НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«До захисту допущено»
В.о. завідувача кафедри

_____

М.В.Грайворонський
(підпис)

“_____”_____2019 р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.170101 «Безпека інформаційних і комунікаційних систем»

на тему: Моделювання методу безпарольної аутентифікації на основі протоколу WebAuthn

Виконав (-ла): студент (-ка) 4 курсу, групи ФБ-51
(шифр групи)

Козовий Віталій Мирославович (підпис)
(прізвище, ім’я, по батькові)

Керівник Орєхов О. А. (підпис)
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

Консультант (підпис)
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали)

Рецензент Савчук М.М. (підпис)
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ - 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.170101 «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____М.В.Грайворонський
(підпис)
«___»_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту

Козовому Віталію Мирославовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Моделювання методу безпарольної аутентифікації на основі протоколу WebAuthn,
науковий керівник роботи: Орєхов Олександр Арсенійович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «____ » 2019 р. № _____

2. Термін подання студентом роботи 10 червня 2019 р.

3. Вихідні дані до роботи

4. Зміст роботи

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

6. Дата видачі завдання 10.10.2019

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Отримання завдання	10.10.2019	
2	Збір інформації	01.02.2019	
3	Дослідження предметної області та існуючих рішень	10.04.2019	
4	Розробка плану роботи	15.04.2019	
5	Моделювання аутентифікації	10.05.2019	
6	Створення інтерфейсу програми	18.05.2019	
7	Оцінка результатів	23.05.2019	
8	Оформлення дипломної роботи	26.05.2019	

Студент _____ Козовий В.М.
(підпис) (ініціали, прізвище)

Науковий керівник роботи _____ Орєхов О.А.
(підпис) (ініціали, прізвище

РЕФЕРАТ

Дипломна робота має обсяг 68 сторінок, містить 11 рисунків, 1 таблицю та 15 бібліографічних джерел.

Моделювання зручних та безпечних методів безпарольної аутентифікації є неймовірно актуальним у сучасному світі. Ці методи допоможуть захистити користувачів від фішингу, програмних ботів та втрати паролів. Веб-сервіси та програми можуть і повинні використовувати нові технології, щоб дати користувачам можливість швидко та безпечно входити в систему за допомогою біометричних даних, мобільних пристроїв та / або ключів безпеки FIDO, а також з набагато більшою безпекою, ніж паролі.

Дана робота містить інформацію про сучасні методи аутентифікації та їх переваги . У ході роботи було змодельовано систему з можливістю реєстрації та аутентифікації, використовуючи API. В подальшому, методику можна застосувати для реєстрації та аутентифікації на веб-сайтах.

Ключові слова: аутентифікація, API, WebAuthn, FIDO, World Wide Web Consortium, Bluetooth, NFC.

ABSTRACT

Graduate work contains 68 pages, 11 illustrations, 1 table, 15 sources.

Simulation of convenient and safe methods of without password authentication is incredibly relevant in the modern world. These methods will help protect users from phishing, software bots and loss of passwords. Web services and applications can and must use new technologies to allow users to quickly and securely log in with biometric data, mobile devices and / or FIDO security keys, as well as much more security than passwords.

This work contains information on modern authentication methods and their benefits. In the course of work, a system with the ability to register and authenticate using the API was modeled. In the future, the technique can be applied for registration and authentication on websites.

Key words: Authentication, API, WebAuthn, FIDO, World Wide Web Consortium, Bluetooth, NFC.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

WebAuthn - Web authentication

UX - User experience

API - Application programming interface

W3C - World Wide Web Consortium

FIDO - Fast Identity Online

ARAE - Adaptive Risk Analysis Engine

U2F - Universal 2nd Factor

MFA - Multi factor authentication

ВСТУП

Найчастішою причину витоку даних в компаніях залишаються слабкі паролі. Вони є винуватцями більшості атак - 81% випадків за даними Verizon. Дослідження Yubico показали, що користувачі витрачають 10.9 годин на рік, вводячи або відновлюючи паролі, а це коштує компаніям в середньому 5.2 мільйонів доларів щорічно.

Коли традиційна багатофакторна аутентифікація (MFA) така, як одноразові SMS коди додають ще один прошарок захисту, вони все ще є вразливими до фішингових атак та не дуже зручні у користуванні.

WebAuthn - це веб-стандарт, опублікований консорціумом World Wide Web. WebAuthn є основним компонентом проекту FIDO2 під керівництвом Альянсу FIDO. Метою проекту є стандартизація інтерфейсу для аутентифікації користувачів до веб-додатків і послуг за допомогою криптографії публічного ключа.

Веб-сервіси та програми можуть і повинні вмикати цю функцію, щоб дати користувачам можливість легше входити в систему за допомогою біометричних даних, мобільних пристроїв та / або ключів безпеки FIDO, а також з набагато більшою безпекою, ніж паролі.

Актуальність роботи. Тема має провідну тенденцію у розвитку методів безпарольної аутентифікації. Ці рішення є сучасними, зручними, безпечними та економними для користувачів, розробників та компаній.

Мета і завдання дослідження. Метою роботи є моделювання та удосконалення методів безпарольної аутентифікації на основі протоколу WebAuthn. Створення зручного та простого у використанні для користувачів інтерфейсу програми.

Відповідно до вищесказаного, були поставлені наступні задачі:

1) Моделювання роботи реєстрації користувача з допомогою токена
2) Моделювання спроби аутентифікації користувача в систему;
3) Дослідження принципів роботи протоколу та API;
4) Розробка удосконаленого методу аутентифікації;
5) Створення інтерфейсу та UX

Об'єкт дослідження. Об'єктом дослідження є сучасні методи безпарольної аутентифікації на основі протоколу WebAuthn.

Предмет моделювання. Предметом моделювання є веб-сторінка з можливістю безпарольної реєстрації та аутентифікації.

Методи дослідження. Для моделювання роботи безпарольної аутентифікації були використані таблиці підтримки браузерів, WebAuthn API, PyWebAuthn та Flask**.**

Наукова новизна одержаних результатів. Моделювання нових методів безпарольної аутентифікації.

Практичне значення одержаних результатів. Практичною цінністю роботи є дослідження різних методів аутентифікації, роботи протоколу. Отримані результати допоможуть зробити роботу протоколу зручнішою та швидшою.

1 СУЧАСНЕ ПОНЯТТЯ АУТЕНТИФІКАЦІЇ

Більшість сучасних систем захисту паролів для Інтернету є недосконалими. Важливість створення надійних паролів забивається в нас щотижня. Ми відвідуємо веб-сайти, які мають форми реєстрації, які повідомляють вам, наскільки захищений вибраний пароль. Ми чуємо про це у новинах. Ми бачимо це в електронних листах, які ми отримуємо кожні кілька місяців, нагадуючи нам про оновлення паролів.

Отже, ви вже подумали б, що більшість людей використовують кілька сильних паролів, чи не так? На жаль, це не так, і цілком імовірно, що небезпечні паролі є одними з найбільших вразливостей, які існують у вашій організації.

1.1 Аутентифікація в кібербезпеці

Аутентифікація - це процедура встановлення належності користувачеві інформації в системі його ідентифікатора.

Технологія перевірки автентичності забезпечує контроль доступу для систем, перевіряючи, чи облікові дані користувача збігаються з обліковими даними в базі даних авторизованих користувачів або на сервері аутентифікації даних.

Користувачі, як правило, ідентифікуються з ідентифікатором користувача, а аутентифікація виконується, коли користувач надає облікові дані, наприклад пароль, який відповідає цьому ідентифікатору користувача. Більшість користувачів найбільш знайомі з використанням

пароля, який, як частина інформації, яка повинна бути відома тільки користувачеві, називається фактором аутентифікації знань. Інші фактори аутентифікації та спосіб їх використання для двофакторної або багатофакторної автентифікації.

Після аутентифікації користувач або процес, як правило, також піддаються процесу авторизації, щоб визначити, чи має дозвіл аутентифікованого об'єкта доступ до захищеного ресурсу або системи. Користувач може бути аутентифікований, але не може отримати доступ до ресурсу, якщо користувачеві не було надано дозволу на доступ до нього.

Терміни аутентифікації та авторизації часто використовуються як взаємозамінні. Хоча вони часто можуть бути реалізовані разом, ці дві функції відрізняються. Хоча аутентифікація є процесом перевірки ідентичності зареєстрованого користувача перед тим, як дозволити доступ до захищеного ресурсу, авторизація є процесом перевірки того, що авторизований користувач отримав дозвіл на доступ до запитаних ресурсів. Процес, за допомогою якого доступ до цих ресурсів обмежується певною кількістю користувачів, називається контролем доступу. Процес аутентифікації завжди приходить до процесу авторизації.

1.2 Сфери застосування

Аутентифікація користувача відбувається в межах більшості взаємодій від людини до комп'ютера за межами облікових записів гостей, автоматично ввійшли в облікові записи та комп'ютерні системи кіоску. Як правило, користувач повинен вибрати ім'я користувача або ідентифікатор користувача та надати дійсний пароль для початку використання системи.

Аутентифікація користувачів дозволяє автономно взаємодіяти з машиною в операційних системах і додатках, а також як дротові, так і бездротові мережі для забезпечення доступу до мережних і підключених до Інтернету систем, програм і ресурсів.

Багато компаній використовують аутентифікацію для перевірки користувачів, які входять до їхніх веб-сайтів. Без відповідних заходів безпеки дані про користувачів, такі як номери кредитних і дебетових карт, а також номери соціального страхування, можуть потрапити в руки кіберзлочинців.

Організації також використовують аутентифікацію, щоб контролювати, які користувачі мають доступ до корпоративних мереж і ресурсів, а також визначати та контролювати, які машини та сервери мають доступ. Компанії також використовують аутентифікацію, щоб дозволити віддаленим працівникам безпечно отримувати доступ до своїх програм і мереж.

Для підприємств та інших великих організацій аутентифікацію можна здійснити за допомогою системи єдиного входу (SSO), яка надає доступ до декількох систем з єдиним набором облікових даних для входу.

1.3 Як працює аутентифікація

Під час аутентифікації облікові дані, що надаються користувачем, порівнюються з даними у файлі в базі даних авторизованих користувачів або в локальній операційній системі або через сервер аутентифікації. Якщо облікові дані збігаються, а авторизований об'єкт має право використовувати цей ресурс, процес завершується і користувачеві

надається доступ. Повернення дозволів і папок визначають як середовище, яке бачить користувач, так і спосіб взаємодії з ним, включаючи години доступу та інші права, такі як обсяг місця для зберігання ресурсів.

Традиційно аутентифікація здійснювалася за допомогою систем або ресурсів, до яких здійснювався доступ; наприклад, сервер автентифікує користувачів, використовуючи власну систему паролів, реалізовану локально, використовуючи ідентифікатори для входу (імена користувачів) і паролі. Знання облікових даних для входу в систему вважається гарантованим, що користувач є автентичним. Кожен користувач спочатку реєструється (або реєструється іншим, наприклад, системним адміністратором), використовуючи призначений або самозваний пароль. При кожному наступному використанні користувач повинен знати і використовувати попередньо оголошений пароль.

