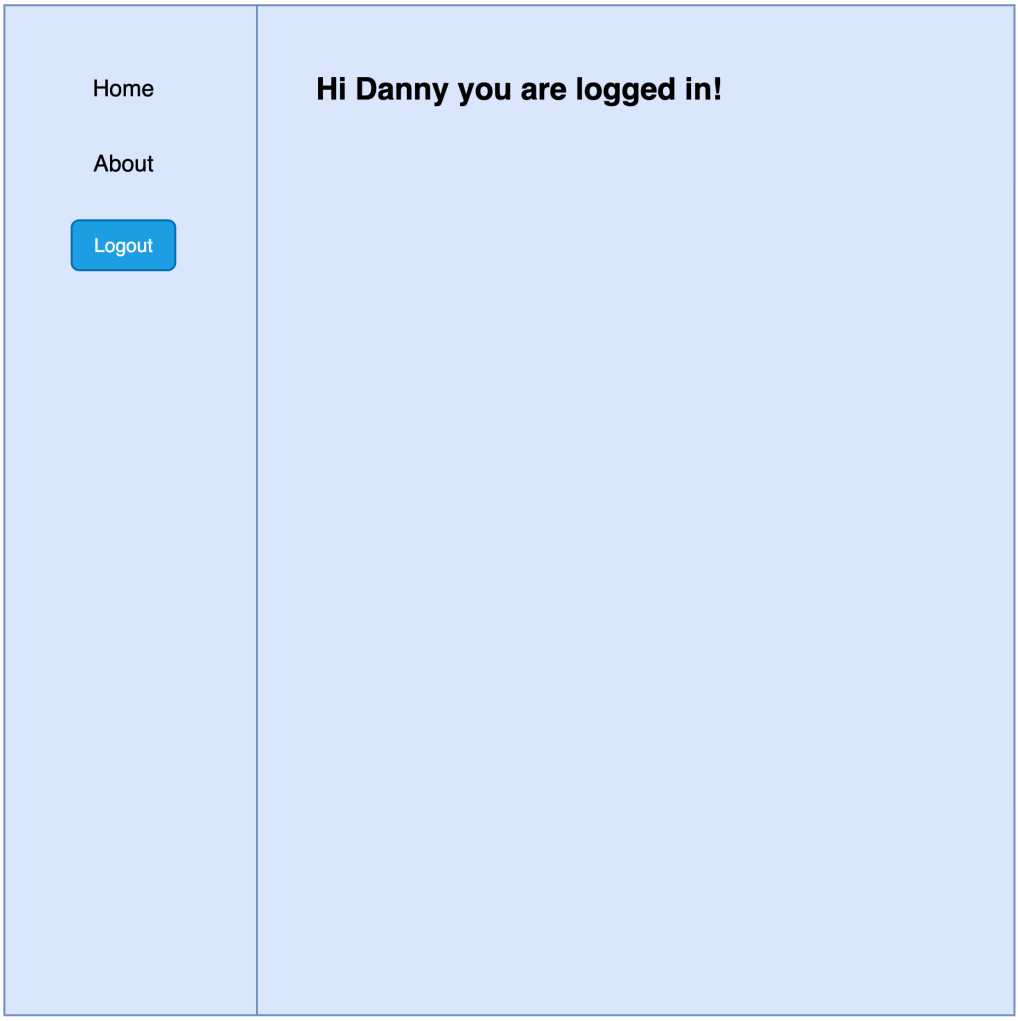


Рисунок 3.11 – Домашня сторінка SPA після аутентифікації користувача

Нарешті, у нижній частині меню надається посилання для виходу, щоб користувач міг вийти, використовуючи лише сайт. Після виходу користувач автоматично повертається на сторінку входу.

3.2.20 Мобільний інтерфейс користувача

Мобільний інтерфейс включає в себе кілька складних частин розробленої нативної програми. На рисунку 3.12 ілюструється простий екран привітання під час першого відкриття програми.

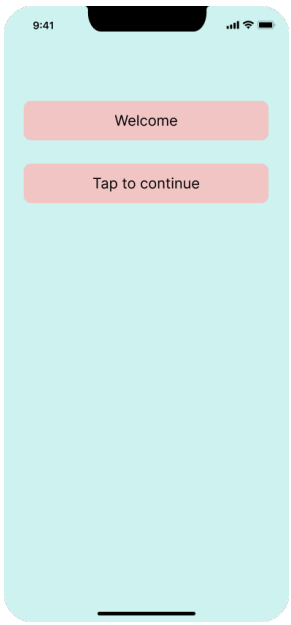


Рисунок 3.12 – Екран привітання в мобільному додатку

Після натискання на екран користувачеві буде запропоновано реєстраційну форму. На рисунку 3.13 зображено реєстраційну форму.

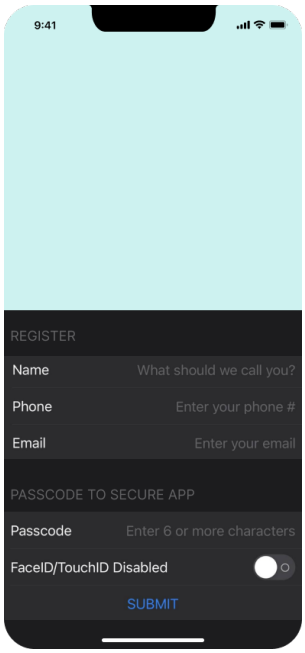


Рисунок 3.13 – Екран реєстрації мобільного додатку

Форма дає вказівки користувачам, підсвічуючи червоним кольором помилки введення. На рисунку 3.14 показано реєстраційну форму.

