МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна технологія підвищення рівня безпеки приладів інтернету речей»

Завідувач випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Великодний Д.В.**

Студента групи ІН.м – 81н **Токмань С.В.**

СУМИ 2020

Форма №У-9.01*

Затв. наказом Мінвузу УРСР

від 3 серпня 1984р. №253

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*

Спеціальність *«Інформатика»*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Токмань Сергій Валерійович
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна технологія підвищення рівня безпеки приладів інтернету речей*
затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми. Постановка задачі дослідження. 2) Дослідження вразливостей в безпеці приладів IoT. 3) Розробка інформаційного та програмного забезпечення для підвищення рівня безпеки приладів IoT.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____
_____
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Аналіз проблеми. Постановка задачі дослідження*		
2.	*Дослідження вразливостей в безпеці приладів IoT.*		
3.	*Розробка інформаційного та програмного забезпечення для підвищення рівня безпеки приладів IoT.*		
4.	*Оформлення пояснювальної записки до дипломної роботи*		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 79 стор., 19 рис., 1 додаток, 14 джерел.

Мета роботи — розробка мобільного додатку, графічний інтерфейс якого дозволяє проводити авторотацію паролів для приладів інтернету речей .

Об'єкт дослідження — прилади інтернету речей.

Предмет дослідження — протоколи безпеки в мережі інтернету речей.

Методи дослідження — моделювання в графічному емуляторі мереж GNS3.

Результати — розроблено мобільний кросплатформний додаток, графічний інтерфейс якого дозволяє отримати доступ до налаштувань роутера за біометричними даними, автоматично змінити паролі авторизації до панелі адміністратора роутера, та проводити авторотацію замінених даних. Додаток створено з використанням мови програмування C# та багатоплатформового інструменту для розробки дво- та тривимірних додатків Unity.

IOT, ІНТЕРНЕТ РЕЧЕЙ, СИСТЕМА БЕЗПЕКИ, ЗАХИСТ ІНФОРМАЦІЇ, АВТЕНТИФІКАЦІЯ, ШИФРУВАННЯ, РОУТЕР

ЗМІСТ

ВСТУП

Інтернет речей (Internet of Things - IoT) включає в себе відразу кілька явищ. Це самі пристрої, які вийшли в мережу і взаємодіють між собою. Це і спосіб підключення - M2M - тобто машини-до-машині, без участі людини. Це і великі дані, які тепер генерують пристрої. Дані, які можна (і потрібно) збирати, аналізувати і надалі використовувати для підвищення комфорту або прийняття бізнес-рішень.

До кінця 2018 року кількість підключених IoT (Internet of things)-пристроїв перевищила 22 мільярди. В 2020 році кількість приладів сягне 31 мільярду. А за прогнозами провідних компаній з безпеки інтернету до 2027 року буде підключено до інтернету 75 мільярдів приладів IoT. Із 7,6 мільярдів чоловік на Землі у 4 мільярдів є доступний до інтернету. Отже, на кожну людину приходиться по 7,75 пристроїв IoT.

У середньому між часом підключенням пристроїв IoT до мережі і в часом першої атаки проходить близько 5 хвилин. Велика частина атак на «розумні» пристрої відбувається автоматизовано. Цифрова безпека повинна бути розроблена в пристрої з нуля і на всіх точках екосистеми, щоб запобігти уразливості в як кожного окремого елемента так і всієї системи вцілому.

В топ 10 головних вразливостей приладів IoT за версією OWASP (Open Web Application Security Project) увійшли такі пункти: небезпечні налаштування за замовчуванням; небезпечна передача і зберігання даних; небезпечні мережеві сервіси, а топ 1 проблема це слабкий, вгадуваний або жорстко заданий пароль.

Типовий прилад IoT зазвичай не обладнаний портом Ethernet, його підключення до інтернету виконується методом під’єднання до роутера, що також є приладом IoT. У середньому один роутер може одночасно мати близько 100 підключень, а отже його безпека це важлива і вразлива складова.

1. АНАЛІТИЧНИЙ ОГЛЯД

Даний розділ присвячений огляду важливих критеріїв безпеки приладів IoT, складання фундаментального списку функціональних можливостей, виділення сильних і слабких сторін вже розробленого програмного забезпечення. Також був проведений аналіз існуючих пакетів і програм перед фазою розробки необхідних алгоритмів, що дозволяє скоротити час на розробку програмних і графічних інтерфейсів та сфокусуватися на поставленій меті

1.1 Огляд існуючих рішень

Зростання кількості приладів IoT несе в собі ряд переваг, так як воно змінить спосіб виконання людьми повсякденних завдань і потенційно перетворить світ. Складова розумного дому така як розумне освітлення може реально зменшити загальне споживання енергії і знизити ваш рахунок за електрику. Нові розробки дозволять автомобілям з'єднатися з інфраструктурою «розумного міста», щоб створити абсолютно іншу екосистему для водія, який просто звик до традиційного способу дістатися з точки А в точку Б. Підключені медичні прилади дають людям глибший і повний погляд на їх власне здоров'я, або відсутність таких, ніж будь-коли раніше. Переваги IoT незаперечні; і тим не менше, гучні атаки в поєднанні з невизначеністю щодо передового досвіду безпеки і пов'язаних з ними витрат утримують багато підприємств від впровадження технології.

Серед проблем інтернету речей таких як формат даних, зв'язок, нехватка кадрів, страх людей перед неконтрольованою автоматизацією найважливішою є безпека, безпека особистих даних, навіть безпека здоров’я [1].

До недавнього часу комп'ютери для автомобілів, верстати, роботи створювалися з урахуванням ізольованою середовища. У цьому сенсі вони були схожі на персональні комп'ютери до широкого поширення інтернету - тоді користувачів мало турбували комп'ютерні віруси. Всілякі машини лише

недавно отримали доступ в інтернет, а з ним прийшли і проблеми інформаційної безпеки.

Наслідки хакерських атак на атомну електростанцію, нафтову свердловину можуть бути незворотними. Проблема настільки серйозна, що розглядається на найвищому рівні. У звіті Всесвітнього економічного форуму йдеться, що вироблення єдиного підходу до вирішення проблеми безпеки - найнеобхідніший крок для розвитку інтернету речей.[2]

Основними напрямами безпеки в інтернеті речей є:

1. Безпека зв'язку

Перевірка автентичності паралельно технологіям шифрування використовуються для захисту каналу зв’язку, щоб в результаті прилади знали, що дані отримані від віддаленої системи валідні. Нові криптографічні технології, такі як ECC (Elliptic Curve Cryptography), працюють в десять разів краще попередників в слабких чипах IoT 8-bit 8MHz [5]

Динамічна маршрутизація, постійні зміни каналу зв’язку використовуються як інструмент приховування магістралей передачі інформації від третіх осіб, але в той же час це вносить нові проблеми та мінуси, адже не всі пристрої, через які передаються або отримують дані, є безпечними. Як тільки дані просочилися, хакери можуть продати їх іншим компаніям, які порушують права на конфіденційність і безпеку даних. Крім того, навіть якщо дані не просочилися з боку споживача, постачальники послуг можуть не відповідати правилам і законам. Це також може привести до інцидентів безпеки.

2. Захист пристроїв

Пристрої IoT вимагають унікальної ідентифікації, яка може бути використана для перевірки автентичності при спробі підключення до шлюзів або систем бекенда. За допомогою цього унікального ідентифікатора

виробники можуть також забезпечити довіру протягом усього життєвого циклу продукту, а також анулювати або замінити облікові дані для зниження ризику.

3. Контроль пристроїв

Оскільки пристрої IoT використовуються все частіше, виробники цих пристроїв зосереджені на створенні нових і не приділяють достатньої уваги безпеці. Більшість з цих пристроїв не отримують достатньо оновлень, в той час як деякі з них ніколи не отримують жодного. Це означає, що ці продукти є безпечними під час покупки, але стають уразливими для атак, коли хакери знаходять деякі помилки або проблеми безпеки.

Коли ці проблеми не усунуті шляхом випуску регулярних оновлень для апаратного і програмного забезпечення, пристрої залишаються уразливими для атак. Для кожної дрібниці, підключеної до Інтернету, регулярні оновлення є обов'язковими. Відсутність оновлень може привести до витоку даних не тільки клієнтів, але і компаній, які їх виробляють

Не дивлячись на всю ідеальність системи вразливості в пристроях IoT все одно будуть, їх потрібно буде патчити, і це може відбуватися протягом тривалого часу після передачі обладнання споживачеві. Навіть код із застосуванням обфускації в критичних системах зрештою реконструюється, і зловмисники знаходять в ньому вразливості. Ніхто не хоче, а часто і не може відправляти своїх співробітників для очного візиту до кожного пристрою IoT для оновлення прошивки, особливо, якщо мова йде, наприклад, про парк вантажівок або про мережу датчиків контролю, розподілених на сотні кілометрів. З цієї причини «керованість по повітрю» (over-the air, OTA), повинна бути вбудована в пристрої до того, як вони потраплять до покупців.

4. Використання слабких облікових даних.

Багато компаній IoT продають пристрої та надають споживачам облікові дані за замовчуванням з такі як ім'я користувача адміністратора,

стандартній пароль. Хакерам достатньо саме цих даних, фактично два слова, це ключ до взлому безпеки. Коли вони знають тільки ім'я користувача, вони здійснюють атаки грубої сили, щоб отримати несанкціонований доступ.