Проте протоколи додатків мережі, HTTP і HTTPS, не мають статусу, а це означає, що сувора аутентифікація потребує повторної аутентифікації кінцевих користувачів кожного разу, коли вони отримують доступ до ресурсу за допомогою HTTPS. Замість того, щоб навантажувати кінцевих користувачів цим процесом для кожної взаємодії через Інтернет, захищені системи часто покладаються на аутентифікацію на основі маркерів, в якій аутентифікація виконується один раз на початку сеансу. Система аутентифікації видає підписаний маркер аутентифікації до програми кінцевого користувача, і цей маркер додається до кожного запиту від клієнта.

Аутентифікацію об'єктів для систем і процесів можна здійснювати за допомогою облікових даних машини, які працюють як ідентифікатор користувача та пароль, за винятком того, що дані облікового запису автоматично подаються відповідним пристроєм. Вони також можуть

використовувати цифрові сертифікати, які були видані та перевірені центром сертифікації як частину інфраструктури відкритого ключа для аутентифікації особи при обміні інформацією через Інтернет.

1.4 Ідентифікаційні фактори

Аутентифікація користувача з ідентифікатором користувача і паролем зазвичай вважається найосновнішим типом аутентифікації, і це залежить від того, що користувач знає дві частини інформації: ідентифікатор користувача або ім'я користувача і пароль. Оскільки цей тип аутентифікації спирається тільки на один фактор аутентифікації, це тип однофакторної аутентифікації.

Сильна аутентифікація - це термін, який формально не був визначений, але зазвичай використовується, щоб означати, що тип аутентифікації, який використовується, є більш надійним і стійким до атаки; досягнення, яке, як правило, визнається вимагає використання принаймні двох різних типів факторів аутентифікації.

Фактор аутентифікації являє собою деякий фрагмент даних або атрибут, який може використовуватися для аутентифікації користувача, який запитує доступ до системи.

Ці три фактори відповідають коефіцієнту знання, фактору володіння і фактору невідповідності. Протягом останніх років було запропоновано і введено в дію додаткові чинники, у багатьох випадках їх розташування є четвертим фактором, а час - п'ятим фактором.

Наразі використовуються такі фактори аутентифікації:

- Фактор знань: "Те, що ви знаєте".
- Фактор володіння: "Те, що у вас є".
- Фактор невідповідності: "Те, ким ти є".
- Розташування фактора: "Де ви знаходитесь".

- Фактор часу: "Під час аутентифікації".

Незважаючи на те, що використовуються в якості додаткових факторів аутентифікації, розташування користувача і поточний час самі по собі не є достатніми, без принаймні одного з перших трьох факторів, для аутентифікації користувача. Тим не менш, поширеність смартфонів допомагає полегшити обтяження багатофакторної аутентифікації для багатьох користувачів. Більшість смартфонів обладнані системою GPS, що дозволяє впевнено підтверджувати місце входу в систему; MAC-адреси смартфонів також можуть використовуватися для аутентифікації віддаленого користувача, незважаючи на те, що MAC-адреси відносно легко обманюють.

1.5 Традиційні методи

Традиційна аутентифікація залежить від використання файлу паролів, в якому ідентифікатори користувачів зберігаються разом з хешами паролів, пов'язаних з кожним користувачем. Під час входу в систему пароль, що подається користувачем, змішується і порівнюється зі значенням у файлі паролів. Якщо два хеши збігаються, користувач аутентифікується.

Додавання факторів автентифікації до процесу аутентифікації зазвичай покращує безпеку. Сильна автентифікація зазвичай відноситься до аутентифікації, яка використовує принаймні два фактори, коли ці фактори мають різні типи. Відмінність важлива; оскільки і ім'я користувача, і пароль можна вважати типом фактора знань, можна сказати, що базове ім'я користувача та автентифікація паролем використовують два

фактори знань для автентифікації - однак це не вважається формою двофакторної аутентифікації (2FA). Так само для систем аутентифікації, які покладаються на "питання безпеки", які також є "щось, що ви знаєте", для доповнення ідентифікатора користувача та паролів.

Двофакторна аутентифікація зазвичай залежить від фактора знання, що поєднується з біометричним фактором або фактором володіння, наприклад, маркером безпеки. Багатофакторна аутентифікація може включати будь-який тип аутентифікації, що залежить від двох або більше факторів, але процес аутентифікації, який використовує пароль, плюс два різних типи біометричних даних, не вважатиметься трьохфакторною аутентифікацією, хоча, якщо процес вимагає фактора знання, володіння фактор і фактор невідповідності, було б. Системи, які викликають ці три фактори плюс географічний чи часовий фактор, вважаються прикладами чотирьохфакторної аутентифікації.

Такий підхід до аутентифікації має ряд недоліків, особливо для ресурсів, розгорнутих в різних системах. По-перше, зловмисники, які можуть отримати доступ до файлу паролів для системи, можуть використовувати атаки грубої сили на хешовані паролі для вилучення паролів. З іншого боку, такий підхід вимагає декількох автентифікацій для сучасних додатків, які отримують доступ до ресурсів у різних системах.

Слабкі місця аутентифікації на основі паролів можуть бути вирішені до певної міри розумнішими іменами користувачів і правилами паролів, такими як мінімальна довжина та умови для складності, наприклад, включаючи великі і символи. Однак, аутентифікація на основі паролів і аутентифікація на основі знань є більш вразливими, ніж системи, які вимагають декількох незалежних методів.

Інші способи аутентифікації включають:

- Двофакторна аутентифікація - додає додатковий рівень захисту процесу аутентифікації. 2FA вимагає, щоб користувач надав другий фактор аутентифікації на додаток до пароля.
- Багатофакторна аутентифікація вимагає від користувачів аутентифікації з більш ніж одним фактором аутентифікації, включаючи біометричний фактор.
- Одноразовий пароль - це автоматично створений числовий або буквено-цифровий рядок символів, що аутентифікує користувача. Цей пароль дійсний лише для одного сеансу входу або транзакції.
- Біометрія - процес перевірки вашої ідентичності за допомогою ваших вимірювань або інших унікальних характеристик вашого тіла.
- Мобільна аутентифікація - це процес перевірки користувача через їхні пристрої або перевірку самих пристроїв. Процес аутентифікації мобільного зв'язку включає багатофакторну аутентифікацію, яка може включати одноразові паролі, біометричну аутентифікацію або перевірку QR-коду.
- Аутентифікація API - стандартні методи керування автентифікацією API: HTTP basic autentification; Ключі API та OAuth.
- Відкрита авторизація (OAuth) - це відкритий стандарт для аутентифікації та авторизації на основі маркерів в Інтернеті.

Машини також повинні авторизувати свої автоматизовані дії в мережі. Інтернет-служби резервного копіювання, системи виправлення та оновлення та системи віддаленого моніторингу, такі як ті, що

використовуються в телемедицині та технологіях інтелектуальних мереж, всі повинні надійно перевіряти автентичність, щоб переконатися, що це авторизована система, задіяна в будь-якій взаємодії, а не хакер.

Автоматична аутентифікація може бути виконана з авторизаційними даними, подібними до ідентифікатора користувача та пароля, що подаються лише відповідним пристроєм. Вони також можуть використовувати цифрові сертифікати, видані та перевірені сертифікаційним органом, як частину інфраструктури відкритого ключа, щоб довести ідентифікацію під час обміну інформацією через Інтернет, як тип цифрового пароля.

Важливо розуміти, що кожна точка доступу є потенційною точкою вторгнення. Кожному мережевому пристрою потрібна сильна автентифікація машини, а також, незважаючи на їхню звичайно обмежену активність, ці пристрої повинні бути налаштовані також для обмеженого доступу до дозволів, щоб обмежити, що можна зробити, навіть якщо вони порушені.

1.6 UX в кіберзахисті

У 1993 році Apple найняла свого першого UX архітектора, Дона Нормана. Сьогодні Норман вважається батьком дизайну, орієнтованого на людину - також відомого як дизайн користувача.

Він навіть придумав термін «користувальницький досвід». Норман пояснює: «Я винайшов цей термін, тому що я думав, що поняття інтерфейсу і зручності використання для людей були занадто вузькими. Я хотів охопити всі аспекти досвіду роботи з системою, включаючи промисловий дизайн, графіку, інтерфейс, фізичну взаємодію і посібник. "

Так само, як продукти Apple повинні бути приємними та легкими у використанні для споживачів, програмне забезпечення B2B повинно бути приємним та простим у використанні для окремих працівників.[8]

Нещодавно практики дизайну UX увійшли до сфери кібербезпеки. Мета полягає в тому, щоб поліпшити поширення кібер-можливостей серед зацікавлених сторін людини, тим самим покращивши кібер-здоров'я даної організації або забезпечивши більш безпечний досвід кінцевому споживачеві. Ми можемо побачити практику дизайну UX, що застосовується в кібернетиці на двох рівнях абстракції: мікро- та макро-.

Мікропрактики: саме це ми звикли бачити. Незважаючи на те, що дизайн UX не був фокусом у дні налаштування брандмауера командного рядка минулих років, сьогодні ми найчастіше бачимо його застосування таким чином:

- Програми для кінцевих користувачів: безперешкодний захист додатків для бізнесу та споживачів, з якими взаємодіють кінцеві користувачі (наприклад, єдина реєстрація, проста багатофакторна аутентифікація)
- Технології безпеки: надання на перший погляд інформаційних панелей і наочних візуалізованих наборів даних, а також зшивання інструментів через API

Макро-практика: хоча мікро-фокус, безумовно, важливий, ми досягли нових висот, де нам потрібно залучити дизайн UX для більших, більш складних проблем. Коли ми думаємо про вдосконалення кібер-стратегії на підприємстві, це стосується завоювання довіри. Кібер-програма не може бути успішною у вакуумі, і для того, щоб всі ці інвестиції в реальні можливості дійсно окупилися, треба подолати багато

перешкод. Кібербезпека - це складна адаптивна система, і проектування UX є ключем у формуванні середовища для успіху. Прикладами застосування дизайну UX у кібербезпеці є:

- Постійне вдосконалення можливостей: отримання зворотнього зв’язку для можливості розробки та налаштування функцій захисту, виявлення та реагування
- Вирівнювання операційної моделі: вдосконалення та спрощення процесів
- Формування культури: зміна культури безпеки в інтернеті для власного захисту[10]

Виділення грошей на UX і UI, також є важливим для маркетингових цілей, оскільки ефективність інструментів кібербезпеки залежить від здатності та бажання клієнта їх використовувати. Платформа кібербезпеки повинна бути як функціональною, так і зручною для залучення уваги клієнтів та фінансування інвесторів. Як підкреслює Hili Geva, COO продуктового агентства Inkod: «Коли йдеться про UX для кібербезпеки, завдання полягає в тому, щоб перетворити рішення компанії на найбільш інноваційну і конкурентну платформу кібербезпеки. UX і UI продукту повинні бути зручними як з клієнтами, так і з інвесторами.