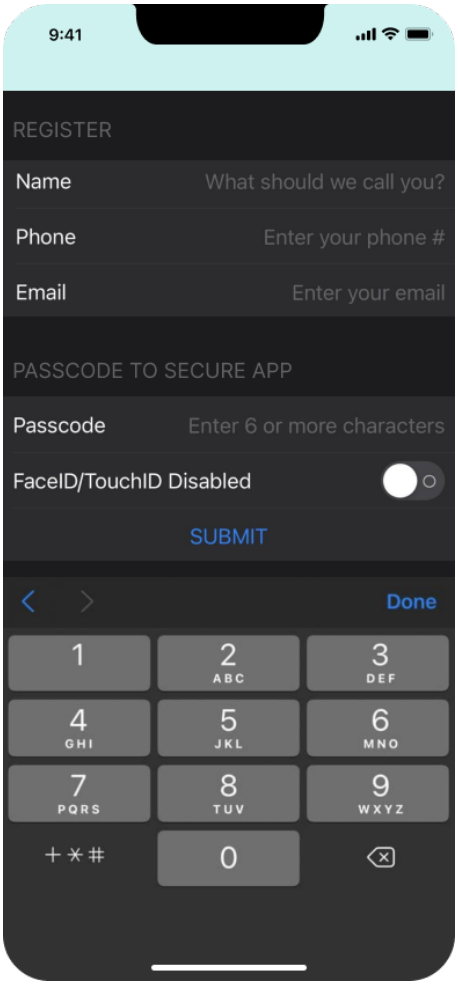


Рисунок 3.14 – Форма реєстрації мобільного додатку

Реєстраційна форма вимагає: ім’я, унікальну електронну адресу, унікальний номер телефону та пароль із шести або більше цифр. Коли форма заповнена, можна натиснути кнопку Надіслати. Після натискання кнопки «Надіслати» на пристрої створюються пари ключів користувача, а відкритий ключ і інформація про користувача безпечно передаються на сервер. Цей процес передбачає коротку затримку, і користувачеві відображається екран завантаження, поки цей процес завершується.

Рисунок 3.15 ілюструє екран завантаження, показаний користувачеві.

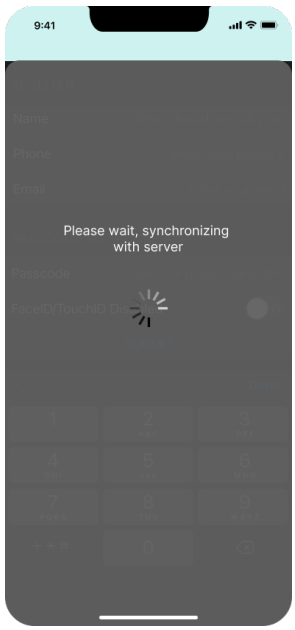


Рисунок 3.15 – Екран завантаження мобільного додатку

У разі помилки надсилання цієї інформації на сервер користувачеві відображається сповіщення та надається можливість повторити спробу. На рисунку 3.16 показано сповіщення, яке відображається у випадку невдалого завантаження.

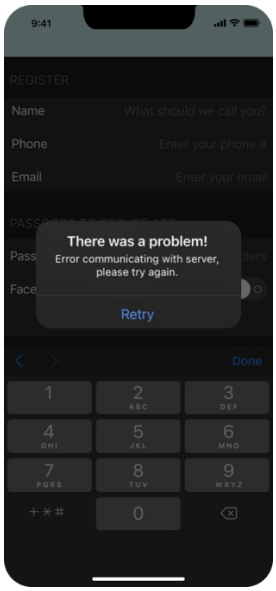


Рисунок 3.16 – Сповіщення про помилку завантаження мобільної програми

Після успішного завантаження інформації користувача на сервер користувач перенаправляється на головний екран. На рисунку 3.17 показано екран домашньої камери.

Рисунок 3.17 – Екран перегляду домашньої камери для мобільних пристроїв

Головний екран – це вид камери з навігаційною панеллю внизу, що дозволяє отримати доступ до екрана налаштувань. Екран «Параметри» в основному містить дані, які використовуються для тестування продуктивності, і містить кнопку, яка дозволяє видалити всі дані користувача. На рисунку 3.18 показано екран налаштувань.

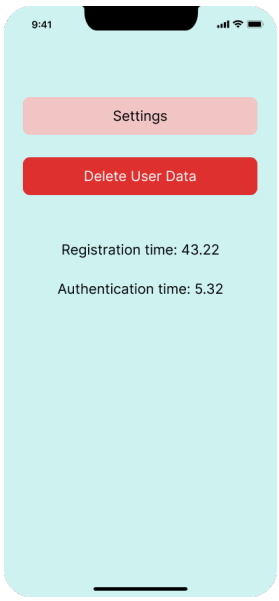


Рисунок 3.18 – Екран налаштувань мобільного додатка

На головному екрані користувач сканує QR-код, згенерований веб-сайтом SPA. Після успішної аутентифікації користувач перенаправляється на інформаційну панель, де користувач може переглянути свою інформацію та вийти, натиснувши кнопку «Вийти». На рисунку 3.19 показана інформаційна панель користувача.

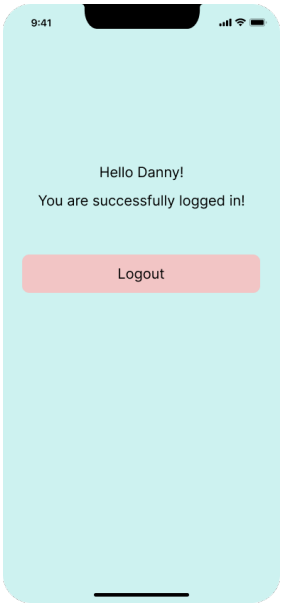


Рисунок 3.19 – Інформаційна панель мобільного додатку

Після успішного виходу користувач повертається на головний екран і відображається сповіщення про вихід, яке нагадує користувачеві закрити браузер, щоб завершити процес виходу. На рисунку 3.20 показано сповіщення, яке відображається після успішного виходу.

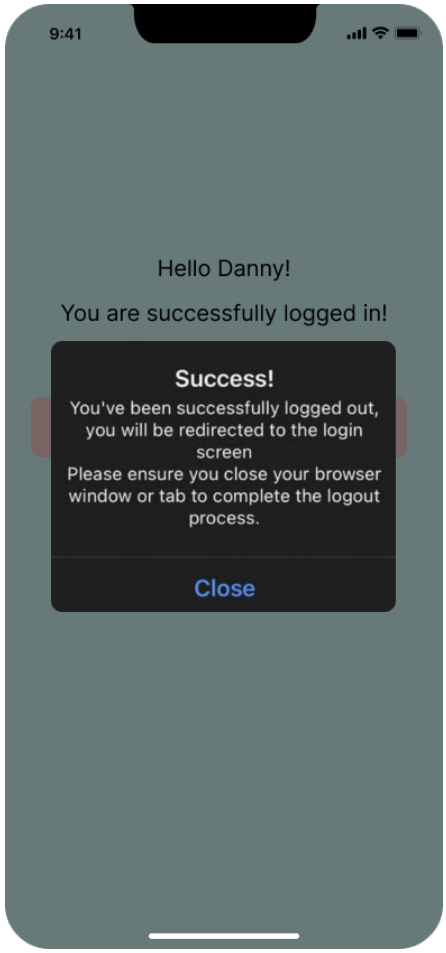


Рисунок 3.20 – Сповіщення про вихід з мобільного додатку

Якщо користувач виходить із програми або виходить із неї, програма негайно блокується, і для повторного входу потрібно ввійти в програму. Якщо під час реєстрації користувач вирішив увімкнути біометричну аутентифікацію, ініціюється біометричний вхід. Якщо користувач відмовився від біометричної аутентифікації, скасовує біометричний вхід або біометричний вхід не вдається, тоді користувач може ввійти за допомогою стандартного пароля. На рисунку 3.21 зображено екран входу

під час виконання біометричної автентифікації.