З іншого боку, безпека програмного забезпечення IoT також є серйозною проблемою для розробників додатків. Є декілька основних пунктів які необхідно враховувати при розробці програмного забезпечення для пристроїв IoT:

1. Фізична безпека

Пристрої IoT зазвичай не мають великого значення як в очах користувачів так і виробників, що означає, що вони можуть бути легко атаковані хакерами. Отже, необхідність наявності окремого компонента безпеки для програмного забезпечення цих пристроїв є одним з головних завдань при розробці.

2. Безпека хмарних сховищ

Оскільки передача даних з пристроїв IoT зазвичай зав’язана на хмарному сховищі дуже важливо забезпечити шифрування даних або використання надійних алгоритмів хешування. Тим самим не дозволяючи отримати доступ до інформації навіть у разі взлому хмарного сховища.

Програмне забезпечення для приладів IoT це відносно мала програма з обмеженим функціоналом. Тому вихідний код, як правило, пишеться на мовах «низко рівневих» - C або C++,, які часто стають жертвами загальних проблем, таких як витік пам'яті і уразливості переповнення буфера. У середовищі IoT витоки пам’яті можуть розмножуватися, фактично впливати на інші пристрої. Найкращий захист тут полягає в тестуванні, тестуванні і повторному тестуванні. Існує безліч добре продуманих інструментів тестування на ринку, які були використані для пристроїв IoT, але таке тестування часто не проводиться.

Атака ботнету Mirai є прикладом, який був здійснений, бо пристрою використовували облікові дані за замовчуванням[10]. Проблема полягає в тому, що пристрої IoT дуже уразливі для атак шкідливих програм. Вони не мають регулярних оновлень безпеки програмного забезпечення, що наприклад має типовій персональний комп'ютер. Таким чином, вони швидко перетворюються в інфікованих зомбі і використовуються в якості зброї для відправки неймовірно величезної кількості трафіку.

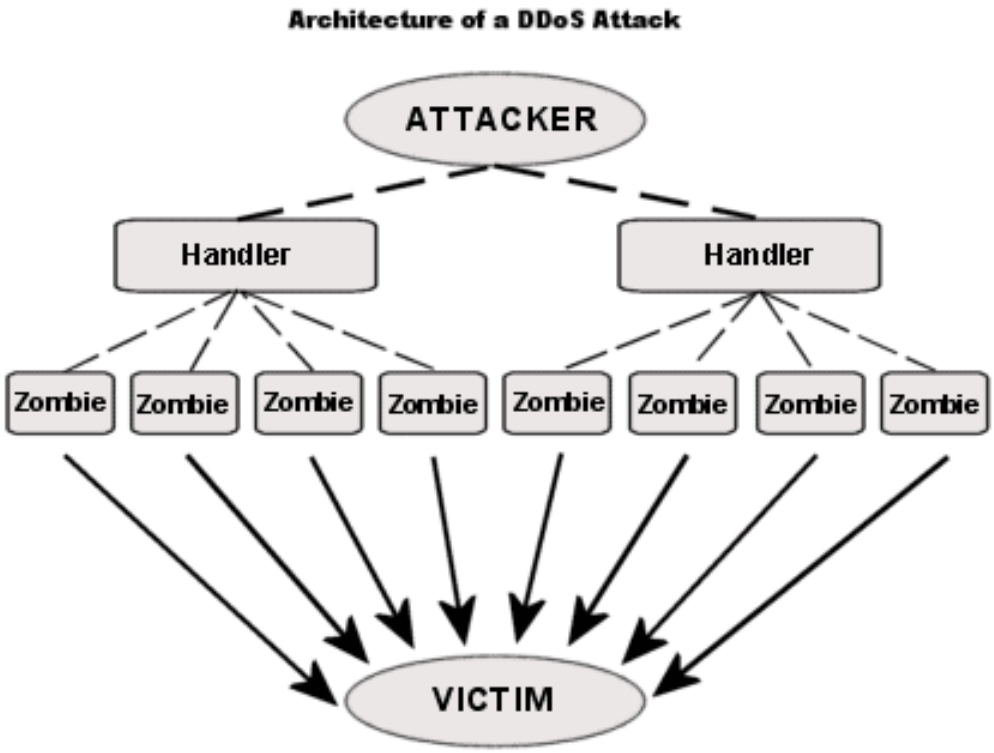

Рисунок 1.1 – Схема атак з використанням IoT [10]

Більш того, ботнет може становити загрозу безпеці для електричних мереж, виробничих підприємств, транспортних систем і водоочисних споруд, що може загрожувати великим групам людей. Наприклад, хакер може одночасно запустити систему охолодження та опалення, створюючи шипи на електромережі; в разі великомасштабної атаки, хакери можуть створити загальнонаціональне відключення електроенергії.

Споживачі повинні змінювати стандартні облікові дані , як тільки вони отримують пристрій, але більшість виробників нічого не говорять в інструкції користувача про внесення таких зміни.

Отже найросповсюдженішою проблемою безпеки інтернету речей є слабкі паролі. Використання легко зламуваних, загальнодоступних або незмінних облікових даних, включаючи бекдори у вбудованому програмному забезпеченні або клієнтському програмному забезпеченні, надає несанкціонований доступ до розгорнутих систем[3].

Впровадження філософії «нульового довіри», коли мова йде про IoT-мережі, є важливим кроком до забезпечення безпеки IT-систем та інтернет-інфраструктури. Традиційна мережева архітектура це просте правило , що будь-який пристрій або користувач, який отримав доступ до мережі, має право перебувати там. Це дозволяє неавторизованих користувачам переміщатися по системі після того, як вони порушують зовнішні брандмауери. Це особливо небезпечно в тих випадках, коли мова йде про пристрої IoT, оскільки багато хто з них є вразливими до використанні в якості шлюзу в IT-системах. Філософія безпеки нульового довіри реалізує набагато сильніші елементи управління аутентифікацією, починаючи з припущення, що всі в мережі потенційно може бути скомпрометовано, а потім вимагаючи більш суворої перевірки для будь-якого запиту доступу. Це запобігає бічний рух, оскільки навіть якщо пристрій IoT скомпрометовано, зловмисникові буде важче перейти в різні частини мережі.

1.2 Аналоги готових продуктів

Користувач придумавши надійний пароль, часто починає використовувати його всюди, оскільки запам'ятати кілька унікальних паролів практично неможливо. У кращому випадку набір надійних символів нумерується відповідно для різних ресурсів. Такий підхід є невірним і

небезпечним. У разі, якщо хтось отримає доступ до паролю, доступні стануть всі ресурси, на яких було проведено реєстрацію. Звичайно, придумувати і записувати всі паролі на стікерах дуже незручно і прямо скажемо не надійно. Зрозуміло пароль можна десь записати, але зберігати його варто в надійному місці, куди ніхто не матиме доступ.

Кожен важливий ресурс повинен мати унікальний пароль. При цьому бажано зберігати його як мінімум в 2 примірниках, на всяк випадок. Отже головне питання як часто потрібно змінювати паролі та як їх зберігати

Прості паролі з невеликою кількістю символів (до 8-10) варто міняти один раз в кілька місяців. На деяких сервісах (особливо платіжних системах) зміна пароля проводиться в примусовому порядку. Тобто після закінчення певного періоду, скажімо року, сайт пропонує вам змінити стару комбінацію на нову.

Надійні паролі з 12-16 символів можна залишати на кілька років. Однак найкраще міняти їх хоча б один раз в 12-18 місяців. Це не дуже складно і гарантує безпеку даних, а найголовніше засобів, що особливо актуально коли мова йде про банківські рахунки і електронних гаманцях.

Замість того, щоб записувати ваші паролі на папері, де їх хтось може знайти, ви можете використовувати менеджер паролів для безпечного зберігання в інтернеті. Менеджер паролів зберігає ваші паролі в надійному сховищі, яке можна розблокувати за допомогою одного головного пароля і, за бажанням, додатковий двохфакторну метод аутентифікації, щоб допомогти зберегти все додатково в безпеці.

Менеджери паролів дозволяють використовувати надійні, унікальні паролі в усьому світі. Деякі екземпляри можуть генерувати вводити пароль на різних сайтах, а це значить, що вам не потрібно вигадувати і

запам'ятовувати довгі паролі. Приклади менеджерів паролів включені в LastPass, 1Password і менеджер паролів Google Chrome.

1. Google Chrome

Google Chrome може зберігати паролі, які ви вводите при вході на різних сайтах. Щоб збережені паролі були доступні на всіх ваших пристроях, необхідно увімкнути синхронізацію в Chrome. Те, де Chrome зберігає паролі, залежить від того, чи включили ви їх синхронізацію. Якщо так, то паролі можна використовувати в Chrome на всіх пристроях і в деяких додатках для Android. Коли в Chrome увімкнена синхронізація паролів, вони зберігаються в акаунті Google. Якщо синхронізація відключена, паролі зберігаються тільки в браузері Chrome на вашому комп'ютері

Недоліки

Відсутня ротація паролів. При відсутності інтернету паролі зберігаються локально. Неможливість використовувати програмне забезпечення на більшості приладів інтернету речей

2. LastPass

LastPass - один з провідних менеджерів паролів, якими користуються 17,8 мільйона людей у всьому світі. Деякі з плюсів LastPass включають функціональність паролів, що додаються легко, спрощений обмін, аудит паролів та генератор паролів у два кліки. Все, що потрібно зробити користувачу, це запам’ятати свій головний пароль LastPass, і LastPass автоматично заповнить будь-які форми веб паролів після його введення.

Недоліки

Відсутня ротація паролів. Неможливість використання без доступу до інтернету бо інформація про паролі зберігається в хмарі. Зберігання паролів в «хмарі» пов'язане з деякою небезпекою. Так, в разі злому аккаунта на сайті, що набагато простіше ніж отримати доступ до додатка на локальному ПК, всі

паролі стануть доступніші зловмисникові . При цьому сам факт довіри своїх паролів сторонньому сервісу, може привести до ще більшого ризику . В якості рекомендацій, досвідчені користувачі радять, паролі особливої важливості зберігати в LastPass в спеціально зміненому вигляді, наприклад ставити зайві символи.