Іншими словами, кращий захист не повинен означати погіршення роботи користувачів. Маючи це на увазі, ось деякі важливі фактори, які необхідно враховувати для поліпшення UX і UI кібербезпеки.[9]

Експерти користувацького досвіду розробляють свою стратегію проектування з урахуванням потреб користувачів. У більшості галузей промисловості максимальною простотою та простотою використання для клієнта є очевидна мета. Але в індустрії кібербезпеки це викликає цікаве

запитання: якщо дійсно існує компроміс між сильною безпекою і хорошим UX, то як безпека може бути ефективною і приємною для користувача?

Багато викликів UX для програм кібербезпеки та інших програмних програм трапляються тому, що вони розроблені з урахуванням технічної коректності - але не обов'язково узгоджуються з інтерпретацією користувача.

Особливо наочним прикладом цього дисонансу є спливаюче повідомлення про помилку Microsoft Windows 3.1 - 98. У спливаючому вікні прочитано: «Ця програма виконала нелегальну операцію і буде закрита». З чисто технічної точки зору, повідомлення лише викликало природну тривогу для користувачів.

Тільки через те, що щось є технічно правильним, це не означає, що це обов'язково зручно для користувачів. Під час зміни або додавання нових функцій продукту двічі перевірте, щоб переконатися, що технічні коригування відображаються користувачам таким чином, що вони інтуїтивно розуміють.

Коли йдеться про введення нової системи кібербезпеки, програмне забезпечення має бути плавно інтегровано в існуючу мережеву інфраструктуру без необхідності видалення або реструктуризації існуючих інструментів. Новий інструмент або програмне забезпечення безпеки не повинен порушувати робочий процес співробітника або втручатися в існуючі програми компанії.

Навіть після процесу вбудовування, засоби кібербезпеки повинні бути максимально простими у використанні. Замість того, щоб переповнювати користувача складними технічними даними, ці інструменти повинні мати наочний огляд інформаційних панелей з зручною для збору інформації.

З особливою увагою до UX, покращена безпека зовсім не навантажена. Навпаки, вона йде пліч-о-пліч з юзабіліті, оскільки працівники частіше дотримуються протоколу безпечного користування у своїй щоденній роботі. Люди хочуть використовувати свої пристрої більш безпечними способами - за умови, що це не спричиняє труднощів або незручностей.

Коли справа доходить до кібербезпеки, багато компаній страждають від низького рівня зрілості UX. Це означає, що більшість співробітників є розробниками, які прийшли з технічною освітою - але є хороші шанси, що жоден з них ніколи не працював з дизайнером UX раніше. З мого досвіду, більшість з них вважає, що UX є в основному UI, і що робить UX-дизайнер, вибирає кольори і приємні шрифти. Лише дуже мало хто розуміє концепцію створення бездоганного, інтуїтивного досвіду в продукті, який включає складні параметри та концепції. Я не звинувачую їх - цей вид підприємств в основному технічно орієнтований і завжди відмовлявся від належного UX-планування, досліджень і дизайну. На жаль, результатом є те, що багато хто з цих продуктів виглядають базовими і непослідовними з точки зору інтерфейсу користувача, часто з довільною інформаційною архітектурою, побудованою на ходу, і різними шаблонами UX для дуже схожих функцій. Не дивно, що всі ми представляємо кібербезпеку підприємства як темне, темне місце з системами, які виглядають так, як вони з 80-х років.

Більшість компаній і продуктів кібербезпеки побудовані швидко і брудно для того, щоб зробити продукт і зібрати валізу готівки, щоб відпочивати. Разом з тим, що засновники, як правило, є технічно орієнтованими та менш обізнаними з тим, що таке UX, вони, як правило, не бачать цінності на поліпшенні платформи або виконуючи належне

планування, оскільки це заважає їм «швидко рухатися». Тільки на більш пізньому етапі вони розуміють, що інтерфейс стає більшим, розширеним і функціонує як безлад, а потім шукають дизайнерів UX, щоб вони виглядали краще.

Дизайнери UX прибувають і починають вивчати складні теми, досліджуючи та розробляючи рішення. Ці рішення показуються на дошці, і відповідь часто:

«Так, це звучить добре, але потрібно багато зусиль, і ми не маємо часу і ресурсів, щоб зробити це зараз. Ми можемо зробити це в майбутньому. "

Дизайнери UX потрапляють у цикл бажань поліпшити продукти, і приносячи рішення, які ледь бачать світло. Між тим, продукт продовжує зростати, і архітектура стає більш складною.

Зрештою, це відбувається тому, що вартість поганого UX все ще недостатньо неприємна для компаній. Тільки тоді, коли поганий UX почне коштувати компаніям великих грошей, вони матимуть достатньо стимулів для того, щоб покласти на нього реальні зусилля.

Інвестиції компанії в UX не можуть бути виміряні кількістю найманих дизайнерів, але кількість розробників, які вони виділяють для поліпшення UX.

Однак для дизайнерів UX ця ситуація може бути надзвичайно розчарувальною, що супроводжується відчуттям, що «неможливо робити хороші UX у кібербезпечних компаніях».

Я вважаю, що навіть у складному середовищі є щось, що можна зробити, щоб створити хороший UX для кібербезпеки.

Мої поради для перемоги UX у кібербезпеки:

- Розвивати освіту людей у цій сфері: організувати швидкі навчальні презентації, щоб підняти важливість UX і пояснити процеси та прийняття рішень
- Досліджувати проблеми і потреби користувачів та обмінюватись результатами: технології кібербезпеки, як правило, швидко змінюються, і спілкування з користувачами може принести багато результатів

- Залишати речі легкими та простими: з точки зору UX та UI слід робити дизайн дуже простим, інвестувати час у правильну та надійну архітектуру, а також відносно обмежений набір шаблонів UX, які використовують у багатьох випадках та областях у системі.[10]

1.7 One tap Sign-up/ Auto sign-in

Новий метод входу дозволяє користувачам реєструватися, не турбуючи їх екраном реєстрації. Користувачі отримують на вашому сайті безпечний обліковий запис на основі маркерів, без паролів, захищений обліковим записом Google.

Програма тестування бета-версії для цього API зараз закрита.

Управління федеративними ідентифікаціями (FIM) - це угода, яка може бути зроблена між декількома підприємствами, щоб дозволити абонентам використовувати ті ж ідентифікаційні дані, щоб отримати доступ до мереж усіх підприємств групи. Використання такої системи іноді називається федерацією ідентифікації.

Федерація ідентифікації пов'язує ідентифікацію користувача з кількома доменами безпеки, кожна з яких підтримує свою власну систему управління ідентифікацією. Коли два домени об'єднані, користувач може пройти аутентифікацію в одному домені, а потім отримати доступ до ресурсів в іншому домені без необхідності виконання окремого процесу входу.

Федерація ідентифікації пропонує економічні переваги, а також зручність для підприємств та їхніх абонентів. Наприклад, кілька корпорацій можуть спільно використовувати одну заявку, що призводить до економії коштів і консолідації ресурсів.

Федерація ідентифікації включає в себе великий набір випадків користувальницьких і прикладних додатків на рівні браузера, а також рівень сервісно-орієнтованої архітектури.[11]

Це новий шлях для користувачів увійти або зареєструватись на веб-сайті з одним клацанням комп’ютерної миші, використовуючи аккаунт, який хостується на 3 веб-ресурсі, що називається identity provider.

Identity federation побудована на стандартах коду, таких як OpenID Connect та OWS З цією технологією, користувачам не потрібно створювати додатковий пароль.

Можна делегувати безпеку виклики ідентичності постачальника послуг отримувати інформацію про профіль від цього постачальника посвідчень. Google - один з таких постачальників ідентичності..

Redfin - компанія в США, яка займається нерухомістю , спостерігала результат збільшення кількості реєстрації нових користувачіва на 80% після впровадження One tap Sign-up/ Auto sign-in. Крім того, більше 40% нових користувачів повернулися на веб-сайт більше п'яти разів після реєстрації.

Trivago є одним з світових лідерів у пошуку та замовленню готелів, ща працює у 55 країнах. Він набрав на 50% більше нових реєстрації користувачів після реалізації цієї бібліотеки.

Cifroclub, Letras, і Palco - популярні музичні сайти в Бразилії для пісень та текстів, отримали в 43 рази більше зареєстрованих користувачів після інтеграції One tap Sign-up/ Auto sign-in.[12]

Також хочу розглянути деякі рішення проблем аутентифікації, коли використовується пароль та логін. Що ми можемо зробити, якщо нападник вже знає пароль та намагається зламати акаунт.

В основному, така атака на користувацькі записи здійснюється ботами. Це означає, що якщо є можливість відфільтрувати запити від ботів, кількість викрадених акаунтів має суттєво зменшитися. І це те, що робить reCAPTCHA.

Рисунок 1.1 - Старий вигляд reCAPTCHA

Декілька років тому, ця технологія змушувала людей прочитати спотворений текст та ввести його у поле. Але це можна зробити простіше.

Тоді була створена reCAPTCHA v2, де користувачі можуть просто натиснути на форму та пройти перевірку. V2 достатньо розумний, щоб визначити, чи є взаємодія вразливою лише одним простим жестом.

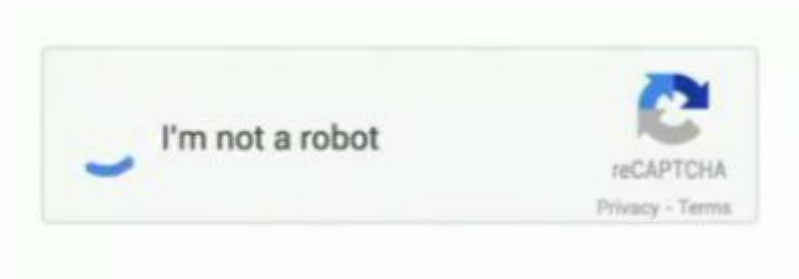

Рисунок 1.2 - Вигляд reCAPTCHA v2

І якщо reCAPTCHA досі не може визначитись, вона вимагає додаткового завдання - вибрати певні зображення з набору.

Рисунок 1.3 - Вигляд додаткового завдання reCAPTCHA

Це приклад питання, на яке багато ботів не зможуть відповісти легко. Ця технологія захищає понад 2 мільйонів веб-сайтів щотижня від спаму. Але боти також розвиваються.

Напади на reCAPTCHA протягом останніх кількох років останні кілька років, еволюціонують з методів грубої сили або довільного

відгадування ботів до розумніші та навіть використовують AI. Почали створюватись рішення для ботів на основі машинного навчання та хакерів, щоб спробувати щоб подолати ці проблеми. Але ми хочемо зупинитися ботів без залежності від того чи зможуть вони знайти певні ознаки в наборі зображень. Сьогодні існує публічна бета-версія reCAPTCHA v3. Нова версіє вдосконалює 3 речі. По-перше, вона не вимагає інтерактивних дій від користувача. По-друге, вона аналізує трафік з допомогою Adaptive Risk Analysis Engine. І по-третє, він зупиняє потік трафік за дією.