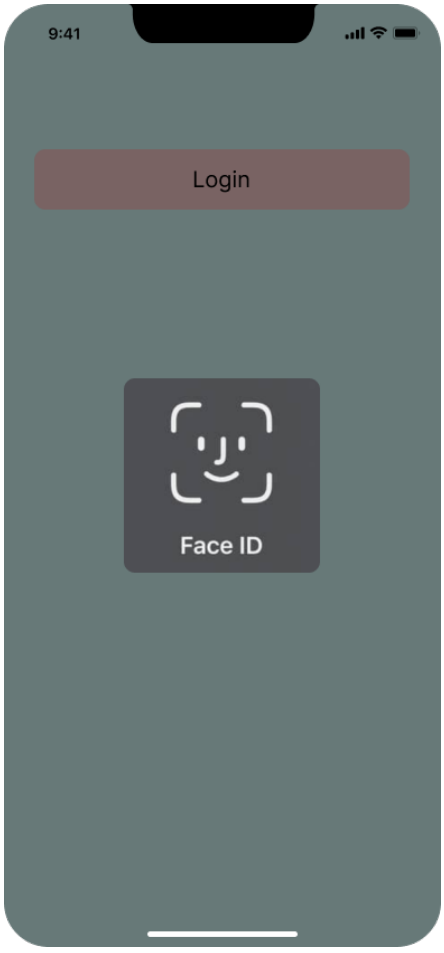


Рисунок 3.21 –Біометричний вхід в мобільний додаток

3.3 Програмне забезпечення

3.3.1 WebStorm

В якості редактора для написання коду було вибране таке інтегроване середовище розробки як WebStorm.

WebStorm – інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA. WebStorm є спеціалізованою версією PhpStorm, пропонуючи підмножину з його можливостей. WebStorm постачається з перед-установленим плагінами

JavaScript (такими як для Node.js), котрі доступні для PhpStorm безкоштовно.

WebStorm підтримує мови JavaScript, CoffeeScript, TypeScript та Dart.

WebStorm забезпечує автодоповнення, аналіз коду на льоту, навігацію по коду, рефакторинг, зневадження та інтеграцію з системами управління версіями. Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (у тому числі, рефакторинг коду JavaScript, що міститься в різних файлах і стеках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (коли в документ на HTML вкладений скрипт на Javascript, в який вкладено інший код HTML, всередині якого вкладений Javascript) — в таких конструкціях підтримується коректний рефакторинг.

Основні можливості:

– модифікація файлів .css, .html, .js з одночасним переглядом результатів (в деяких джерелах ця функціональність називається «редагування файлів на льоту» або «в реальному часі» або «без перезавантаження сторінки»);

– підтримка HTML5, Node.js, JS;

– можливості Zen Coding і Emmet;

– налагодження коду на JavaScript;

– віддалене розгортання по протоколах FTP, SFTP, на монтованих мережевих дисках і т. Д. З можливістю автоматичної синхронізації;

– інтеграція з системами управління версіями: Subversion, Git, GitHub, Perforce, Mercurial, CVS підтримуються з коробки з можливістю створення списків змін і відкладених змін;

– інтеграція з системами відстеження за помилками.

На рисунку 3.22 зображено інтерфейс редактору WebStorm.