3. 1Password

Вміє працювати без підключення до мережі і синхронізувати сховище через мережеві папки, Wi-Fi або «хмари» (Dropbox і iCloud). Ви можете налаштувати доступ для інших користувачів або вказати довірені контакти. Можуть бути налаштовані сповіщення про взлом використовуваних ресурсів, тобто якщо база користувачів отримала публічний доступ програма повідомить про це і запропонує змінити пароль.

Недоліки

Хоч на перший погляд коло довірених осіб здається зручним доповненням до програми, але в той же час це шлюз для атак методами соціальної інженерії. Програма не повністю кросплатформна і не може працювати з системами Linux, яка є операційною системою приблизно 70% світових серверів. Відсутність авторотації паролів.

Сервіси що дають змогу зберігати паролі загалом мають схожі недоліки. Вузько направлену сферу використання-часто лише для профілів і інтернеті, зовсім унеможливлюючи використання в області інтернету речей. Відсутність можливості ротації паролів або їх генерації. Існують випадки трансферу паролів через незахищені шифруванням протоколами.

Як виявилося, парольні менеджери часом зберігають паролі у вигляді відкритого тексту, а в їхньому коді можна виявити ключі шифрування, які там, очевидно, забули випадково. У деяких випадках дістатися до паролів користувача без особливих зусиль зможе будь-який шкідливий додаток, встановлений на пристрій.

Основний недолік менеджерів паролів це несумісність с приладами IoT. Фактично їх функціонал обмежується функціоналом простої бази даних без ротації та автоматичної зміни даних. Доступ до даних автоматично надається всім хто має фізичний доступ до пристрою, що дає змогу проводити атаки в основу яких покладено методи соціальної інженерії.

Такі недоліки обумовлюються тягою видавця до охоплення найбільшої частини ринку і не зацікавленість в вузько направлених продуктах через складний етап розробки, необхідність постійного оновлення (частішого чим в разі продукту для типового користувача) та малу потенційну базу клієнтів.

1.3 Вибір методу вирішення задачі

Для вирішення задачі необхідно створити кросплатформне програмне забезпечення з єдиним центром керування паролями. Платформою для механізму керування може бути операційна система(ОС) Android. Android - це не телефон чи додаток, це комплекс програм, що забезпечує управління апаратними засобами прилад, що організує роботу з файлами і виконання прикладних програм, що здійснює введення і виведення даних, заснована на ядрі Linux. У своєму найпростішому визначенні Linux - це операційна система, яка найчастіше зустрічається на серверах та настільних комп’ютерах. Android не є просто версією Linux через велику кількість змін, що буди внесені в ядро. Найсильніший рівень захисту - сама операційна система. Додатки для Android працюють в ізольованій програмному середовищі, яке обмежує доступ до вашої інформації іншим додаткам. Компоненти ОС також знаходяться під захистом, що не дає зловмисникам використовувати системні помилки в своїх цілях. Так як всі дані на пристрої шифруються, вони залишаться в безпеці, навіть якщо ви втратите телефон.

Основні протоколи керування приладами інтернету речей для даної системи це telnet та ssh.

Протокол SSH (також званий Secure Shell) - це метод безпечного віддаленого входу з одного комп'ютера на інший.

Telnet - це мережевий протокол, який використовується для віртуального доступу до комп'ютера і забезпечення двостороннього, спільного і текстового каналу зв'язку між двома пристроями[11].

Призначення протоколу telnet у наданні достатньо спільного, двонаправленого, восьмибітового байт-орієнтованого засобу зв'язку. Його основне завдання полягає в тому, щоб дозволити термінальним пристроям і термінальним процесам взаємодіяти один з одним. Передбачається, що цей протокол може бути використаний для зв'язку виду термінал-термінал («зв'язування») або для зв'язку процес-процес («розподілені обчислення»).

В основу більшості приладів інтернету речей закладена операційна системі Linux. Експерти погоджуються, що Linux - найбезпечніша ОС. Це найпопулярніша ОС на серверах. Але якщо говорити про корпоративне використанні, то компанії не поспішають її впроваджувати для співробітників. Головна риса безпеки Linux - відкритий код. Те, що програмісти можуть читати і коментувати код один одного, може здатися небезпечним. Але це не так. Саме те, що у Linux відкритий код, робить його найбезпечнішою ОС. Кожен може переглянути код і переконатися, що в ньому немає багів або секретних ходів - все співтовариство програмістів забезпечує безпеку Linux.

Katherine Noyes пояснює: «Microsoft може скільки завгодно хвалитися своєю армією найнятих програмістів, але навряд чи їх команда зрівняється зі всесвітньої базою програмістів-користувачів Linux» [3].

Інший фактор безпеки Linux, про який говорить PC World: у Linux краще налаштована модель дозволів. На Windows користувачам зазвичай дають доступ адміністратора за замовчуванням. На Linux навпаки: там рутовий доступ сильно обмежений. Третій фактор безпеки Linux - кількість

дистрибутивів. Деякі з них спеціально розроблені таким чином, щоб протистояти специфічним атакам.

Для безпечної процедури збереження та передачі паролів їх необхідно шифрувати, хешувати, адже передача через мережу інтернет паролю у відкритому вигляді це перша помилка в розробці схожих систем. Для вирішення задачі з хешування обрано алгоритм SHA-256 (Secure Hash Algorithm - безпечний алгоритм хешування). SHA-256 - одна з наступних хеш-функцій для SHA-1 (спільно їх називають SHA-2) і є однією з найсильніших доступних хеш-функцій. SHA-256 не є набагато складнішим кодом, ніж SHA-1, і досі жодним чином не піддається компрометації.

Хешовані паролі не є чимось унікальним через детерміновану природу функції хеша: при однаковому вході проводиться один і той же вихід завжди. Щоб нівелювати шкоду, яку може завдати райдужна таблиця або атака словника, в правила хеш функції додають так звану сіль. Згідно принципам OWASP, сіль являє собою фіксований криптографічне випадкове значення, яке додається до вхідних даних хеш-функцій для створення унікальних хешів для кожного введення, незалежно від того, що вхідні дані не унікальний. Сіль робить функцію хеша недетермінованою, що допомагає приховати кореляцію частин даних, що можуть бути використаними для взлому.

Для того щоб підтримувати можливість відновлення доступу до втрачених даних через забутий або скомпрометований пароль в якості солі використовується оброблена інформація отримана з відбитку пальця. Тобто сіль кожного можна створити у відповідності до відбитку та отримати хеш який є максимально персоналізованим, але в той же час це дозволяє звернутися до виробника і після підтвердження своєї особи досить швидко повернути доступ до своїх даних.

Обє’днавши вище описані системи в одну отримуємо систему здатну працювати з більшістю девайсів інтернету речей, з функціоналом менеджера та генератора паролів та з високим рівнем захищеності передачі та локального збереження паролів.

1.4 Постановка задачі

Для вирішення проблеми потрібно створити програмний додаток який зможе контролювати безпеку роутера за допомогою зміни його паролю для авторизації в адмінпанель та для доступа до wi-fi. Для створення додатка необхідно вирішити проблему аутентифікації через протоколи telnet та ssh. На деяких девайсах порти що відповідають за ці протоколи закриті, тому авторизація є неможлива. Є два шляхи вирішення даної проблеми. Перший це просто обмежити діапазон роутерів що будуть підтримуватись додатком. Другий це використовувати програмне забезпечення типу OpenWrt та DD-WRT для роутера , що дозволить використовувати порт 23 який і відповідає за telnet/ssh. Також необхідно виконати реалізацію терміналу керування на базі Android/IOS. Тобто розробити систему введення та передачі команд до роутера через wi-fi та створити інтуітивно зрозумілий інтерфейс керування який в свою чергу давав доступ до перегляду всіх логів, змін налаштувань девайсу. Додатково необхідно розробити функцію автопідключення до точки доступу (роутера) з зміненими налаштуваннями та функцію швидкого розповсюдження нового паролю доступу. Механізмом який дозволить передавати інформацію про зміни у паролі можу бути технологія генератора QR кодів. Функція автоматично підключення клієнта через QR код який містить інформацію про назву мережу, тип шифрування і пароль у відкритому вигляді існує і легко імплементується до проекту.

Доступ до додатку потрібно надавати лише після авторизації за біометричними даними користувача. При наявності спеціальних сканерів відбитку пальці або сканера обличчя ця функція повинна бути обов’язкова і невимикаєма. Якщо такі сканери не доступні для використання з причин

відсутності або програмної зміни програмного забезпечення функція авторизації має працювати з використанням слова-паролю. Функція примусової ротації паролів авторизації є обов’язковою.

Одним із завершальних кроків наукової роботи є проведення тестування на діапазоні найбільш розповсюджених приладів. Так як порти за допомогою яких проводиться переналаштування девайсу є стандартними для більшості систем, то створене програмне забезпечення можна використовувати для налаштування будь-якого приладу з IoT. Тому необхідно провести тестування на пристроях кардинально відмінних від роутерів , наприклад таких як чайник або обігрівач з доступом до мережі інтернет.