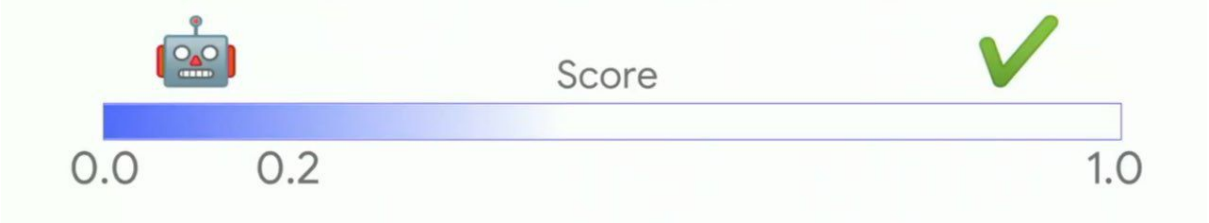

Рисунок 1.4 - Оцінка шкідливості reCAPTCHA

У v3 reCAPTCHA виявляє, чи є взаємодія з веб-сайтом є шкідливою навіть без натиску на екран. Це означає, що ви можете зберегти свій веб-сайт безпечни, не турбуючи користувачів. І замість простого так чи ні, це дасть вам оцінку яка знаходяться в діапазоні від 0 до 1,0. Оцінка розраховується Adaptive Risk

Analysis Engine і сигналами від взаємодій з вашим веб-сайтом. Базуючись на цих показниках можна визначити власний поріг довіри для відповіді на запит.

Висновки до розділу

Аутентифікація є важливою, оскільки вона дозволяє організаціям підтримувати безпеку своїх мереж, дозволяючи лише авторизованим користувачам отримувати доступ до захищених ресурсів, які можуть включати комп'ютерні системи, мережі, бази даних, веб-сайти та інші мережеві програми або служби.

Авторизація включає в себе процес, за допомогою якого адміністратор надає права аутентифікованим користувачам, а також процес перевірки дозволів користувача облікового запису, щоб переконатися, що користувачеві було надано доступ до цих ресурсів. Привілеї та переваги, надані авторизованому обліковому запису, залежать від дозволів користувача, які зберігаються локально або на сервері аутентифікації. Налаштування, визначені для всіх цих змінних середовища, встановлюються адміністратором.

Системи і процеси можуть також потребувати авторизації своїх автоматизованих дій в мережі. Інтернет-служби резервного копіювання, системи виправлення та оновлення та системи віддаленого моніторингу, такі як ті, що використовуються в телемедицині та технологіях інтелектуальних мереж, всі повинні надійно перевіряти аутентичність.

2 АУТЕНТИФІКАЦІЯ НА ОСНОВІ WEBAUTHN

Коли користувачі входять до порталу, який використовує аутентифікацію без паролів, вони отримують одноразовий код автентифікації через текстове повідомлення, повідомлення про мобільний додаток або електронну пошту. Цей код замінює стандартний пароль і дозволяє користувачам автоматично входити до програми. Вона може використовувати аутентифікацію без паролів для програм, мобільних веб-додатків або порталів мобільних сайтів, але вона також може працювати для підключення до Wi-Fi або мобільного VPN.

Нові пропозиції від постачальників, таких як Yubico, забезпечують гібридний підхід до мобільної аутентифікації без паролів. Yubico спирається на свій маркер безпеки YubiKey - невеликий фрагмент обладнання, що забезпечує рівень аутентифікації для користувачів. Ключі безпеки можуть функціонувати як єдиний чинник або як частина багатофакторного підходу до аутентифікації.

Amazon, Cisco і Microsoft пропонують аутентифікацію без паролів, але на ринку також є менш відомі постачальники, такі як Auth0 і Hypr. Auth0 дозволяє текстові повідомлення та повідомлення електронної пошти як методи аутентифікації. Програма перевірки автентичності корпорації Майкрософт для Apple iOS і Google Android дає змогу користувачам затверджувати вхід до інших програм Microsoft за допомогою мобільного push-сповіщення.

"Аутентифікатор" може бути апаратним ключем безпеки, який користувач підключив до свого комп'ютера, або біометричним ідентифікатором, який можна придбати з ПК або датчиків смартфона -

таких, як відбитки пальців, сканування обличчя, сканування діафрагми та інші.

WebAuthn аутентифікація - це розширення API керування обліковими записами, що дозволяє здійснювати сильну аутентифікацію за допомогою криптографії відкритого ключа, дозволяючи аутентифікацію без паролів і / або захищаючи аутентифікацію другого фактора без SMS-повідомлень.

На багатьох веб-сайтах вже є сторінки, які дозволяють користувачам реєструвати нові облікові записи або входити до існуючого облікового запису, а API веб-перевірки автентичності діє як заміна або доповнення до тих веб-сторінок.

2.1 Паролі не потрібні

Однією з основних недоліків аутентифікації на основі паролів є те, що пароль є загальним секретом.

Перевірка автентичності без паролів створює нові можливості для IT, оскільки вона не дозволяє користувачам мобільних пристроїв приймати неефективні рішення щодо безпеки. Аутентифікація на основі пароля відкриває двері для численних помилок користувачів, які негативно впливають на безпеку організації. Під автентифікацією на основі пароля користувачі можуть встановлювати та використовувати короткі або легко вгадувані паролі, приймати персональні та ділові паролі або повторно використовувати той самий пароль для кількох програм і систем. За допомогою автентифікації без паролів організації можуть уникнути всіх цих вразливостей.

Однак аутентифікація без паролів для мобільних пристроїв не забезпечується автоматично, але її безпека залежить від її реалізації. Існують певні сценарії загроз, в яких зловмисники можуть використовувати автентифікацію без паролів, наприклад, коли вони мають доступ до мобільного пристрою або облікового запису електронної пошти користувача. Тим не менш, аутентифікація без паролів є більш безпечною, ніж у більшості організацій: на шляху найменшого опору зі слабкими паролями, загальними паролями тощо.

Фахівці з інформаційних технологій, які прагнуть здійснити аутентифікацію без паролів, повинні виконати належну перевірку, розробити вимоги та цілі для технології, а потім виконати підтвердження концепції з постачальником або двома, щоб побачити, як працює технологія. Якщо організація правильно реалізовує, аутентифікація без паролів є безпечним засобом для вирішення завдань входу, з якими стикаються користувачі та IT щодня.[5]

2.2 WebAuthn

Консорціум World Wide Web (W3C) оголосив у березні 2019 року, що веб-аутентифікація (WebAuthn) офіційно стала веб-стандартом. Він служить наступним кроком у прагненні галузі до усунення або, принаймні, зменшення залежності від паролів, і замість цього зосереджується на біометрії та інших методах аутентифікації.

Отже, що таке WebAuthn? Це API керування обліковими записами, який дозволяє веб-програмам перевіряти автентичність користувачів, не зберігаючи їх на серверах. API використовує криптографію з відкритим

ключем, яка передбачає використання приватної публічної клавіші, де ви зберігаєте приватний ключ на своєму пристрої, і сервер має відкритий ключ, який нічого не вартий без закритого ключа.

Що ж означає WebAuthn для користувачів? Раніше користувачі надавали загальний секрет (наприклад, свій пароль) під час входу до облікових записів, які потім зберігаються на сервері, який може бути або не бути безпечним. Таким чином, погані учасники отримали доступ до вашого облікового запису: можливо, пароль зберігається у звичайному тексті, сервер не належним чином захищений, або ваш пароль досить легкий для вгадування. Пароль - це єдиний ключ, який підтверджує, що ви насправді ви. Якщо хакер вдається вкрасти його, вони можуть повністю видати себе за вас, якщо ви не є одним з 28% користувачів, які використовують двофакторну аутентифікацію.

Тому WebAuthn усуває необхідність мати серверні паролі. Замість цього сервер реєструє облікові дані WebAuthn за допомогою приватної публічної клавіші, яка також включає ідентифікатор для цього користувача. Щоб запобігти атакам відтворення, сервер повинен створити виклик, який складається з випадково згенерованого рядка.[3][4]

2.3 FIDO2

Альянс FIDO знаходиться на хорошій позиції з переорієнтацією на галузь до умовного доступу / нульової довіри. Все більше організацій потрапляють в ідею входу без паролів, які стандарти FIDO забезпечують за допомогою апаратних ключів і біометричної аутентифікації.

Незадовго до оголошення WebAuthn, Google та FIDO Alliance виявили, що Android 7+ тепер сертифікований FIDO2. Сертифікація дозволить розробникам Android-програм увімкнути вхід без пароля.

FIDO2 є наступником універсального протоколу FIDO 2-го фактора (U2F). Аутентифікація FIDO2 має всі переваги U2F - основна відмінність полягає в тому, що аутентифікатор FIDO2 є багатофакторним аутентифікатором.

Аутентифікатор FIDO2 може використовуватися в однофакторному або багатофакторному режимі. У однофакторному режимі аутентифікатор активується тестом присутності користувача (TUP), який зазвичай складається з простого натискання кнопки. У багатофакторному режимі аутентифікатор активується або PIN-кодом, або біометричним способом.

У будь-якому випадку аутентифікація FIDO2 не вимагає традиційного пароля. Запам'ятовуваний секрет (PIN-код) необхідний для перевірки користувача в багатофакторному режимі, але цей секрет не надсилається веб-сайту. PIN-код зберігається на аутентифікаторі і ніколи не залишає аутентифікатора. Крім того, один PIN-код працює з усіма веб-сайтами.

На додаток до PIN-коду, аутентифікатор FIDO2 може підтримувати біометричний (як правило, відбиток пальця). Як і PIN-код, біометричні дані зберігаються на аутентифікаторі і ніколи не залишають аутентифікатора.

PIN-код та біометричні дані на аутентифікаторі працюють разом, як PIN-код і біометричний на вашому смартфоні. На практиці відбиток пальця використовується для забезпечення зручного доступу до вашого смартфона, але зрідка доступ до відбитків пальців не відбувається, і в

цьому випадку телефон повертається на PIN-код. Аутентифікатор FIDO2 з PIN-кодом і біометричною аналогією.

Оскільки загальний секрет не передається по мережі, аутентифікація FIDO2 іноді називається веб-аутентифікацією без пароля, яка дещо вводить в оману (через PIN-код).[13][15]

2.4 Принцип роботи протоколу

Як і його попередник FIDO U2F, веб-аутентифікація W3C (WebAuthn) включає веб-сайт, веб-браузер і аутентифікатор:

- Сайт є відповідним WebAuthn Relying Party
- Браузер є відповідним клієнтом WebAuthn (В основі WebAuthn API лежать два базових методи, відповіді на запит і вхід до системи: navigator.credentials.create () і navigator.credentials.get ())
- Аутентифікатор FIDO2, що сумісний з клієнтом WebAuthn (Він керує ідентифікаційними даними та відповідає за генерацію публічних ключів для облікових записів.)