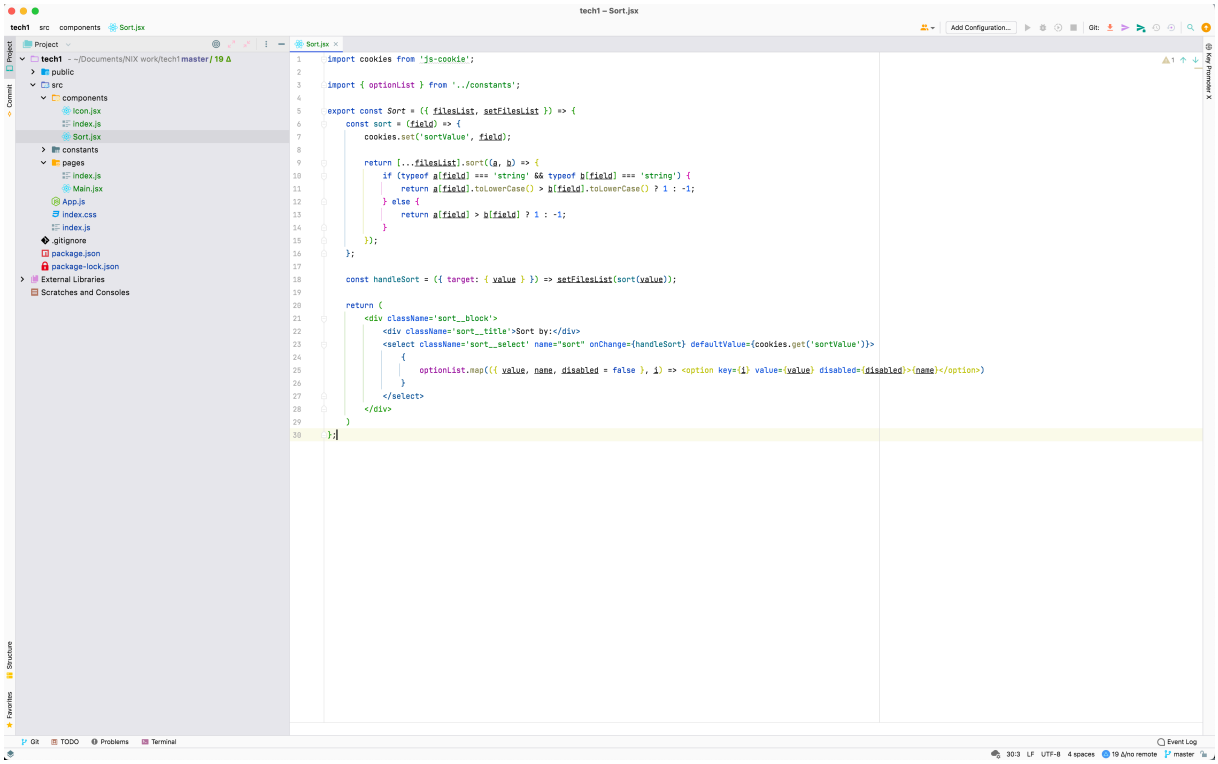


Рисунок 3.22 – Інтерфейс редактору WebStorm

3.3.2 Figma та Photoshop

Макет сайту – це практично весь дизайн. Він визначає порядок всіх елементів ресурсу, колірні відтінки, форми, конкретне розташування компонентів по відношенню один до одного.

На початковій стадії створюється каркас веб-сайту, відомий як схема сторінки або екрана. Він включає в себе елементи, які ви плануєте розмістити на ресурсі, і їх розташування.

Наступний етап – макет веб-сторінки. Визначається стиль, колірне рішення, форма і дизайн всіх елементів. Розробляючи макет, можна уявити, наскільки ергономічним буде проект, а також зовнішній вигляд сторінки в цілому.

Прототип – кінцевий інтерактивний продукт, з яким можна детально ознайомитися. Простіше кажучи, це готова робота, що володіє певним набором функцій.

Зрозуміло, коли створюєте продукт, початковий етап дуже важливий. Прототип робиться за бажанням. Макет веб-дизайну розробляється в обов'язковому порядку.

Щоб правильно оформити макет сайту для верстки, важливо знати його головне призначення, яку програму для його розробки використовувати і вимоги до зовнішнього вигляду.

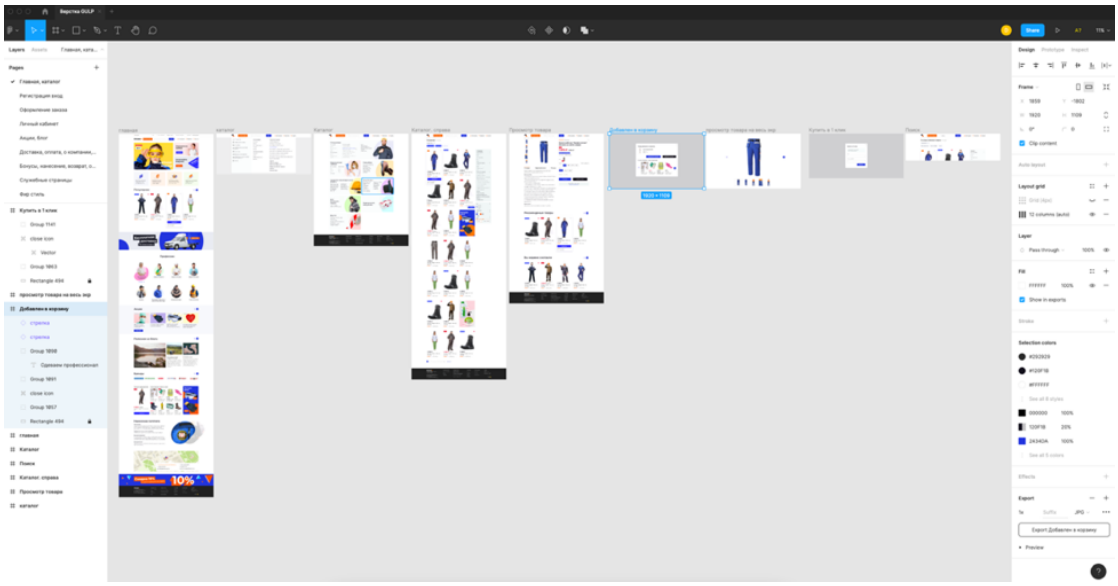
Для своєї роботи я використовував сервіс Figma.

Figma – векторний онлайн-сервіс розробки інтерфейсів та прототипування з можливістю організації спільної роботи, що розробляється однойменною компанією. Працює у двох форматах: у браузері та як клієнтський додаток на десктопі користувача. Зберігає онлайн-версії файлів, з якими працював користувач. Є безкоштовним для індивідуальних користувачів і платним для фахових команд.

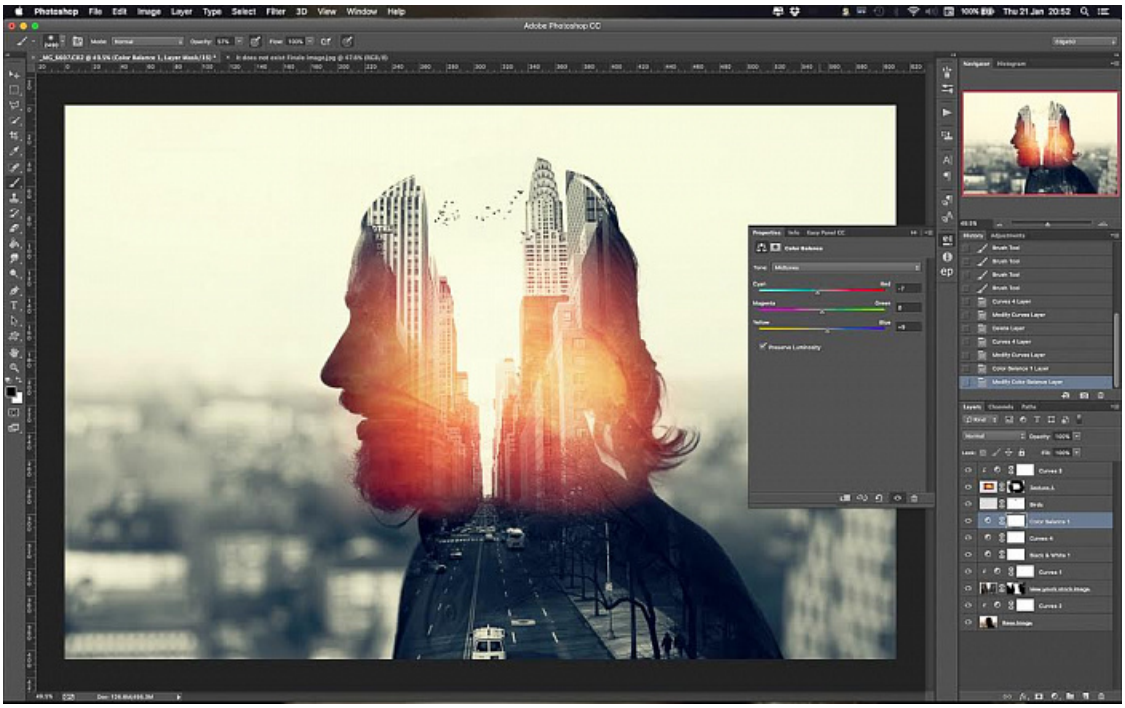
Даний редактор підходить як для створення простих прототипів і дизайн- систем, так і складних проектів (мобільні додатки, портали). Платформа стала одним із тих інструментів для розробників і дизайнерів, що найбільш швидко розвиваються. Стійка модель розвитку Figma, спростила співробітництво в усьому процесі створення цифрових продуктів для дизайнерів, розробників, менеджерів і маркетологів.