Отже для створення додатку потрібно реалізувати такі модулі:

1. Модуль авторизації користувача за біометричними даними.
2. Модуль підключення до пристрою IoT
3. Модуль генерації та зміни авторизаційних даних IoT
4. Модуль шифрування та збереження нових даних
5. Модуль швидкого підключення нових користувачів через QR код або WPS

2. ОПИС ТЕХНОЛОГІЇ

2.1. Основні поняття

QR код- це 2-мірний тип штрих-коду і функціонує так само, як штрих-коди, які ми бачимо в продуктових магазинах і супермаркетах. Проте, різниця в тому, що коди QR можуть зберігати різні види інформації та набагато більше даних. Штрих-код на відміну від QR коду обмежується тільки наданням вам чисельного значення продукту.

QR коди мають два різних типи: статичні і динамічні.

Статичні коди не дозволяють відсівати або оновлювати вміст даних, після його генерації. Інформація жорстко закодована і залишається як є. Крім того, він не дозволяє відслідковувати дані сканування. Таким чином, він підходить тільки для бізнесу.

Тим не менш, ви можете створити свій статичний код безкоштовно. і термін дії статичного коду не обмежений.

Динамічний код є розширеним типом коду, який функціонує декількома способами. Це дозволяє змінювати дані і змінювати адресу призначення вашого коду, на який відбувається перенаправлення. Крім того, він дозволяє відстежувати дані, кількість сканувань і показує панель моніторингу даних про:

- Загальна сканування
- Унікальні відвідувачі
- Середнє сканування в день
- Географічне розташування сканерів
- Тип пристрою (iPhone / Android)

Як зазначається в роботі The Secure Shell (SSH)Transport Layer Protocol «SSH - це протокол прикладного рівня. SSH-сервер зазвичай прослуховує з'єднання на TCP-порту 22. Специфікація протоколу SSH-2 міститься в RFC

4251. Для аутентифікації сервера в SSH використовується протокол аутентифікації сторін на основі алгоритмів електронно-цифрового підпису RSA або DSA, але допускається також автентифікація за допомогою пароля (режим зворотної сумісності з Telnet) і навіть ip-адреси хоста. Для створення загального секрету (сеансового ключа) використовується алгоритм Діффі - Хеллмана (DH). Для шифрування переданих даних використовується симетричне шифрування, алгоритми AES, Blowfish або 3DES.»[12]

Протокол надає кілька альтернативних варіантів надійної аутентифікації, і захищає безпеку і цілісність зв'язку з сильним шифруванням. Це безпечна альтернатива незахищеним протоколам входу (таким як telnet, rlogin). Протокол працює в моделі клієнт-сервер, що означає, що з'єднання встановлюється клієнтом SSH, підключеним до сервера SSH. Клієнт SSH управляє процесом налаштування з'єднання і використовує криптографію відкритих ключів для перевірки особистості сервера SSH. Після фази налаштування протокол SSH використовує сильні симетричні алгоритми шифрування і хешування для забезпечення конфіденційності і цілісності даних, якими обмінюється клієнт і сервер.

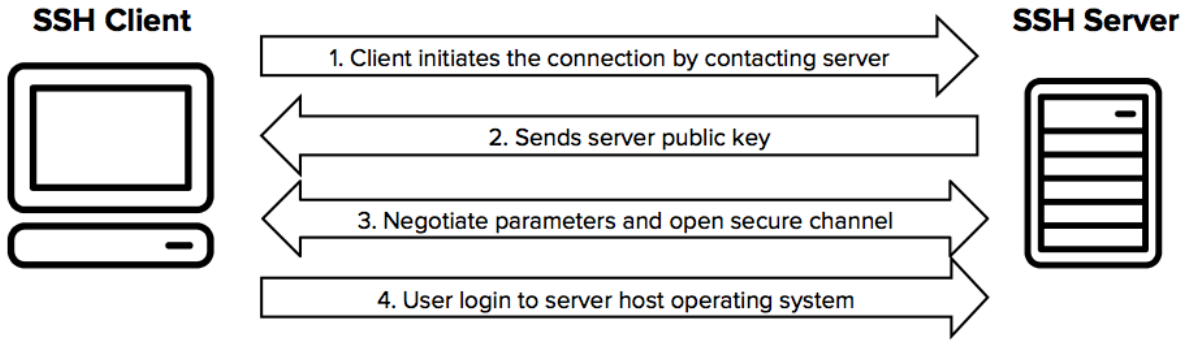

Рисунок 2.1 – Схема роботи протокол ssh [12]

SHA-256 - це одностороння функція, яка перетворює текст будь-якої довжини в рядок 256 біт. Це відомо як функція хешування. У цьому випадку це криптографічно захищена функція хешування, оскільки знання результату дуже мало говорить про вхід. Це модифікована версія SHA1, яка в свою чергу є модифікованою SHA0. Фактично алгоритм за допомогою

математичних формул перетворює вхідні дані в набір символів не зв’язаних між собою будь-яким змістом. Алгоритм є всесвітньо відомим і признаним та сама деструктирізація алгоритму і його роз’яснення виходить за рамки даної роботи.

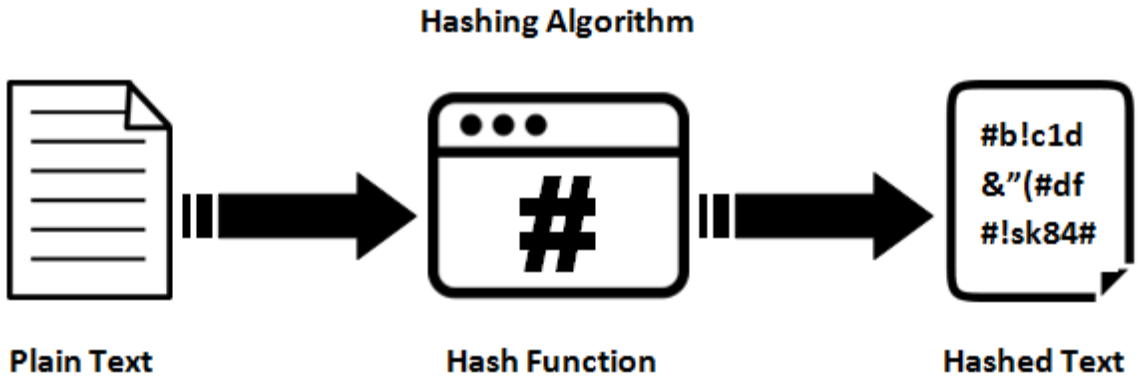

Рисунок 2.2 – Схема хешування

2.2. Абстрактна модель

Терміналом для керування девайсами та одночасно менеджером і генератором паролів виступає смартфон на базі операційної системи Android з встановленим спеціальним додатком. Внутрішня пам'ять смартфону виконує роль локальної бази даних. За потреби паролі можуть бути збережені в особистій хмарі. Екран смартфону виконує роль інтерфейсу відображення QR коду для швидкого підключення інших девайсів в локальну мережу.

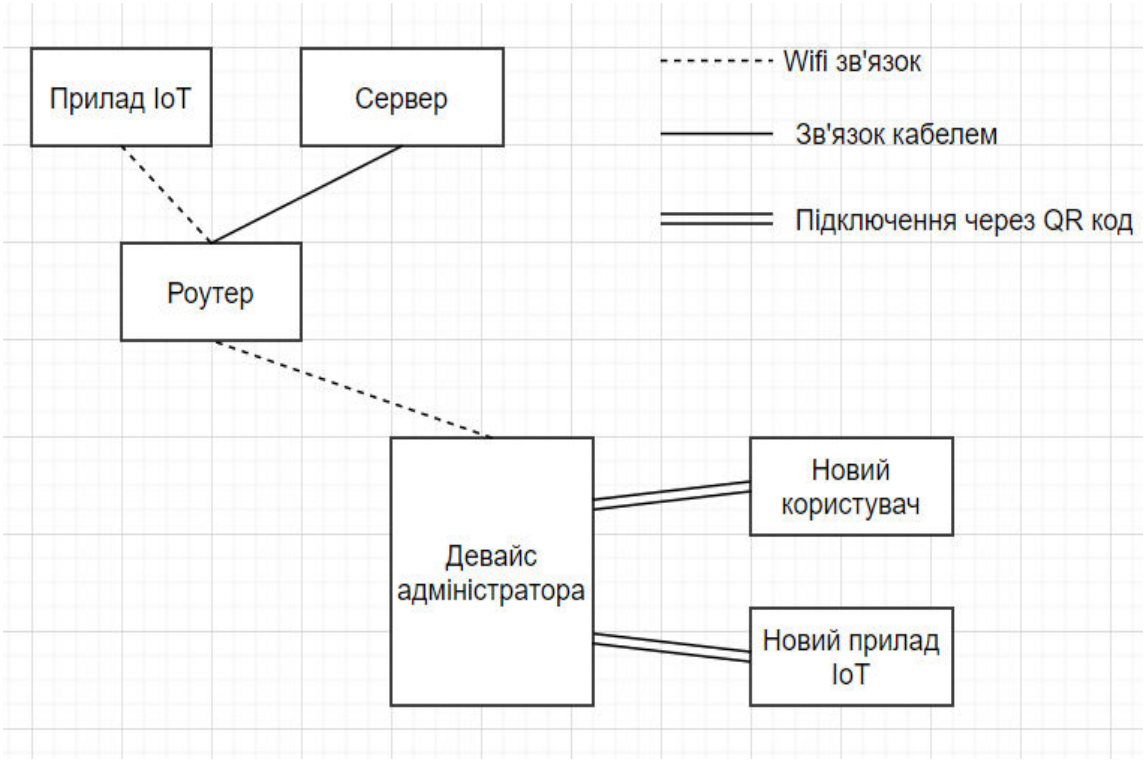

Рисунок 2.3 – Модель взаємодії пристроїв

Модель взаємодії пристроїв показує схему за якою працюватиме додаток та прилади IoT в одній локальній мережі. Кожен модуль додатку повинен виконувати свою роль, що дає змогу легкого майбутнього модифікування програмного забезпечення.