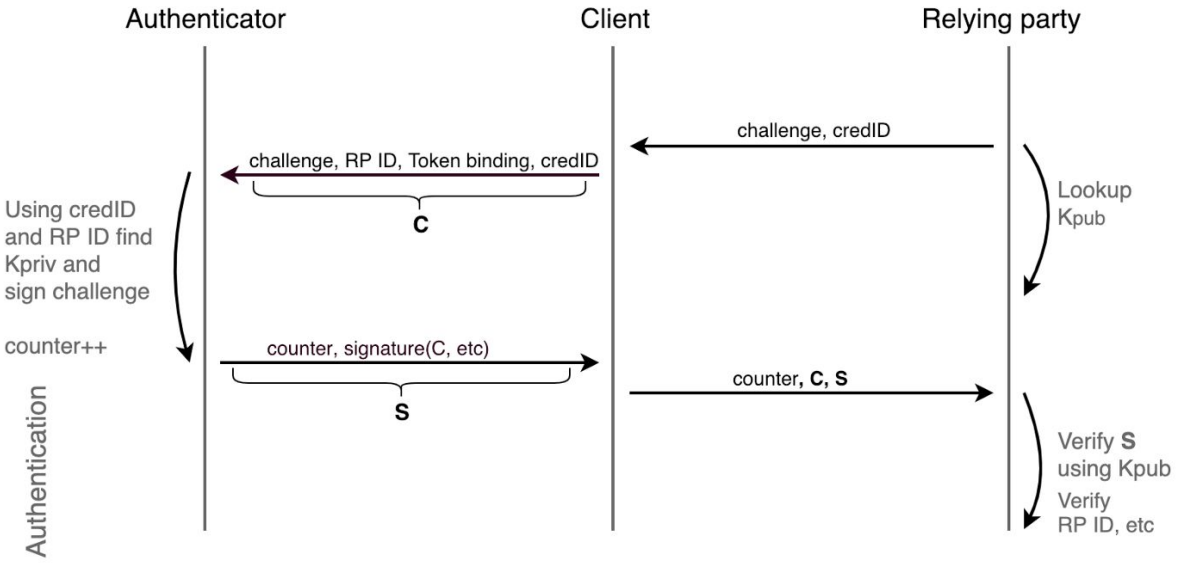

Рисунок 2.1 - Схема роботи FIDO

Щоб ініціювати потік аутентифікації WebAuthn, сторона, що довіряє WebAuthn, вказує свої наміри на клієнту WebAuthn (тобто браузер) через JavaScript. Клієнт WebAuthn спілкується з аутентифікатором за допомогою JavaScript API, реалізованого в браузері. Роумінг-аутентифікатор відповідає протоколу FIDO-клієнт для аутентифікатора.[1]

Процедура авторизації виглядає так:

- Користувач заходить на сайт та обирає опцію безпарольної аутентифікації.
- Сайт направляє клієнту WebAuthn(браузеру) відповідний JavaScript запит.
- Браузер звертається до аутентифікатору, щоб той згенерував ключі і направив їх перевіряючій стороні.
- Сервер перевіряє дані для входу.
- Якщо все добре, користувач авторизується на сайті.

2.2.1 Реєстрація

Типовий процес реєстрації має шість етапів, як показано на Рисунку 1 і описані нижче. Це спрощення даних, необхідних для процесу реєстрації, що має на меті лише надати огляд.

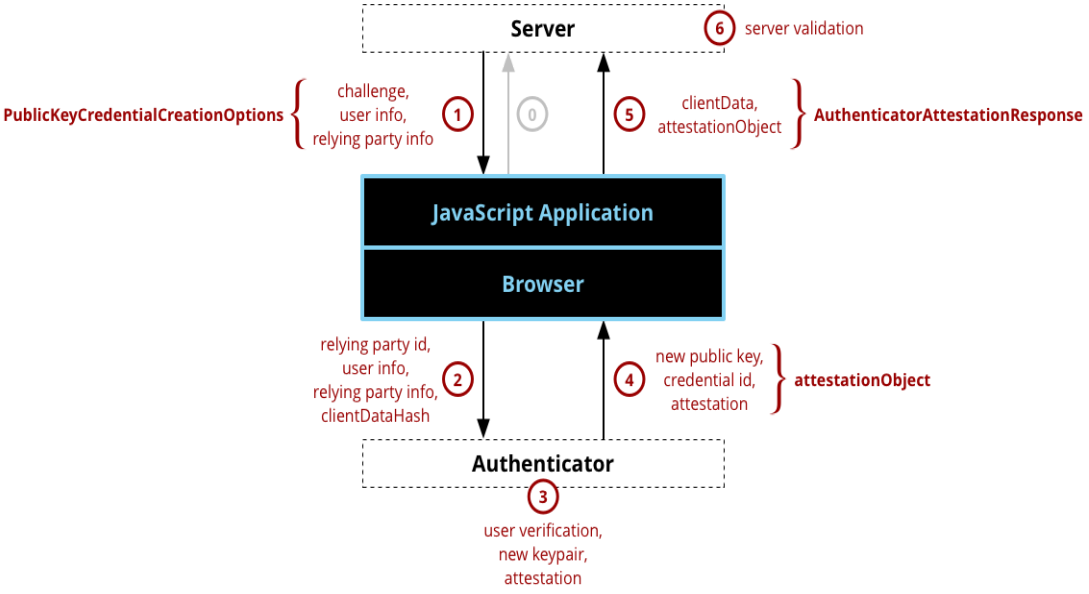

Рисунок 2.2 - Схема реєстрації

Процедури реєстрації:

1) Application Requests Registration - додаток робить початковий запит на реєстрацію. Протокол і формат цього запиту виходять за рамки API веб-аутентифікації.
2) Server Sends Challenge, User Info, and Relying Party Info - Сервер відправляє виклик, інформацію про користувача та Relying Party Info до JavaScript запиту. Протокол для спілкування з сервером не вказаний і знаходиться за межами області API веб-перевірки автентичності. Параметри, отримані від сервера, будуть передані

виклику create (), як правило, з невеликою або зовсім відсутністю модифікації, і повертає Promise, який використовується в PublicKeyCredential, що містить AuthenticatorAttestationResponse.

3) Браузер викликає authenticatorMakeCredential() на аутентифікаторі - внутрішньо браузер перевірить параметри і заповнить будь-які значення за замовчуванням, які стануть AuthenticatorResponse.clientDataJSON. Одним з найважливіших параметрів є походження, яке записується як частина клієнтських даних (походження може бути перевірено сервером пізніше). Параметри виклику create() передаються автентифікатору разом із хешем SHA-256 clientDataJSON (надсилається лише хеш, оскільки посилання на аутентифікатор може мати низьку пропускну здатність NFC або Bluetooth, а аутентифікатор просто збирається підписати хеш, щоб переконатися, що він не підроблений).
4) Аутентифікатор створює ключову пару та атестацію - Перш ніж робити що-небудь, аутентифікатор зазвичай просить певну форму перевірки користувача. Це може бути введення PIN-коду, використання відбитка пальця, сканування діафрагми тощо, щоб довести, що користувач присутній і погоджується на реєстрацію. Після перевірки користувача аутентифікатор створить нову пару асиметричних ключів і безпечно збереже приватний ключ для подальшого використання. Відкритий ключ стане частиною атестації, яку аутентифікатор підпише з приватним ключем, який було спалено в аутентифікаторі під час його виробничого процесу, і який має ланцюжок сертифікатів, який можна перевірити до кореня довіри.

5) Authenticator Returns Data to Browser - Новий відкритий ключ, унікальний ідентифікатор облікового запису в усьому світі та інші дані атестації повертаються браузеру, де вони стають об'єктом `attestationObject`.
6) Браузер створює кінцеві дані, програма відсилає відповідь на сервер - запит create() вирішує `PublicKeyCredential`, який має `PublicKeyCredential.rawId`, який є глобальним унікальним ідентифікатором облікового запису разом з відповіддю, що є `AuthenticatorAttestationResponse`, що містить `AuthenticatorResponse.clientDataJSON` і `AuthenticatorAttestationResponse.attestationObject`. `PublicKeyCredential` посилається назад на сервер за допомогою будь-якого бажаного форматування та протоколу.
7) Сервер перевіряє та завершує реєстрацію - сервер повинен виконати серію перевірок, щоб переконатися, що реєстрація була завершена і не підроблена.

2.2.2 Аутентифікація

Після того, як користувач зареєструвався з веб-аутентифікацією, вони згодом можуть автентифікувати (логін або вхід) з послугою. Потік аутентифікації виглядає схожим з потоком реєстрації, а ілюстрація дій на рисунку 2 може бути розпізнається як подібна до ілюстрації дій реєстрації на рисунку 1. Основними відмінностями між реєстрацією та аутентифікацією є те, що:

1) Аутентифікація не вимагає від користувача або інформації, що покладається на сторону;
2) Аутентифікація створює твердження, використовуючи раніше створену пару ключів для послуги, а не створює атестацію з парою ключів, яка була спалена в аутентифікаторі під час виробництва;

Знову ж таки, опис аутентифікації нижче - це загальний огляд, а не врахування всіх параметрів і функцій API веб-аутентифікації.

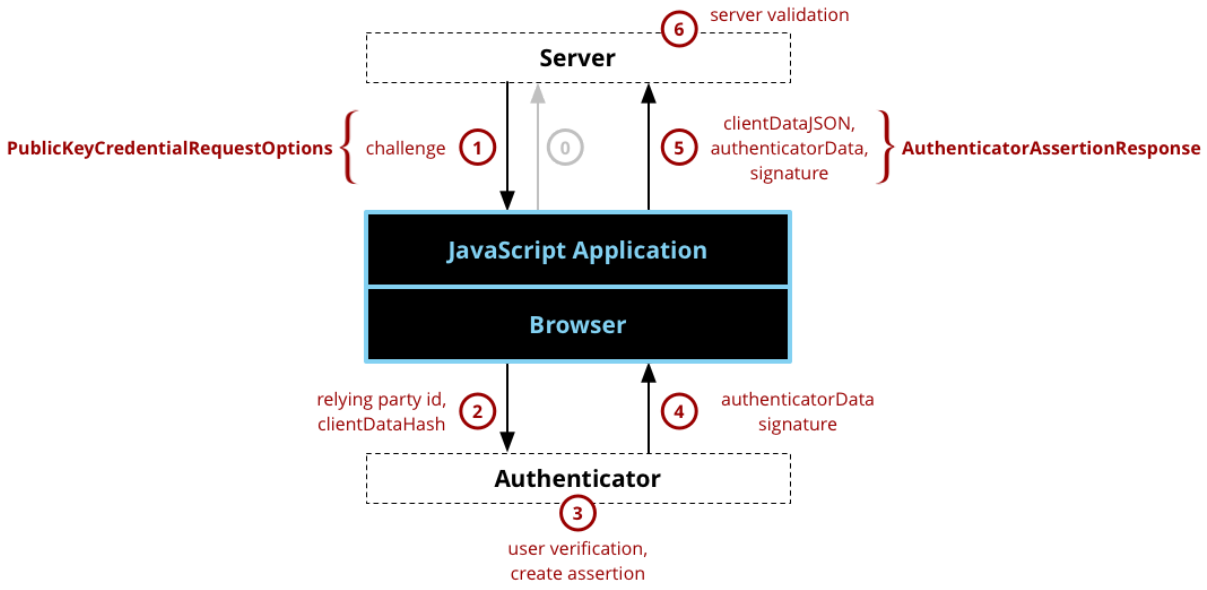

Рисунок 2.3 - Схема аутентифікації

Процедури реєстрації:

1) Аутентифікація запитів програми - програма виконує початковий запит аутентифікації. Протокол і формат цього запиту виходять за рамки API веб-аутентифікації.
2) Сервер надсилає виклик - сервер надсилає виклик програмі JavaScript. Протокол для спілкування з сервером не вказаний і знаходиться за межами області API веб-перевірки аутентичності.