Також для обробки фотографій використовується такий сервіс як Photoshop. Photoshop – графічний редактор, розроблений і поширюваний фірмою Adobe Systems. Photoshop призначений для редагування цифрових фотографій та створення растрової графіки. Особливості Adobe Photoshop полягають у багатому інструментарії для операції створення і обробки зображень, високій якості обробки графічних зображень, зручності й простоті в експлуатації, широких можливостях до автоматизації обробки растрових зображень, які базуються на використанні сценаріїв, механізмах роботи з кольоровими профілями, які допускають їх втілення в файли зображень з метою автоматичної корекції кольорових параметрів при виводі на друк для різних пристроїв, великому наборі команд фільтрації, за

допомогою яких можна створювати найрізноманітніші художні ефекти. На рисунку 3.23 зображено інтерфейс редактору Figma, а на рисунку 3.24 – інтерфейс редактору Photoshop.



На рисунку 3.23 зображено інтерфейс редактору Figma.



На рисунку 3.24 зображено інтерфейс редактору Photoshop.

4 РЕЗУЛЬТАТИ РОЗРОБЛЕНОЇ ПРОГРАМИ

Ефективність аутентифікації без пароля оцінювалася за допомогою рандомізованих даних користувача, створених PHP Faker, що оцінює теоретичний час реєстрації користувача та повторює тестування часу аутентифікації для кількох випадкових користувачів. Час реєстрації та аутентифікації повідомляється на екрані налаштувань у мобільному додатку. Були розглянуті різні атаки та їхні можливі наслідки для безпеки.

4.1 Час реєстрації

Було створено двадцять випадкових користувачів, і їх ім’я, електронну пошту, телефон і коди доступу від 6 до 9 цифр були введені в мобільний додаток. Потім ця інформація була відправлена на сервер. Загальний час реєстрації, включаючи час введення інформації про користувача та надсилання інформації на сервер, було розраховано та усереднено разом. Середній час реєстрації користувача становив 38,20 секунди. Медіана становила 37,054 секунди. Хоча ці результати не є повною ознакою досвіду реєстрації середнього користувача, вони надають деяку базову інформацію про те, що середній користувач повинен мати змогу завершити реєстрацію за одну хвилину або менше.

4.2 Час аутентифікації

Використовуючи п’ять змодельованих користувачів, створених вище, вибраних навмання, аутентифікація була виконана десять разів для кожного користувача. Загальний час аутентифікації обчислювався на початку сканування QR-коду, закінчуючись JWT, отриманим мобільним пристроєм. Часи були усереднені для кожного користувача. Це зображено на рисунку 4.1.

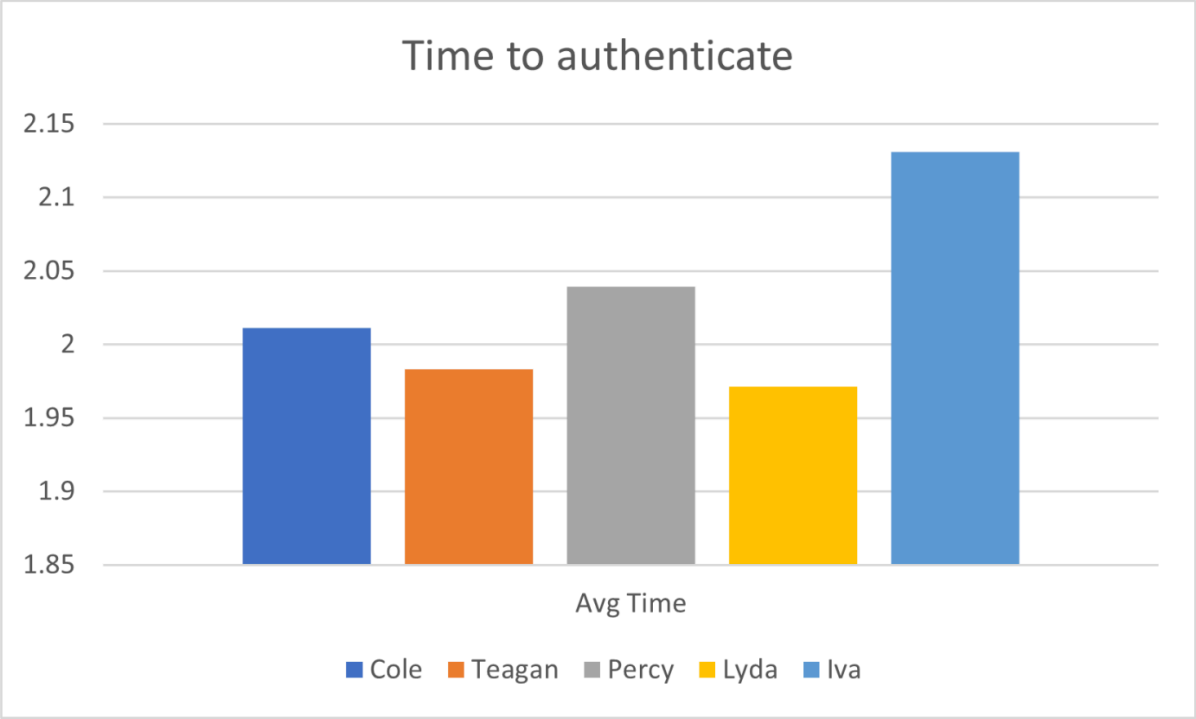


Рисунок 4.1 – Середній час аутентифікації для кожного користувача

Середнє значення для всіх аутентифікацій для всіх користувачів становило 2,027 секунди. Знову ж таки, це не обов'язково вказує на досвід автентифікації середнього користувача, але надає базову інформацію про те, що для середнього користувача автентифікація повинна тривати менше п'яти секунд.