Модуль авторизації користувача виконує функцію перевірки відповідності занесених біометричних даних користувача-адміністратора та особи що намагається отримати доступ. Автоматично біометричні дані використані для авторизації зберігаються і в подальшому використовуються як сіль для хеш функції.

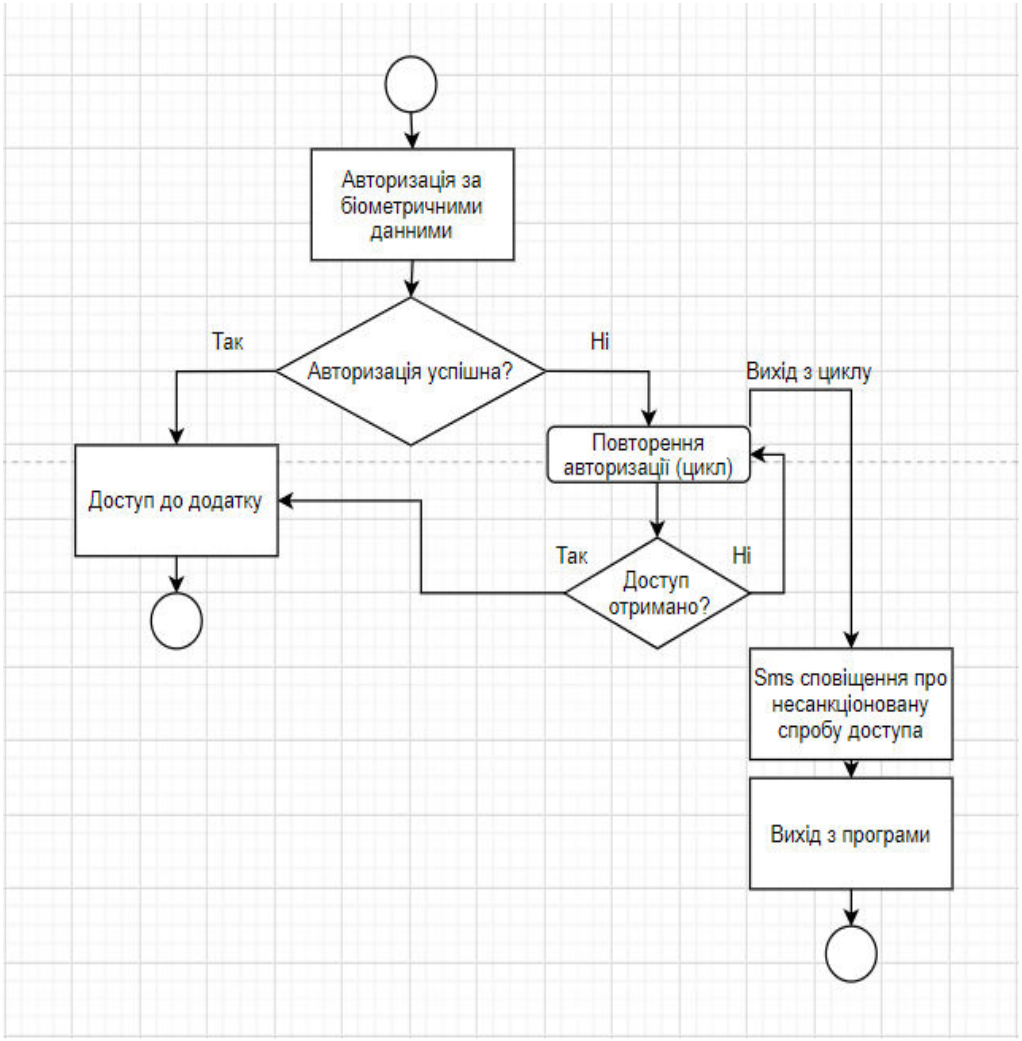

Рисунок 2.4 – Блок-схема роботи модулю авторизації

Модуль встановлення зв’язку відповідає за встановлення ssh з’єднання з пристроем IoT (роутером)

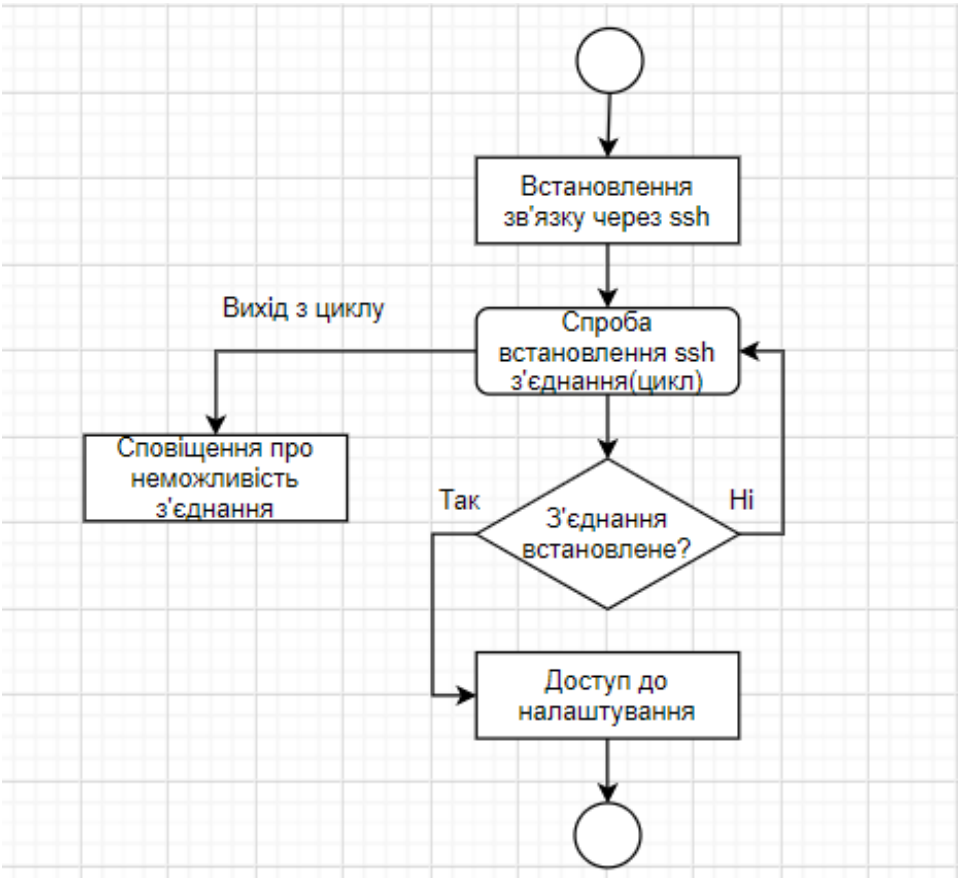

Рисунок 2.5 – Блок-схема роботи модуля зв’язку

Модуль ротації даних відповідає за зміну паролів через заданий проміжок часу або за потребою

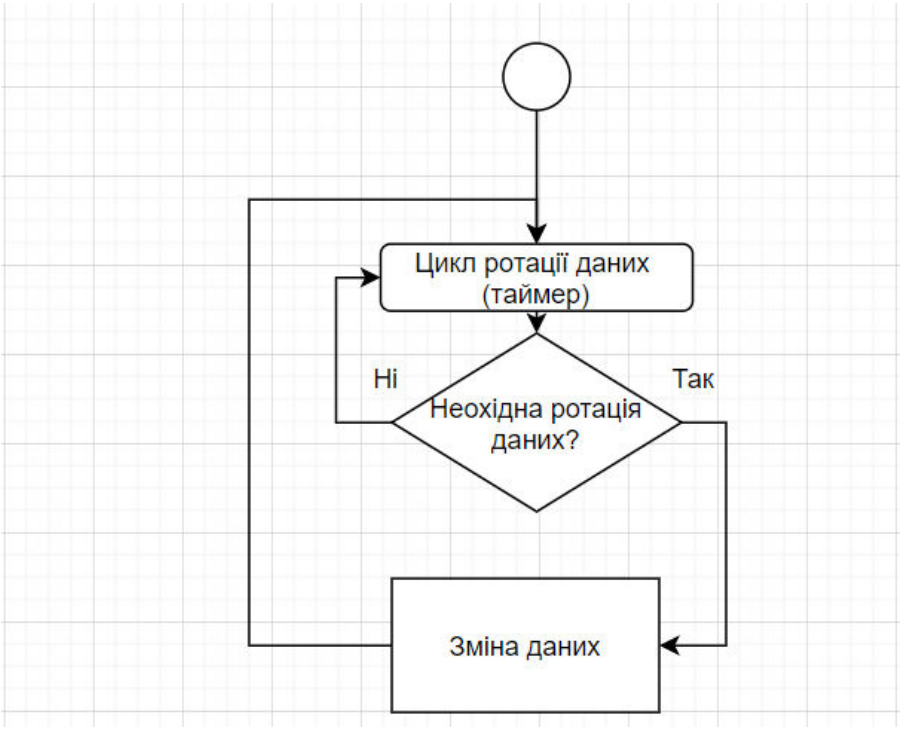

Рисунок 2.6 – Блок-схема роботи модуля ротації даних

Модулі збереження та генерації QR коду можуть бути об’єднані в одну блок схему та відповідають за шифрування , збереження та відображення нових авторизаційних даних.