Параметри, отримані від сервера, будуть передані виклику `get()`, як правило, з невеликими змінами або без них.

3) Браузер викликає `authenticatorGetCredential()` на аутентифікаторі - внутрішньо браузер перевірить параметри і заповнить будь-які значення за замовчуванням, які стануть `AuthenticatorResponse.clientDataJSON`.
4) Аутентифікатор створює твердження - знаходить облікові дані для цієї служби, яка відповідає ідентифікатору надійної сторони та спонукає користувача на згоду на аутентифікацію. Припускаючи, що обидві ці кроки є успішними, аутентифікатор створить нове твердження шляхом підписання над `clientDataHash` і `authenticatorData` з закритим ключем, створеним для цього облікового запису під час реєстраційного виклику.
5) Аутентифікатор повертає дані браузеру - аутентифікатор повертає аутентифікатор даних і підпис твердження браузеру.
6) Браузер створює кінцеві дані, програма відсилає відповідь на сервер - браузер вирішує Promise до `PublicKeyCredential` з `PublicKeyCredential.response`, що містить `AuthenticatorAssertionResponse`. Програма JavaScript передає ці дані назад на сервер, використовуючи будь-який обраний протокол і формат.
7) Сервер перевіряє та завершує реєстрацію - сервер повинен виконати серію перевірок, щоб переконатися, що реєстрація була завершена і не підроблена.

2.2.3 Дані аутентифікатора

WebAuthn API використовує асиметричну криптографію замість паролів або SMS-текстів для реєстрації. Це вирішує значні проблеми безпеки, пов'язані з фішингу, порушеннями даних і атаками на тексти SMS або інших методів аутентифікації другого фактора, в той же час значно збільшуючи простоту використання (оскільки користувачам не потрібно керувати десятками складних паролів).

Структура даних аутентифікації кодує контекстні прив'язки, зроблені аутентификатором. Ці прив'язки контролюються самим автентифікатором і отримують їхню довіру від оцінки WebAuthn Relying Party властивостей безпеки аутентифікатора.

В одному крайньому випадку аутентифікатор може бути вбудований в клієнта, і його прив'язки можуть бути не більш надійними, ніж дані клієнта. З іншого боку, аутентифікатор може бути дискретним об'єктом з апаратним забезпеченням і програмним забезпеченням високої безпеки, підключеним до клієнта через захищений канал. В обох випадках довіряюча сторона отримує дані аутентифікатора в тому ж форматі і використовує свої знання про аутентифікатор для прийняття рішень про довіру.

Дані аутентифікатора мають компактне, але розширюване кодування. Це є бажаним, оскільки аутентифікатори можуть бути пристроями з обмеженими можливостями та низькими потребами в енергії, зі значно простішими програмними стеками, ніж клієнтська платформа.

Структура даних автентифікатора - це байтовий масив 37 байт або більше.

Ідентифікатор RP спочатку отримується від клієнта, коли створюються облікові дані та генерується твердження. Однак, він відрізняється від інших даних клієнта деякими важливими способами. По-перше, на відміну від даних клієнта, ідентифікатор RP облікового запису не змінюється між операціями, а залишається незмінним протягом терміну дії цих облікових даних. По-друге, він перевіряється за допомогою аутентифікатора під час операції `authenticatorGetAssertion`, перевіряючи, що ідентифікатор RP, який запитаний обліковий запис визначений, точно відповідає ідентифікатору RP, наданому клієнтом.

Аутентифікатори виконують наступні кроки для створення структури даних аутентифікатора:

- Hash RP ID використовуючи SHA-256 для генерування `rpIdHash`.
- Прапор UP має бути встановлений тільки тоді, коли аутентифікатор виконав перевірку присутності користувача. Прапор УФ повинен бути встановлений тільки тоді, коли аутентифікатор виконав перевірку користувача. Біти RFU повинні бути встановлені в нуль.
- Для підписів атестації аутентифікатор повинен встановлювати прапор AT і вмикати `attestedCredentialData`. Для підписів аутентифікації прапор AT не повинен бути встановлений, і `attestedCredentialData` не повинні вмикатися.
- Якщо аутентифікатор не містить будь-яких даних розширення, він повинен встановити прапор ED до нуля, і до одного, якщо дані розширення включені.

На малюнку нижче показано візуальне представлення структури даних аутентифікатора:

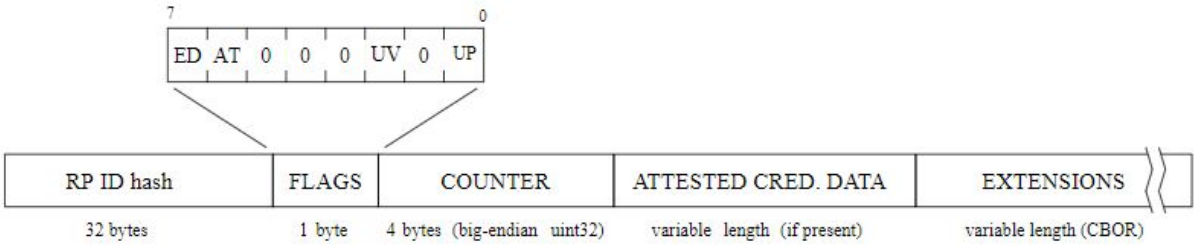

Рисунок 2.4 - Схема структура даних аутентифікатора

Формат для підписів затвердження, які підписують конкатенацію структури даних аутентифікатора і хеш серіалізованих даних клієнта, сумісні з форматом підпису аутентифікації FIDO U2F.

Це відбувається тому, що перші 37 байт підписаних даних у повідомленні відповіді ідентифікації UID FIDO складають дійсну структуру даних автентифікатора, а решта 32 байта є хешем даних серіалізованого клієнта. У цій структурі даних аутентифікатора `rpIdHash` є параметром програми FIDO U2F, всі прапори, крім UP, завжди нульові, а засвідченіданідальні дані та розширення ніколи не присутні. Підписи ідентифікації FIDO U2F можуть бути перевірені тим же самим способом, що й інші сигнатури підтвердження, створені операцією `authenticatorMakeCredential`.

Аутентифікатори можуть реалізовувати різні транспортні засоби для спілкування з клієнтами. Для цього потрібно вказати можливий тип з’єднання у наведеному нижче коді.

```
enum AuthenticatorTransport {
    "usb",
    "nfc",
```

```
        "ble",
        "internal"
};
```

Багато аспектів дизайну API веб-автентифікації обумовлені проблемами конфіденційності. Основною проблемою, що розглядається в цій специфікації, є захист особистої ідентичності користувача, тобто ідентифікація людини або співвідношення окремих ідентичностей як належності до однієї і тієї ж людини. Незважаючи на те, що API веб-перевірки автентичності не використовує або не надає будь-якої форми глобальної ідентичності, використовуються такі види потенційно корегуючих ідентифікаторів:

- Ідентифікатори облікових даних користувача та відкриті ключі облікових даних.Вони зареєстровані WebAuthn Relying Party і згодом використовуються користувачем для підтвердження власного особистого секретного ключа. Вони також видно клієнту в зв'язку з аутентифікатором
- Особисті дані користувача, що стосуються кожної Relying Party, наприклад, імена користувачів і user handles
- Біометричні характеристики користувача, наприклад, відбитки пальців або дані розпізнавання осіб
- Моделі автентифікаторів користувача, наприклад, назви продуктів. Ця інформація викладена в атестаційній заяві, наданій довіреній стороні під час реєстрації
- Ідентифікатори користувача, наприклад, серійні номери

Біометричні аутентифікатори виконують внутрішнє біометричне розпізнавання в аутентифікаторі - хоча для аутентифікаторів платформи

біометричні дані також можуть бути видимими для клієнта, залежно від реалізації. Біометричні дані не розкриваються довіряючій стороні WebAuthn. Вона використовується лише локально для виконання перевірки користувача, що дозволяє створювати та реєструвати облікові дані відкритого ключа або використовувати аутентифікацію. Отже, зловмисна сторона, не може виявити особисту ідентифікацію користувача за допомогою біометричних даних, а порушення безпеки в довіряючій стороні, не може викривати біометричні дані для зловмисника, які вони використовують для підробки вхідних даних у інших довіряючих сторін.

2.5 Криптографічні виклики та підтримка браузерів

У серпні 2018 року Paragon Initiative Enterprises зробили аудит безпеки майбутнього стандарту WebAuthn. Хоча вони не могли знайти жодних конкретних експлуатацій, вони виявили деякі серйозні недоліки в тому, як використовується криптографія, що лежить в основі, і затверджена стандартом.

Основні моменти критики обертаються навколо двох потенційних проблем, які в минулому були проблематичними в інших криптографічних системах, і тому їх слід уникати, щоб не стати жертвою одного класу атак:

Завдяки мандатному використанню COSE (RFC 8152) WebAuthn також підтримує RSA з PKCS1v1.5. Ця особлива схема заповнення, як відомо, є вразливою до конкретних атак протягом принаймні двадцяти років, і вона була успішно атакована в інших протоколах і реалізаціях криптосистеми RSA в минулому. У контексті WebAuthn важко використовувати в даних умовах, але, зважаючи на те, що існують більш безпечні криптографічні примітиви та схеми заповнення, це все ще

поганий вибір і більше не вважається найкращою практикою серед криптографів.

Альянс FIDO стандартизований на асиметричній криптографічній схемі, яка називається ECDAA. Це версія прямої анонімної атестації на основі еліптичних кривих і у випадку WebAuthn призначена для перевірки цілісності аутентифікаторів, а також збереження конфіденційності користувачів, оскільки це не дозволяє здійснювати глобальне співвідношення ручок. Однак, ECDAA не враховує деякі уроки, які були вивчені в останні десятиліття досліджень в області криптографії еліптичної кривої, оскільки обрана крива має деякі дефіцити безпеки, притаманні цьому типу кривої, що значно знижує гарантії безпеки. Крім того, стандарт ECDAA передбачає випадкові, недетерміновані підписи, які вже були проблемою в минулому.

Підприємства Paragon Initiative також критикували спосіб, у який стандарт був спочатку розроблений, так як пропозиція не була оприлюднена заздалегідь, а досвідчені криптографи не просили про пропозиції та відгуки. Тому стандарт не підлягав широкому криптографічному дослідженню академічного світу.

Незважаючи на ці недоліки, Paragon Initiative Enterprises все ще заохочує користувачів продовжувати користуватися WebAuthn, але придумали деякі рекомендації для потенційних розробників та розробників стандарту, який, на їхню думку, можуть бути реалізовані до завершення стандарту. Якнайшвидше уникнення таких помилок завадить промисловості від будь-яких викликів, які вводяться порушеними стандартами і необхідністю зворотної сумісності.