4.3 Безпека сервера

Для захисту підключення до сервера було використано кілька засобів, включаючи TLS, захист маршрутів, проміжне програмне забезпечення та захист CSRF. Цей проект не міг захистити від усіх напрямків потенційної загрози, оскільки багато потенційних векторів атак вийшли за межі його дії. Однак основними загрозами вважаються атаки з відтворенням у часі та XSS.

4.3.1 Атака Timed Replay

Timed Replay атака – це атака на систему аутентифікації шляхом запису та подальшого відтворення раніше надісланих коректних повідомлень або їх частин.

Оскільки процес авторизації покладається на одноразовий код, доступний за допомогою QR-коду, який легко сканувати та потенційно вразливий до атаки через плече, зловмисник може просканувати код іншого користувача та отримати одноразовий код користувача під час входу. Вікно TTL кешу для отримання JWT дуже коротке, дві секунди, і nonce видаляється з кешу після використання. Однак, за точного часу зловмисник може дочекатися автентифікації користувача та надіслати запит на публікацію до кінцевої точки API /login/confirm за секунду до того, як клієнт користувача отримає JWT користувача. Тестування підтвердило, що це потенційна вразливість. Використовуючи Postman (клієнт API Representational State Transfer (REST)) і точно визначивши час запиту POST, вдалося отримати JWT іншого користувача. Для цього знадобилося кілька спроб, і хоча це цілком можливо, на практиці це малоймовірно, оскільки умови, необхідні для використання цієї вразливості, вимагали такої точності. Необхідні умови: копія nonce користувача, що автентифікується, яку потрібно було вручну скопіювати в Postman перед автентифікацією користувача. Він також вимагав синхронізації запиту на публікацію, щоб він надсилався після автентифікації користувача, але до того, як браузер користувача зробить окремий запит POST. Тим не менш, користувачі повинні бути в курсі свого оточення та остерігатися людей, які, схоже, намагаються сфотографувати його чи її екран. Користувачі повинні оновити сторінку, щоб отримати новий nonce, якщо вони вважають, що інший користувач намагався відсканувати або захопити їхній QR-код. Майбутню роботу слід виконати, щоб забезпечити додатковий захист штрих-коду, щоб запобігти скануванню через плече.

4.3.2 Атака Cross Site Scripting

Cross Site Scripting – це тип атаки на веб-системи, що полягає у впровадженні на сторінку шкідливого коду (який буде виконаний на комп'ютері користувача при відкритті ним цієї сторінки) і взаємодії цього коду з веб-сервером зловмисника.

Оскільки JWT зберігається в локальному сховищі браузера, що дозволяє отримати до нього доступ за допомогою JavaScript, він потенційно вразливий до атак XSS. Незважаючи на те, що ця атака вийшла за межі об’єму, і її не вдалося виконати під час тестування, одним із можливих рішень є збереження JWT у безпечному файлі cookie з прапорцем HTTPOnly, який не дозволяє доступ JavaScript. На жаль, це також ускладнює використання JWT у SPA, оскільки зовнішній клієнт не зможе отримати доступ до JWT. Файли cookie також потенційно вразливі до атак CSRF і вимагають додаткових заходів, щоб гарантувати, що інший домен не матиме доступу до захищених файлів cookie.

4.4 Безпека програми

Оскільки досконала безпека та зручність зазвичай виключають одне одного, їх потрібно збалансувати на користь зручності використання. Доступ до пристрою та програми дозволяє повністю аутентифікувати користувача, тому вважається високим ризиком для безпеки, якщо пристрій скомпрометовано. Хоча сам закритий ключ можна використовувати для ідентифікації та автентифікації користувача, доступ до закритого ключа малоймовірний, оскільки він зберігається в Secure Enclave. Тому сама програма вважається найбільш вразливим вектором безпеки. Було зроблено кілька кроків, щоб забезпечити додатковий захист програми та закритого ключа користувача. Сама програма вимагає входу, а закритий ключ потребує біометричної автентифікації, щоб розблокувати його для підпису.

Це вимагає від зловмисника не тільки отримати доступ до програми, але й мати унікальний біометричний аутентифікатор користувача.

Оскільки користувачі можуть володіти своїм мобільним пристроєм із будь-якою аутентифікацією, до програми додано додатковий етап аутентифікації. Основною метою цього проекту було дозволити вхід повністю без пароля, і цей проект передбачав, що більшість користувачів виберуть біометричну аутентифікацію. Однак стандартний PIN-код був наданий для застарілих цілей. Також було надано стандартний пароль на випадок, якщо біометрична аутентифікація не вдасться або буде недоступна. Було вирішено використовувати простий цифровий пароль, тому що це аутентифікація за умовчанням на багатьох пристроях і тому, що це був найпростіший метод для реалізації. Для того, щоб забезпечити більшу зону безпеки, ніж стандартний чотиризначний PIN-код, було вирішено ввести мінімальний шість цифр без максимального значення. Для того, щоб зловмисник отримав доступ до програми, не знаючи коду доступу чи володіння біометричним аутентифікатором, потрібна або атака грубою силою, або атака за словником. Хоча можливі й інші атаки, вони виходять за рамки.

4.4.1 Атака Brute-Force

Brute-Force – це атака зламування пароля шляхом перебору всіх можливих варіантів ключа.

Атака методом грубої сили просто намагається змінити всі потенційні паролі, поки не буде отримано доступ або зловмисник не відмовиться. У разі мінімального шестизначного числа зловмисник, який має доступ до пристрою та програми, розпочне з 000000, перейде до 000001... і т.д., спробувавши всі можливі комбінації паролів до 999999. Оскільки багато користувачів відповідатимуть лише мінімальним вимогам безпеки, напад такого характеру, ймовірно, зрештою вдасться. Таким чином, мета полягає

в тому, щоб зробити робочий фактор атаки достатньо дорогим, щоб зловмисник здався передчасно, або дати власнику пристрою достатньо часу, щоб відновити доступ до пристрою, віддалено стерти пристрій або змінити закритий ключ. Нижче наведено аналіз зони атаки та робочий фактор, необхідний для цієї атаки.