Рисунок 2.7 – Блок-схема роботи модулів збереження даних та генерації QR коду

Основний інтерфейс користувача складатимуть декілька головних пунктів маніпуляції таких як:

- Кнопка генерації пароля
- Кнопка налаштування додатку
- Дисплей відображення таймеру
- Дисплей відображення QR коду

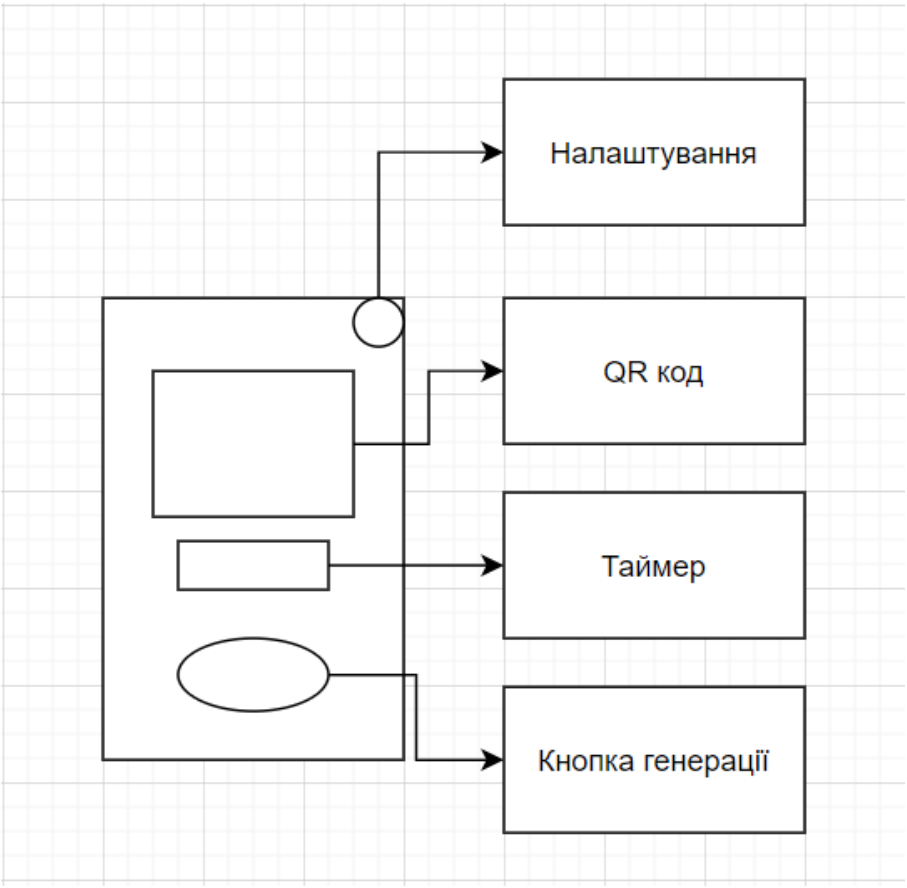

Рисунок 2.8 – Схема UI користувача

Передача інформації про налаштування нового паролю відбуватиметься через захищені протоколи. Повна схема роботи системи заключатиметься в генерації складного паролю в додатку на смартфоні. Збереженні його в локальному сховищі або в зашифрованому вигляді в «хмарі». Відображенні у вигляді QR коду на екрані девайсу, що в свою чергу унеможливлює його взлом методом запам’ятовування третіми особами. Далі передача налаштування на прилад інтернету речей. За необхідності ротація

паролю через певний проміжок часу з повторенням вище описаних етапів. Останній крок це переналаштування терміналу на використання нового паролю.

Отже в такий спосіб генерації, збереження та налаштування паролю для приладів інтернету речей гарантує помітне посилення безпеки систем та мереж взаємодії користувачів та елементів IoT. Що в свою чергу мінімізує шанси дестабілізації роботи всіх компонентів та збільшує надійність збереження особистих даних користувачів.

3. РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПІДВИЩЕННЯ РІВНЯ БЕЗПЕКИ ІОТ

Відповідно до загальних вимог, дослідження можна розділити на наступні підзадачі:

1. Автентифікація
2. Встановлення зв’язку с приладом IoT та зміна його налаштувань
3. Збереження, шифрування та відображення нових даних налаштувань
4. Тестування

3.1. Автентифікація

Розпізнавання відбитків пальців є активною науковою сферою на сьогодні. Важливим компонентом у системах розпізнавання відбитків пальців є алгоритм відповідності відбитків пальців. Відповідно до проблемної області алгоритми відповідності відбитків пальців класифікуються на дві категорії: алгоритми перевірки відбитків пальців та алгоритми ідентифікації відбитків пальців. Метою алгоритмів перевірки відбитків пальців є визначення того, походять два відбитки пальців від одного і того ж пальця чи ні. З іншого боку, алгоритми ідентифікації відбитків пальців шукають відбиток пальця запиту в базі даних, шукаючи відбитки пальців, що надходять з того ж пальця.

Розпізнавання обличчя - це метод виявлення або перевірки особи, використовуючи їх обличчя. Системи розпізнавання обличь можна використовувати для ідентифікації людей на фотографіях, відео або в режимі реального часу. Але дані розпізнавання обличчя можуть бути схильними до помилок. Програмне забезпечення для розпізнавання обличчя особливо погано розпізнає афроамериканців та інших етнічних меншин, жінок та молодих людей, часто неправильно ідентифікуючи або зовсім не може виконати ідентифікацію.

Існують модулі з відкритим програмним забезпеченням яка надають API для використання і інтеграції їх у власні додатки, що в свою чергу скорочує час затрачений на розробку і тестування модулю авторизації. Тому для створення додатку використано модуль біометричної авторизації розроблений групою програмістів Miguel Angel Medina Pérez, Milton García Borroto, Andres Eduardo Gutierrez Rodriguez, Octavio Loyola-González.

Процес авторизації вимагає від користувач надати біометричні дані методами доступними на використовуваному приладі. Перше зчитування даних відповідає за налаштування адміністратора, що спрощує початкове налаштування. Отриманий знімок відбитка від ОС девайсу зберігається для використання як сіль у процесі хешування паролів. Валідність відбитку перевіряється методом пошуку схожих відбитків заданих в ОС.

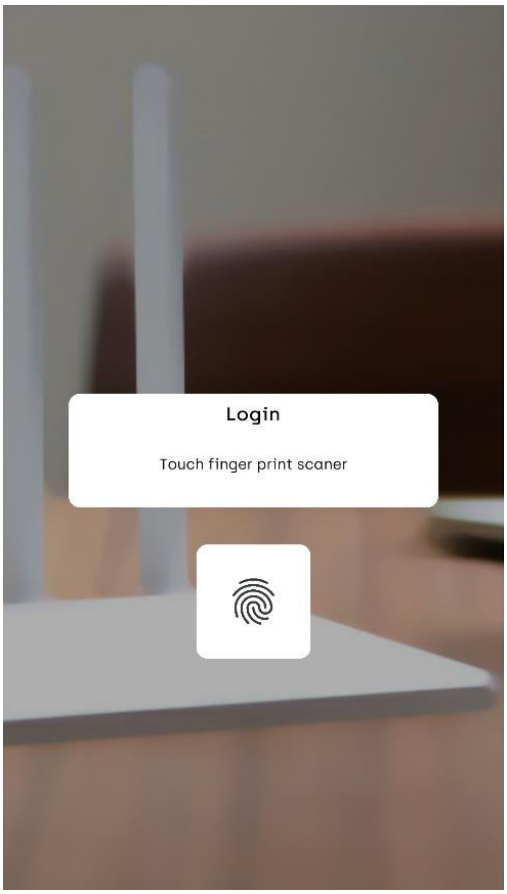

Рисунок 3.1 – Інтерфейс авторизації

Після авторизації користувачу доступний інтерфейс налаштування паролю приладу IoT за який відповідає модуль встановлення з’єднання. Необхідно задати дані адміністратора для першого входу на контрольований прилад IoT (роутер). Дані потрібно ввести лише перший раз після чого вони зберігаються у вигляді хешу. У будь-який час можна переналаштувати їх.

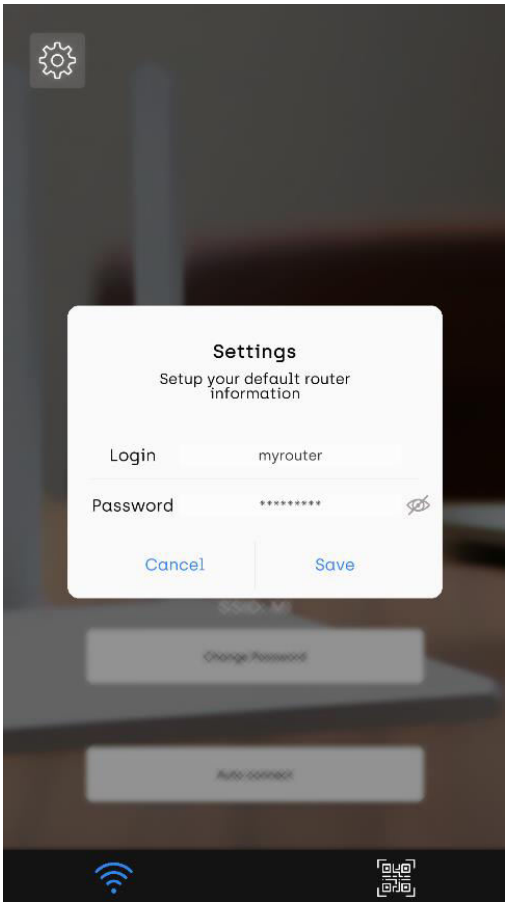

Рисунок 3.2 – Інтерфейс налаштування даних адміністратора

3.2. Встановлення зв’язку с приладом IoT та зміна його налаштувань

Після налаштування користувачу доступний головний екран програми з його функціоналом. Дві кнопки перша з яких відповідає за автоматичне під’єднання , автоматичну ротацію паролю, та його збереження. Інша потрібна для ручного під’єднання до роутера з новими налаштуваннями. Це необхідна функція, адже час за який роутер змінює свій пароль не є

універсальним тому якщо користувач впевнений, що роутер вже виконав перезавантаження він може під’єднатися вручну не очікуючи автоматичного з’єднання.