Таблиця 3.1 - Підтримка браузерів

	ВЕРСІЯ	ПІДТРИМКА WebAuthn	КОРИСТУВАЧІ, %
Chrome	4-66	Ні	2.18
	67-74	Так	29.49
	75	Так	0.06
	76 - 78	Так	0.03
Chrome for Android	74	Так	32.28
IOS Safari	3.2 - 12.1	Ні	3.28
	12.2	Ні	7.39
Firefox	2 - 59	Ні	0.73
	60 - 66	Так	3.95
	67	Так	0.13
Safari	3.1 - 12	Ні	2.17
	12.1	Ні	1.34
IE	6-10	Ні	0.46
	11	Ні	2.15
Edge	13 - 17	Можлива	1.32
	18	Так	0.78
Opera	10 - 53	Ні	0.19
	12.1	Можлива	0.64
Opera Mini	-	Ні	1.43
Android Browser	-	Ні	0.6
Firefox for Android	-	Так	0.2
UC Browser for Android	11.8	Ні	3.2
Samsung Internet	4 - 8.2	Ні	1.17
	9.2	Ні	2.47
KaiOS Browser	-	Ні	0.38

Висновки до розділу

У цьому розділі були описані основні теоретичні відомості про структуру та роботу протоколу.

Також описані процедури реєстрації та аутентифікації та структури даних аутентифікатора.

Хоча аудит показав, що у протокола існують потенційні загрози в майбутньому, автори залишили рекомендації для розробників, щоб забезпечити безпечну роботу протоколу.

Підтримка браузерів - основна вимога для реалізації та використання протокола. З таблиці можна побачити, що протокол працює у всіх основних браузерах нових версій, таких як Chrome, Edge та Firefox. Протокол ще не вдалося реалізувати в Safari IOS, який може бути основною платформою для використання безпарольної аутентифікації. Також у деяких браузерах є можливість експериментального використання протоколу з допомогою окремих прапорів доступу та зовнішніх програм.

3 МОДЕЛЮВАННЯ ПРОТОКОЛУ

Для моделювання методу безпарольної аутентифікації на основі протоколу WebAuthn я використовував:

- PyWebAuthn - модуль Python, який може бути використаний для обробки та затвердження WebAuthn. Наразі WebAuthn підтримується у Firefox, Chrome та Edge.
- Web Authentication API - API(англ. Application Programming Interface, API), що дозволяє створювати та використовувати повноцінні облікові дані, засновані на відкритих ключах, за допомогою веб-додатків для цілей аутентифікації користувачів.
- Flask - фреймворк для створення веб-додатків на мові програмування Python, що використовує набір інструментів Werkzeug, а також шаблонизатор Jinja2. Відноситься до категорії так званих мікрофреймворків.

3.1 Використання можливостей API

З допомогою методів, описаних в API змоделюю структуру аутентифікаціЇ та основних методів протоколу:

Для створення параметрів облікових даних використовую:

```
make_credential_options                                          =
webauthn.WebAuthnMakeCredentialOptions(
                    challenge,
                    rp_name,
                    rp_id,
                    user_id,
                    username,
                    display_name,
                    icon_url)
```

Створення об'єкту `WebAuthnUser`:

```
webauthn_user = webauthn.WebAuthnUser(
                    user.id,
                    user.username,
                    user.display_name,
                    user.icon_url,
                    user.credential_id,
                    user.pub_key,
                    user.sign_count,
                    user.rp_id)
```

Створення параметрів затвердження (для передачі у navigator.credentials.get)**:**

```
webauthn_assertion_options                                    =
webauthn.WebAuthnAssertionOptions(
                    webauthn_user,
                    challenge)
```

Перевірка відповіді на реєстрацію:

```
webauthn_registration_response                                =
webauthn.WebAuthnRegistrationResponse(
                    RP_ID,
                    ORIGIN,
                    registration_response,
                    challenge,
                    trust_anchor_dir,
                    trusted_attestation_cert_required,
                    self_attestation_permitted,
                    none_attestation_permitted,
                    uv_required=False)  # User Verification

try:
                                    webauthn_credential       =
webauthn_registration_response.verify()
except Exception as e:
       return jsonify({'fail': 'Registration failed. Error:
{}'.format(e)})

# Create User
```

Створення користувача та підтвердження відповіді на підтвердження (результат navigator.credentials.get):

```
webauthn_user = webauthn.WebAuthnUser(
                    user.ukey,
                    user.username,
                    user.display_name,
                    user.icon_url,
                    user.credential_id,
                    user.pub_key,
                    user.sign_count,
                    user.rp_id)


webauthn_assertion_response                                    =
webauthn.WebAuthnAssertionResponse(
                    webauthn_user,
                    assertion_response,
                    challenge,
                    origin,
                    uv_required=False)  # User Verification

try:
    sign_count = webauthn_assertion_response.verify()
except Exception as e:
         return  jsonify({'fail':  'Assertion  failed.  Error:
{}'.format(e)})

# Update counter.
user.sign_count = sign_count
```

3.2 Опис роботи протоколу:

Наведені нижче сценарії використання ілюструють використання двох дуже різних аутентифікаторів.

Реєстрація на телефоні:

1) Користувач переходить до веб-переглядача example.com у веб-переглядачі та входить до існуючого облікового запису,

використовуючи будь-який метод, який вони використовували (можливо, існуючий метод, наприклад пароль), або створює новий обліковий запис

2) У телефоні з'являється запит: "Ви бажаєте зареєструвати цей пристрій у example.com?"
3) Користувач погоджується
4) Телефон запитує користувача на попередньо налаштований жест авторизації (PIN, біометричний тощо); користувач надає це
5) Веб-сайт показує повідомлення "Реєстрація завершена"

Аутентифікація на комп’ютері:

1. Користувач з'єднує свій телефон з ноутбуком або робочим столом через Bluetooth
2. Користувач переходить до example.com у веб-переглядачі та ініціює вхід
3. Користувач отримує повідомлення від веб-переглядача "Будь ласка, завершіть цю дію на своєму телефоні"

Наступні дії з телефоном:

1) Користувач бачить дискретні підказки або сповіщення "Увійти до example.com".
2) Користувач вибирає цю підказку / сповіщення.
3) Користувач відображає список їхніх ідентифікаторів example.com, наприклад, "Увійти як Аліса / Увійти як Боб".
4) Користувач вибирає ідентифікатор, запитується про жест руху авторизації (PIN, біометричний тощо) і надає це.

Веб сторінка на комп’ютері показує, що вибраний користувач увійшов у систему, а також переходить до сторінки, на якій він підписаний.

Наступний сценарій використання показує, яким чином довірена сторона може використовувати комбінацію роумінг-автентифікатора (USB-ключа) і аутентифікатора платформи (вбудованого датчика відбитків пальців), таким чином, що користувач має:

- "первинний" роумінг-автентифікатор, який вони використовують для аутентифікації на нових клієнтських пристроях (наприклад, ноутбуках, настільних комп'ютерах) або на таких клієнтських пристроях, які не мають автентифікатора платформи
- сильна повторну аутентифікацію на клієнтських пристроях, що мають аутентифікатори платформи

По-перше, на настільному комп'ютері (без аутентифікатора платформи):

1) Користувач переходить до веб-переглядача example.com у веб-переглядачі та входить до існуючого облікового запису, використовуючи будь-який метод, який вони використовували (можливо, існуючий метод, наприклад пароль), або створює новий обліковий запис.
2) Користувач переходить до налаштувань безпеки облікового запису і вибирає "Реєстрація ключа безпеки".
3) Веб-сайт закликає користувача підключити USB-брелок безпеки; користувач.
4) Клавіша безпеки USB блимає, щоб вказати, що користувач повинен натиснути на ньому кнопку; користувач.

5) Веб-сайт показує повідомлення "Реєстрація завершена".

Пізніше на комп’ютері (який містить аутентифікатор платформи):

1) Користувач переходить до example.com у веб-переглядачі та ініціює вхід.
2) Веб-сайт пропонує користувачу підключити ключ безпеки USB.
3) Користувач підключає попередньо зареєстрований ключ безпеки USB і натискає кнопку.
4) Веб-сайт показує, що користувач увійшов до системи та переходить на сторінку, що підписана.
5) Веб-сайт підказує, "Ви хочете зареєструвати цей комп'ютер за допомогою example.com?"
6) Користувач погоджується.
7) Ноутбук пропонує користувачеві попередньо налаштований жест авторизації (PIN, біометричний тощо); користувач надає це.
8) Веб-сайт показує повідомлення "Реєстрація завершена".
9) Користувач підписується.

Пізніше, знову на комп’ютері:

1) Користувач переходить до example.com у веб-переглядачі та ініціює вхід.
2) Веб-сайт показує повідомлення "Будь ласка, виконайте вказівки вашого комп'ютера для завершення входу".
3) Ноутбук пропонує користувачеві жест доступу (PIN-код, біометричний тощо); користувач надає це.
4) Веб-сайт показує, що користувач увійшов до системи та переходить на сторінку, що підписана.

3.3 Зовнішній вид та UX

Поліпшення UX і UI не створює проблем для кібербезпеки. Користувачі неохоче дотримуються заходів безпеки, які заважають їм насолоджуватися своєю роботою та іншими веб-переглядами. Працівники, чиї компанії наполягають, наприклад, на тому, що вони не відвідують певні веб-сайти, можуть відчувати обмеженість і бачитимуть стратегію безпеки як неприємність, а не допомогу.

Таким чином, заходи кіберзахисту повинні брати до уваги вимоги UX. Засоби та програмне забезпечення безпеки мають бути простими, досить чіткими та плавно інтегруватися з регулярним інтерфейсом користувача та робочим процесом.

Рисунок 3.1 - Зовнішній вигляд програми

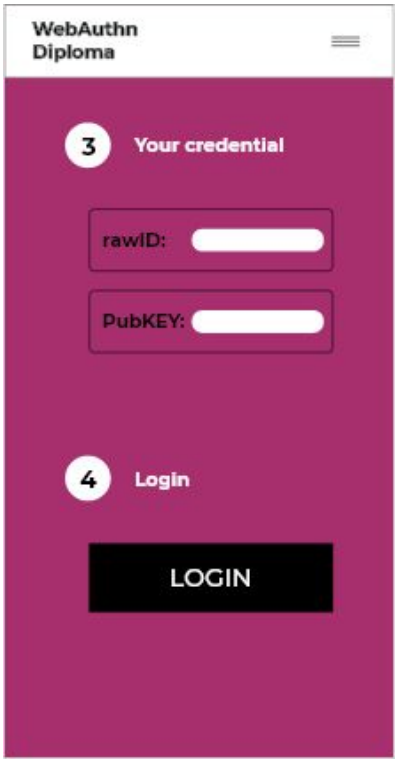

Рисунок 3.2 - Зовнішній вигляд програми

Рисунок 3.3 - Зовнішній вигляд програми

Висновки до розділу

В цьому розділі була описана структура програмного коду та основні функції для моделювання роботи безпарольної аутентифікації на основі протоколу WebAuthn з допомогою API.