Оскільки повторювані цифри не застосовуються, користувач може вибрати мінімальну кількість цифр і встановити пароль до шести повторюваних цифр (наприклад, 000000). Якщо припустити, що користувач не має аутентифікації на пристрої, найкращим варіантом атаки буде отримання зловмисником негайного доступу до програми за допомогою шести повторюваних нулів. Зловмисник може евристично спробувати будь-яку іншу комбінацію повторюваних цифр і отримати доступ менш ніж за хвилину. Інші евристичні атаки, такі як спроби лінійного прогресування паролів, 1-6, 2-7, 3-8 тощо, також можуть використовуватися, але виходять за межі цього проекту. Атака в найгіршому випадку буде методичною спробою кожної іншої перестановки P з десяти цифр n з усіх можливих шестизначних комбінацій у порядку r, вираженому як $P = n^r = 10^6$.

Як згадувалося раніше, паролі хешуються за допомогою bcrypt із робочим фактором у дві секунди. Тоді для зловмисника з простором для атаки 10^6 можливих перестановок паролів дає загальний коефіцієнт роботи $2s \times 10^6$ або приблизно двадцять три дні. Це короткий проміжок часу з огляду на вимоги до паролів для загальнодоступних серверів, де злам паролю може не виявлятися протягом тривалого періоду часу. Проте двадцяти трьох днів має бути достатньо, щоб користувач усвідомив, що його пристрій зламано, і вжив відповідних заходів.

4.4.2 Dictionary attack

Dictionary – це атака на систему захисту, що використовує метод повного перебору передбачуваних паролів, що використовуються для

аутентифікації, що здійснюється шляхом послідовного перегляду всіх слів (паролей у чистому вигляді або їх зашифрованих образів) певного виду та довжини зі словника з метою подальшого злому системи та отримання доступу до секретної інформації.

Атака за словником або більш формально, атака за попередньо обчисленим словником – це метод, за допомогою якого зловмисник може отримати доступ до бази даних із зашифрованими паролями. Зловмисник створює словник усіх можливих хешованих значень і зіставляє їх із нехешованим еквівалентом. Потім зловмисник просто шукає хешований пароль із викраденої бази даних до нехешованого значення в словнику. Однак, оскільки використовувався bcrypt, який хешує пароль із випадково згенерованою солі, цей словник потрібно створити заново за допомогою bcrypt і значення солі для всіх можливих перестановок пароля.

Фактор роботи важко визначити, оскільки припускається, що пристрій не використовує жодної автентифікації, що дозволяє зловмиснику завантажити вміст keychain пристрою на свій комп'ютер. Це означає, що дві секунди робочого фактора на пристрої можуть бути набагато меншими на машині зловмисника. Крім того, коли враховується підхід грубої сили, порядок коду доступу має значення. Однак, оскільки хешування повторюваних цифр r, наприклад 111111, завжди повертатиме той самий хешований результат, кількість можливих десятизначних n хешованих комбінацій паролів P значно зменшується. Це можна виразити як

$$P(n,r) = \frac{n!}{(n-r)!}, \tag{4.1}$$

$$P(10,6) = \frac{10!}{(10-6)!} = \frac{10!}{6!} = 151{,}200 \text{ різних комбінацій}, \tag{4.2}$$

що значно менше, ніж 106 можливих комбінацій, виражених вище.

Тестування простої реалізації bcrypt у Node.js на локальній машині з Windows 10 із використанням шість округленого значення, яке є таким самим, як і на пристрої, дало робочий фактор приблизно 5 мс (4,73 мс). Загальний коефіцієнт роботи, необхідний для створення всіх можливих хешованих комбінацій, можна виразити як $5ms \times 151{,}200 \approx 12.6$ minutes.

На жаль, навіть щоб отримати рекомендований робочий коефіцієнт принаймні 241 мс на комп'ютері з Windows 10, потрібно було округлити дванадцять, що вимагало надзвичайно дорогого робочого коефіцієнта понад шість хвилин (400 секунд) на мобільному пристрої. Це означає, що сама програма є винною в цьому типі атаки, припускаючи, що пристрій не захищено жодним видом автентифікації. Якщо пристрій захищено автентифікацією, то сам брелок пристрою зашифрований, що значно ускладнює доступ. Однак, навіть якщо зловмисник отримує доступ до програми, сам приватний ключ все ще захищений у Secure Enclave, і для створення підписів із приватним ключем все одно потрібен біометричний аутентифікатор, щоб розблокувати його.

ВИСНОВКИ

В цій кваліфікаційній роботі ми виконали свої основні цілі. Ми використали існуючі, перевірені системи криптографії, поєднавши їх у новий спосіб. Також вдалося створити робочий прототип системи MFA без пароля, який є ефективним, простим і простим у використанні. Імітована реєстрація користувача передбачає, що реєстрацію можна завершити за одну хвилину або менше. Повторні симульовані тести аутентифікації показують, що аутентифікацію можна виконати менше ніж за п’ять секунд.

Це контрастує з архаїчною та багатоскладною системою паролів. 2FA пропонується як рішення, але додає додаткової складності. SMS 2FA вимагає надсилання додаткових повідомлень, які можна перехопити. TOTP забезпечує високий рівень додаткової безпеки, але потребує окремої програми або додаткового апаратного забезпечення та вимагає, щоб секретне початкове число зберігалося у відкритому вигляді на сервері, який став метою серйозного порушення національної оборони.

Асиметрична криптографія може забезпечити високий рівень безпеки за умови, що закритий ключ користувача ніколи не розкривається. Вбудовані апаратні системи, як Secure Enclave, на сучасних мобільних пристроях можуть забезпечити безпечний доступ до закритих ключів, одночасно запобігаючи їхньому розкриттю, якщо пристрій скомпрометовано. Біометрична аутентифікація може забезпечити ефективну та точну ідентифікацію без використання паролів. Комбінація біометричної аутентифікації та цифрових підписів може забезпечити безпарольну систему MFA, забезпечуючи набагато кращий досвід користувача, ніж традиційний підхід із паролем.