Функція під’єднная до Wi-Fi мережі потребує від користувача надати програми такі дозволи як:

- CHANGE_WIFI_STATE- Потрібно для увімкнення та відключення Wi-Fi на пристрої. Додавання та видалення конфігурацій до пристрою.
- ACCESS_WIFI_STATE - Потрібно для отримання доступу до поточного стану Wi-Fi або стану пристрою.
- ACCESS_COARSE_LOCATION- Потрібно для необхідно для сканування довколишніх Wi-Fi мереж.

Операційна система немає спеціального API для доступу до налаштувань Wi-Fi мережі сторонніми програми. Тому необхідно використовувати обхідні функції доступу. Код побудований на базі сторонньої пакету з назвою Android Wifi Manager.

Для налаштувань потрібно виконати таку послідовність дій як «підключення, налаштування, очищення, підключення»

Етап очищення дуже важливий адже система буде автоматично намагатися встановити старе підключення за старими правилами, тим самим не дозволяючи автоматичне підключення після ротації паролю, що робіть додаток нефункціональним.

```
    public static bool DisableNetwork(int networkId){
try
{
  using (var activity = GetActivity())
  using (var wifiManager = GetWiFiManager(activity))
  {
    return wifiManager.Call<bool>("disableNetwork", networkId);
  }
}
catch (Exception e)
{
  Debug.LogException(e);
}

return false;
```

}

Як видно на прикладі функції повного відключення від мережі доступ до API управління виконується через отримання «Activity», що є нетиповим способом управління ОС Android з використанням мови C#.

В програму необхідно занести стандартні дані адміністратора для роботи від його облікового запису. Встановлення ssh зв’язку виконується з використання вище наданих даних. З’єднання виконується за технологією клієнт-сервер. Роутер виступає в ролі сервера, а смартфон це клієнт. У програмному забезпеченні приладу IoT необхідна функція налаштування через ssh тобто пор 23 повинен бути відкритий.

Надалі програма автоматично виконує команду «ssh <username>@<ip> -p <password>» тим самим отримуючи доступ до налаштувань роутера, звичайно якщо дані валідні.

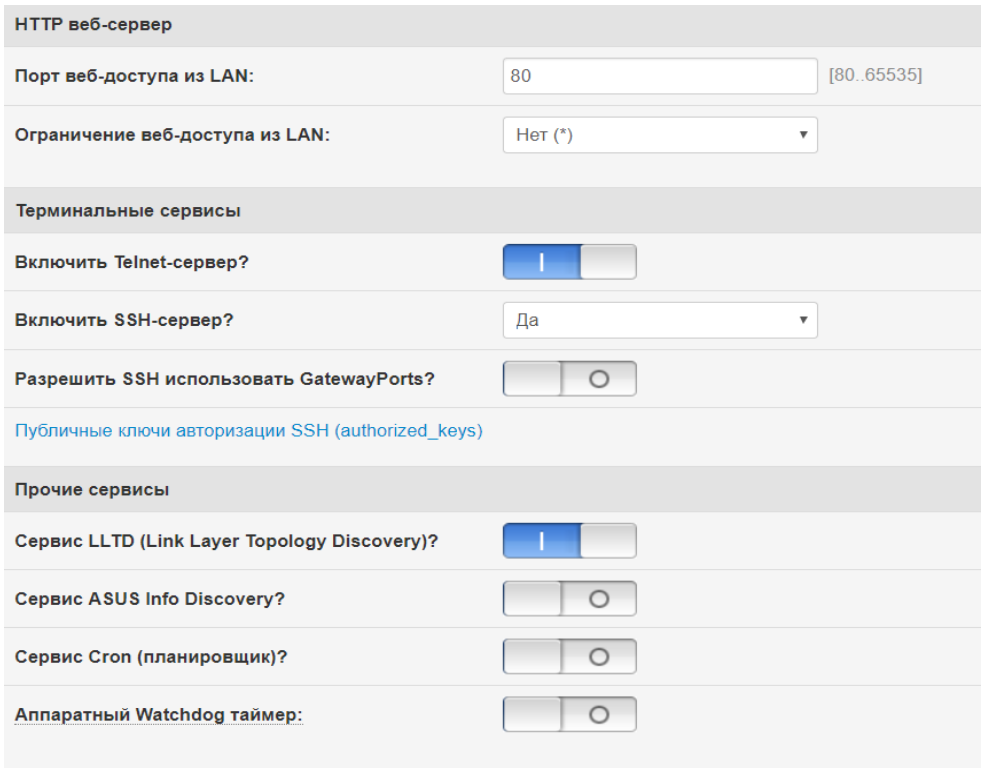

Рисунок 3.3 – Приклад увімнкеного ssh

Після створення ssh зв’язку програма автоматично звертається до NVRAM приладу та вносить зміни. NVRAM це скорочення для енергонезалежної пам'яті з випадковим доступом, NVRAM - це пам'ять, яка

зберігає дані, незалежно від того, живлення включено чи вимкнено. Сьогодні хорошим прикладом NVRAM є флеш-пам’ять, як та, що використовується в Jump-накопичувачі. NVRAM також знаходиться у вашому моніторі комп'ютера, принтерах, автомобілях, смарт-картах та інших пристроях, які потребують запам’ятованих налаштувань.

Для ручного пошуку паролю в NVRAM виконується команда nvram show | grep <password>. Додаток робить автоматичну зміну паролю командою nvram set <variable-name>="value".Змінна <variable-name>=" плаформозалежна. Для роутера марки ASUS вона дорівнює rt_wpa_psk. Після зміни даних виконується команда nvram commit для збереження внесених змін. Останнім кроком необхідно перезавантажити роутер командою reboot. Після перезавантаження роутера відбувається автоматичне підключення до точки доступу з використанням нових данних.

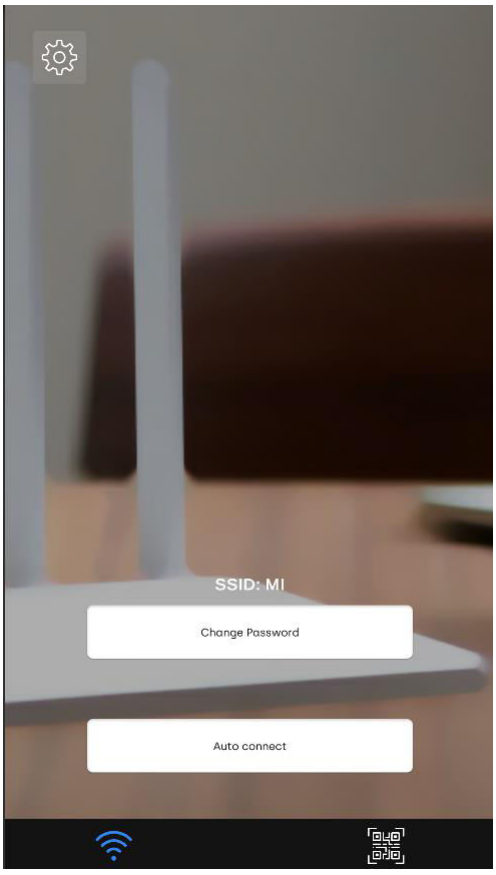

Рисунок 3.4 – Інтерфейс користувача

Виконання команд на роутері потребує часу. Деякі з них виконуються декілька секунд. Код внесення команд до NVRAM в якому явно вказано затримку на 5 секунд після запиту на виконання всіх операцій після встановлення з’єднання.

```
        Connector connector=new Connector(_host,_username,_password);
gameObject.DoLateAction(() =>
{
   connector.ExecuteCommand(ChangePasswordCommand());
   Debug.Log("Connected");

   connector.ExecuteCommand(CommitCommand());
   Debug.Log("Commited");

   connector.ExecuteCommand(RebootCommand());
   Debug.Log("Rebooted");

   UIWindows.instance.Open(WindowType.Alert);

},5);
```

Ssh приймає команди в текстовому форматі тому структура побудови команд це проста конкатенація певних символів та системозалежних змінних.

```
        public string ExecuteCommand(string commandString)
        {
   SshCommand command= SshClient.RunCommand(commandString);
   string answer = command.Result;
   return answer;
}
```

3.3. Збереження, шифрування та відображення нових даних налаштувань

Одночасно з тим на головному екрані доступні кнопки навігації на екран з QR кодом автоматичного підключення. QR код необхідний для швидкого підключення приладів через мережу Wi-Fi. Якщо певні пристрої втратили зв'язок то технологія виконання команд через код наявна в більшості пристроїв. Його відображення за замовчуванням приховане, що також є вимогою безпеки. Структура команд, що кодуються максимально

спрощена, але тим самим її розуміння без спеціальних знань може бути ускладнене.

```
public const string template = "WIFI:T:WPA;S:mynetwork;P:mypass;;";
```

Синтаксис повинен в точності відповідати прикладу наведеному вище. Де параметр «T» це тип шифрування такий як WEP або WPA. Параметр «S» це SSID мережі, тобто її ім'я. Параметр «P» повинен містити пароль. Існує ще один параметр «H» який додається якщо Wi-Fi мережа є прихованою. Зазвичай ця модифікація рідка аде програма її підтримує.