Окремо був описаний алгоритм для реєстрації користувача з отриманням токенів у системі та подальшої аутентифікації з отриманим ключем.

Спроектований сайт має бути зручним та простим у використанні для користувачів. Також варто розробити окремий розділ для ознайомлення з технологією.

Досвід користувачів не є важливим лише для компаній B2C, як Apple. Це однаково важливо для продуктів і послуг B2B, таких як програмне забезпечення компанії.

Завдяки UX для забезпечення кібербезпеки багато поставлено на карту. Безпека компаній, крім успіху самого програмного забезпечення, значною мірою залежить від простоти використання програмного забезпечення.

ВИСНОВКИ

Дана робота присвячена моделюванню універсального методу безпарольної аутентифікації на основі протоколу WebAuthn. Були розглянуті основні відомості про основні способи, принципи роботи, види та фактори аутентифікації.

Були розглянуті сучасні методи реєстрації та аутентифікації користувача у системі та принципи роботи протоколу WebAuthn. Також були розглянуті криптографічні проблеми протоколу.

Часто пароль - це все, що лежить між шкідливим користувачем і нашими банківськими рахунками, обліковими записами соціальних медіа та іншими конфіденційними даними.

Паролі створюють ряд нових проблем. По-перше, це повторне використання однакових паролів паролів на декількох різних веб-сайтах. Інша проблема - це фішинг. Зловмисники змушують користувачів вводити дані своїх облікових записів на підроблених веб-сайтах. Історично ці питання були актуальними для розробників, користувачів та хакерів, тому що велику роль тут грає людський фактор.

Якщо певний користувач, якого ми назвемо Алісою має акаунт на 50 різних веб-сайтах, паролі на 20% сайтів будуть співпадати. Це значить, що якщо зловмисник знайде пароль для 1 сайту, він отримає доступ ще до 9 інших.

Користувачі повинні турбуватися про те, що паролі викрадаються з допомогою фішингових інструментів. Згідно з іншим дослідженням, протягом періоду лише одного року, викрадаються всього майже 2 мільярди логінів та паролів користувачів. Також відомо, що 12,4 мільйонів

користувачів стали жертвами фішингу за 2018 рік. Ось чому рекомендується використовувати ключі безпеки.

Аутентифікація без паролів забезпечує зручність та додаткову безпеку, особливо для великих організацій, які не можуть відстежувати реєстраційні дані користувачів мобільних пристроїв.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Web Authentication API [Електронний ресурс]: - Режим доступу до ресурсу: https://developer.mozilla.org/
2. Web Authentication:An API for accessing Public Key Credentials [Електронний ресурс]: - Режим доступу до ресурсу: https://www.w3.org/
3. Стандарт WebAuthn офіційно завершений [Стаття]: - Режим доступу до ресурсу: https://habr.com/ru/company/1cloud/blog/445534/
4. Pressrelease-webauthn [Стаття]: - Режим доступу до ресурсу: https://www.w3.org/2019/03/pressrelease-webauthn-rec.html
5. Yubico’s 2019 State of Password and Authentication Security Behaviors Report [Електронний ресурс]: - Режим доступу до ресурсу: https://www.yubico.com/press-releases/yubicos-2019-state-of-password-and-authentication-security-behaviors-report/
6. Стандарт Web Authentication API:[Електронний ресурс]: - Режим доступу до ресурсу: https://habr.com/globalsign/blog/358622/
7. Нові стандарти для беспарольної аутентифікації [Електронний ресурс]: - Режим доступу до ресурсу: https://habr.com/1cloud/blog/353966/
8. Cyber Security requires strong UX [Електронний ресурс]: - Режим доступу до ресурсу: https://hackernoon.com/cyber-security-requires-an-important-ingredient-strong-ux-d0727a0c076
9. The challenge on doing good UX [Електронний ресурс]: - Режим доступу до ресурсу:

https://uxdesign.cc/the-challenge-on-doing-good-ux-on-cybersecurity-startups-3b747079def1

10. UX design for cybersecurity [Електронний ресурс]: - Режим доступу до ресурсу: https://medium.com/@MatthewDoan/
11. Federated identity managment [Електронний ресурс]: - Режим доступу до ресурсу: https://searchsecurity.techtarget.com/definition/federated-identity-management
12. Google I/O ‘18 [Відео]: - Режим доступу до ресурсу: https://www.youtube.com/watch?v=kGGMgEfSzMw&t=95s
13. FIDO U2F [Електронний ресурс]: - Режим доступу до ресурсу: https://www.yubico.com/solutions/fido-u2f/
14. Authentication essentials [Електронний ресурс]: - Режим доступу до ресурсу: https://searchsecurity.techtarget.com/definition/authentication
15. FIDO2 Beyond passwords [Електронний ресурс]: - Режим доступу до ресурсу: https://www.youtube.com/watch?v=CkJf8zv1yBw

ДОДАТОК А Тексти програмного коду

app.py

```
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///{}'.format(
    os.path.join(os.path.dirname(os.path.abspath(__name__)), 'webauthn.db'))
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
sk = os.environ.get('FLASK_SECRET_KEY')
app.secret_key = sk if sk else os.urandom(40)
db.init_app(app)
login_manager = LoginManager()
login_manager.init_app(app)

RP_ID = 'localhost'
ORIGIN = 'https://localhost:5000'

TRUST_ANCHOR_DIR = 'trusted_attestation_roots'


@login_manager.user_loader
def load_user(user_id):
    try:
        int(user_id)
    except ValueError:
        return None

    return User.query.get(int(user_id))


@app.route('/')
def index():
    return render_template('index.html')


@app.route('/webauthn_begin_activate', methods=['POST'])
def webauthn_begin_activate():
    # MakeCredentialOptions
    username = request.form.get('register_username')
    display_name = request.form.get('register_display_name')

    if not util.validate_username(username):
        return make_response(jsonify({'fail': 'Invalid username.'}), 401)
    if not util.validate_display_name(display_name):
        return make_response(jsonify({'fail': 'Invalid display name.'}), 401)

    if User.query.filter_by(username=username).first():
        return make_response(jsonify({'fail': 'User already exists.'}), 401)

    if 'register_ukey' in session:
        del session['register_ukey']
    if 'register_username' in session:
        del session['register_username']
    if 'register_display_name' in session:
        del session['register_display_name']
```

```
    if 'challenge' in session:
        del session['challenge']

    session['register_username'] = username
    session['register_display_name'] = display_name

    rp_name = 'localhost'
    challenge = util.generate_challenge(32)
    ukey = util.generate_ukey()

    session['challenge'] = challenge
    session['register_ukey'] = ukey

    make_credential_options = webauthn.WebAuthnMakeCredentialOptions(
        challenge, rp_name, RP_ID, ukey, username, display_name,
        'https://example.com')

    return jsonify(make_credential_options.registration_dict)


@app.route('/webauthn_begin_assertion', methods=['POST'])
def webauthn_begin_assertion():
    username = request.form.get('login_username')

    if not util.validate_username(username):
        return make_response(jsonify({'fail': 'Invalid username.'}), 401)

    user = User.query.filter_by(username=username).first()

    if not user:
        return make_response(jsonify({'fail': 'User does not exist.'}), 401)
    if not user.credential_id:
        return make_response(jsonify({'fail': 'Unknown credential ID.'}),
401)

    if 'challenge' in session:
        del session['challenge']

    challenge = util.generate_challenge(32)

    session['challenge'] = challenge

    webauthn_user = webauthn.WebAuthnUser(
        user.ukey, user.username, user.display_name, user.icon_url,
        user.credential_id, user.pub_key, user.sign_count, user.rp_id)

    webauthn_assertion_options = webauthn.WebAuthnAssertionOptions(
        webauthn_user, challenge)

    return jsonify(webauthn_assertion_options.assertion_dict)


@app.route('/verify_credential_info', methods=['POST'])
def verify_credential_info():
    challenge = session['challenge']
    username = session['register_username']
    display_name = session['register_display_name']
    ukey = session['register_ukey']
```

```
    registration_response = request.form
    trust_anchor_dir = os.path.join(
        os.path.dirname(os.path.abspath(__file__)), TRUST_ANCHOR_DIR)
    trusted_attestation_cert_required = True
    self_attestation_permitted = True
    none_attestation_permitted = True

    webauthn_registration_response = webauthn.WebAuthnRegistrationResponse(
        RP_ID,
        ORIGIN,
        registration_response,
        challenge,
        trust_anchor_dir,
        trusted_attestation_cert_required,
        self_attestation_permitted,
        none_attestation_permitted,
        uv_required=False)  # User Verification

    try:
        webauthn_credential = webauthn_registration_response.verify()
    except Exception as e:
        return jsonify({'fail': 'Registration failed. Error: {}'.format(e)})
    credential_id_exists = User.query.filter_by(
        credential_id=webauthn_credential.credential_id).first()
    if credential_id_exists:
        return make_response(
            jsonify({
                'fail': 'Credential ID already exists.'
            }), 401)

    existing_user = User.query.filter_by(username=username).first()
    if not existing_user:
        if sys.version_info >= (3, 0):
            webauthn_credential.credential_id = str(
                webauthn_credential.credential_id, "utf-8")
        user = User(
            ukey=ukey,
            username=username,
            display_name=display_name,
            pub_key=webauthn_credential.public_key,
            credential_id=webauthn_credential.credential_id,
            sign_count=webauthn_credential.sign_count,
            rp_id=RP_ID,
            icon_url='https://example.com')
        db.session.add(user)
        db.session.commit()
    else:
        return make_response(jsonify({'fail': 'User already exists.'}), 401)

    flash('Successfully registered as {}.'.format(username))

    return jsonify({'success': 'User successfully registered.'})


@app.route('/verify_assertion', methods=['POST'])
def verify_assertion():
    challenge = session.get('challenge')
```

```
    assertion_response = request.form
    credential_id = assertion_response.get('id')

    user = User.query.filter_by(credential_id=credential_id).first()
    if not user:
        return make_response(jsonify({'fail': 'User does not exist.'}), 401)

    webauthn_user = webauthn.WebAuthnUser(
        user.ukey, user.username, user.display_name, user.icon_url,
        user.credential_id, user.pub_key, user.sign_count, user.rp_id)

    webauthn_assertion_response = webauthn.WebAuthnAssertionResponse(
        webauthn_user,
        assertion_response,
        challenge,
        ORIGIN,
        uv_required=False)  # User Verification

    try:
        sign_count = webauthn_assertion_response.verify()
    except Exception as e:
        return jsonify({'fail': 'Assertion failed. Error: {}'.format(e)})

    # Update counter.
    user.sign_count = sign_count
    db.session.add(user)
    db.session.commit()

    login_user(user)

    return jsonify({
        'success':
        'Successfully authenticated as {}'.format(user.username)
    })


@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('index'))


if __name__ == '__main__':
    app.run(host='0.0.0.0', ssl_context='adhoc', debug=True)
```