Вже проведено багато досліджень щодо нових і кращих систем аутентифікації. В цій кваліфікаційній роботі ми показали, що ще потрібно багато працювати, щоб випередити зловмисників. Ми використовували QR-коди через просту реалізацію, але можна було створити штрих-коди, які

дозволено сканувати лише авторизованим користувачам. Більш безпечні QR-коди, такі як SQRC від DENSO WAVE Incorporated, можуть бути реалізовані для запобігання атакам через плече та повторним повторам. Зашифровані файли cookie, які містять стійкий до CSRF маркер, можуть захистити від атак XSS і CSRF, а також доступні для фреймворків JavaScript. Система авторизації для ідентифікації авторизованих пристроїв після аутентифікації забезпечить більш безпечну та плавну передачу пристрою.

Цей проект використовував штрих-коди для аутентифікації, але можливі досягнення NearField Communication, подібні до того, що зараз робиться для безконтактних платіжних систем, можуть бути використані для простої та ефективної аутентифікації. Хоча Bluetooth є дещо небезпечним, він також може надати інший шлях для меншої, більш ефективної системи шифрування аутентифікації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Department of Defense. Dod instruction 8520.03, identity authentication for information systems. 2011.

2. National Institute of Standards and Technology (U.S.). Guideline for the use of advanced authentication technology alternatives. Gaithersburg, MD : Computer Systems Laboratory, National Institute of Standards and Technology, U.S. Dept. of Commerce, Technology Administration, 1994. 58 с.

3. A M. G. The magical number seven, plus or minus two: Some limits on our capacity for processing information. Washington D.C : Psychological review, 1956. 81 с.

4. D. V. Klein. Foiling the cracker: A survey of, and improvements to, password security. 1990.

5. A. Vance, «If your password is 123456, just make it hackme» The New York Times, Jan. 2010. https://www.nytimes.com/2010/01/21/technology/21password.html (дата звернення: 12.04.2023)

6. W. Cheswick, «Rethinking passwords» Commun. ACM, Feb. 2013. https://doi.org/10.1145/2408776.2408790 (дата звернення: 17.04.2023).

7. A. Adams and M. A. Sasse, «Users are not the enemy» Commun. ACM, с. 40–46, Dec. 1999. https: //doi.org/10.1145/322796.322806 (дата звернення: 24.04.2023).

8. R. P. Jover, «Security analysis of sms as a second factor of authentication: The challenges of multifactor authentication based on sms, including cellular security deficiencies, ss7 exploits, and sim swapping». https://doi.org/10.1145/3424302.3425909 (дата звернення: 27.04.2023).

9. D. Kogan, N. Manohar, and D. Boneh, «T/key: Second-factor authentication from secure hash chains» in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 983–999. https://doi.org/10.1145/3133956.3133989 (дата звернення: 7.04.2023).

10. C. Drew, «Security firm offers to replace tokens after attack» The New York Times, Jun. 2011. https://www.nytimes.com/2011/06/07/ technology/07hack.html? r=1 (дата звернення: 01.05.2023).

11. Yubico. (2021, Apr) Yubikey 5 series. Yubico. https: //www.yubico.com/products/yubikey-5-overview/ (дата звернення: 03.04.2023).

12. C. Jacomme and S. Kremer, «An extensive formal analysis of multi-factor authentication protocols» ACM Trans. Priv. Secur., vol. 24, no. 2, Jan. 2021. https://doi-org.ezproxy.library.ewu.edu/10.1145/3440712 (дата звернення: 24.04.2023).

13. M. A. Sasse, S. Brostoff, and D. Weirich, «Transforming the weakest link — a human/computer interaction approach to usable and effective security» BT Technology Journal, vol. 19, no. 3, pp. 122–131, Jul. 2001. http://dx.doi.org/10.1023/A:1011902718709 (дата звернення: 24.04.2023).

14. W. S. D. of Licensing. Steps to getting your first driver license: Proof of identity. Washington State Department of Licensing. https://www.dol.wa.gov/driverslicense/idproof.html (дата звернення: 25.04.2023).

15. J. Steven, J. Walton, and K. Wall, «Owasp threat model for secure password storage» Open Web Application Security Project, Tech. Rep., Jul. 2012. https://owasp.org/www-pdf-archive/Secure Password Storage.pdf (дата звернення: 25.04.2023).

16. L. Johnson, «Chapter 11 - security component fundamentals for assessment» in Security Controls Evaluation, Testing, and Assessment Handbook (Second Edition), 2nd ed., L. Johnson, Ed. Academic Press, 2020, pp. 471– 536. https://www.sciencedirect.com/science/article/pii/ B9780128184271000112 (дата звернення: 17.04.2023).

17. A. Mirian, J. DeBlasio, S. Savage, G. M. Voelker, and K. Thomas, «Hack for hire: Exploring the emerging market for account hijacking» in The World Wide Web Conference, ser. WWW ’19. New York, NY, USA: Association

for Computing Machinery, 2019, p. 1279–1289. https://doi.org/10.1145/3308558.3313489 (дата звернення: 24.04.2023).

18. D. M'Raihi, S. Machani, M. Pei, and J. Rydell, «Totp: Time-based one-time password algorithm» Internet Requests for Comments, RFC Editor, RFC 6238, May 2011.

19. R. S. LLC. Securid® access. RSA Security LLC. https://www.rsa.com/en-us/products/rsa-securid-suite/rsa-securid-access (дата звернення: 27.04.2023).

20. M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, «Protection in operating systems» Commun. ACM, vol. 19, no. 8, p. 461–471, Aug. 1976. https://doi-org.ezproxy.library.ewu.edu/10.1145/360303.360333 (дата звернення: 25.04.2023).

21. T. Bray, «The javascript object notation (json) data interchange format» Internet Requests for Comments, RFC Editor, RFC 7159, Mar. 2014. https://tools.ietf.org/html/rfc7159 (дата звернення: 26.04.2023).

22. M. Jones, J. Bradley, and N. Sakimura, «Json web token (jwt)» Internet Requests for Comments, RFC Editor, RFC 7519, May 2015. https://tools.ietf.org/html/rfc7519 (дата звернення: 28.04.2023).

ДОДАТОК А

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	73 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Кутовий Д.О.			Розробка та дослідження методів аутентифікації користувачів в інформаційних системах	Шифр групи	Код напр./спец.
Перевірив		Філатов В.О.				СШІм-20-1	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					