Саме зображення QR коду це текстура 256x256 пікселів генерація якої виконується послідовним створенням окремих пікселів за заданими правилами.

```
        private static Color32[] Encode(string textForEncoding, int width, int height){
  var writer = new BarcodeWriter {
     Format = BarcodeFormat.QR_CODE,
     Options = new QrCodeEncodingOptions {
        Height = height,
        Width = width
     }
  };
  return writer.Write(textForEncoding);
        }
```

Мови програмування такі як C# та Java мають спеціальні бібліотеки які можуть значно полегшити процес написання створення та обробки QR кодів.

Якщо прилад не має можливості під'єднання через QR код то це можна зробити увімкнувши WPS на роутері за допомогою додатку. Тим самим налаштувавши всю мережу в стан який був до ротації паролю.

Рисунок 3.5 – Інтерфейс відображення QR коду

Генерація нового паролю відбувається за заданими в програмі правилами надійності. Пароль у відкритому вигляді не зберігається, а лише один раз показується користувачу, ця формальність необхідна для того щоб користувач впевнився в тому, що вигляд пароль має інший. Надалі пароль хешується. Абстрактна хеш функція має вигляд «дані + сіль=хеш». Сіль в даному випадку це оброблена інформація з відбитку пальця. Це дозволяє відновити втрачений доступ за допомогою окремої програми яка має спеціальні налаштування з використанням біометричної інформації користувача, тим самим підбір паролю виконується швидше. Так як використовується хеш функція то іншого способу відновлення даних лише як підбір не існує.

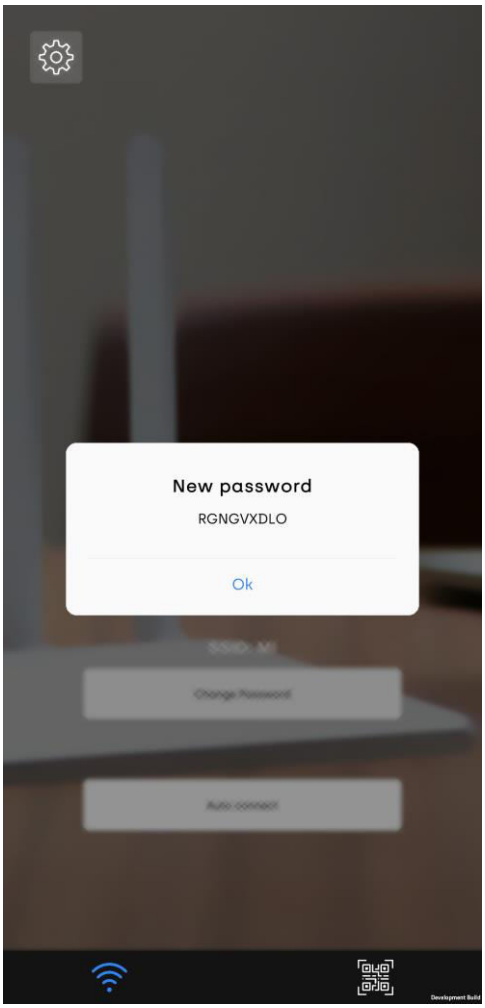

Рисунок 3.6 – Показ нового паролю

Новий пароль зберігається у місці вказаному користувачем (локально або в хмарі за наявності доступу до інтернету).

Після цього QR код перегенеровується та виставляться таймер на сповіщення користувача про наступну заміну пароля.

Ротація паролів відбувається відповідно до налаштувань вбудованого таймера. Складність паролів виставляється в налаштуваннях програми. За замовчуванням пароль змінюється кожну годину що в свою чергу унеможливлює його дешифрування із пійманого handshake роутера та іншого приладу IoT.

3.4. Тестування

Тестування проводилось з використанням смартфона Samsung A30 (Android 10) та роутера TP-Link WR841N (прошивка)

Після входу в панель адміністратора на роутері стає доступним для перегляду пароль для бездротового підключення.

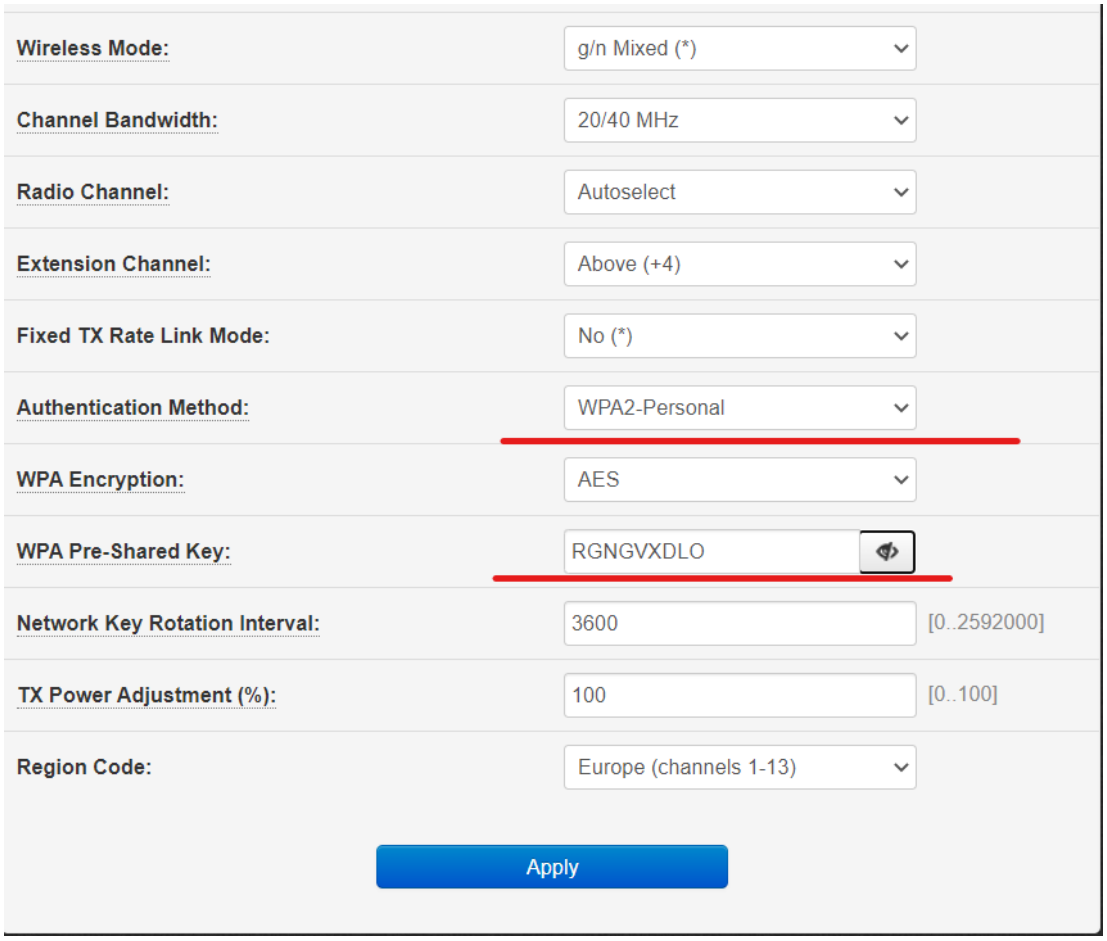

Рисунок 3.7 – Пароль бездротового підключення до авторотації паролю

Виконавши авторизацію в мобільний додаток та задавши аутентифікаційні данні адміністратора виконується команда авторотації пароля.

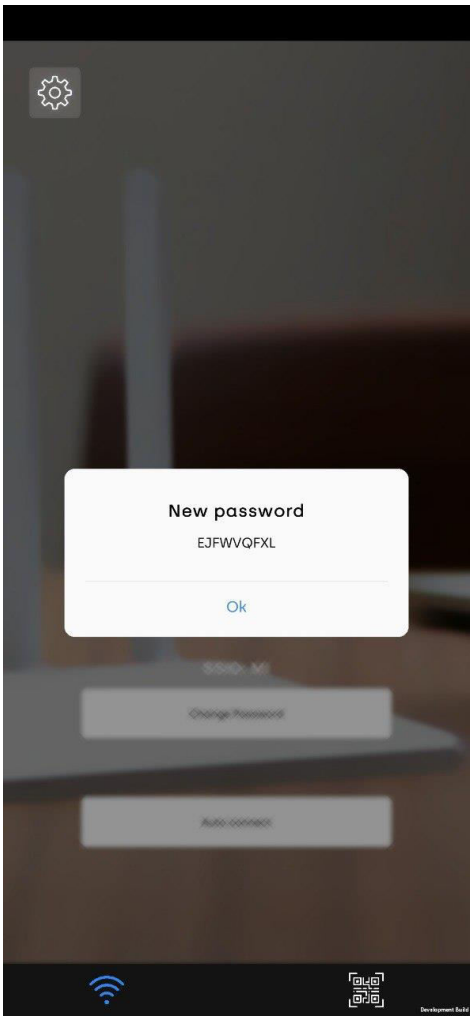

Рисунок 3.8 – Результат авторотації паролю

Відповідно до отриманого демонстраційного екрану з новим паролем хеш паролю з додаванням опрацьованих біометричних даних користувача збережені на пристрої та мають вид

5cbb6eb0060ca85f93be47117b0ffe822a4f1fd572d6e9b5683ab7493c005ede

Такий набір символі гарантує підвищений рівень безпеки, а також його відносно безпечно можна зберігати і на віддаленому сховищі.

Останнім кроком тестування є перевірка чи справді пароль на роутері змінився і відповідає заданому вище. Для цього повторюємо операцію підключення до роутера, але вже з використанням нового паролю.

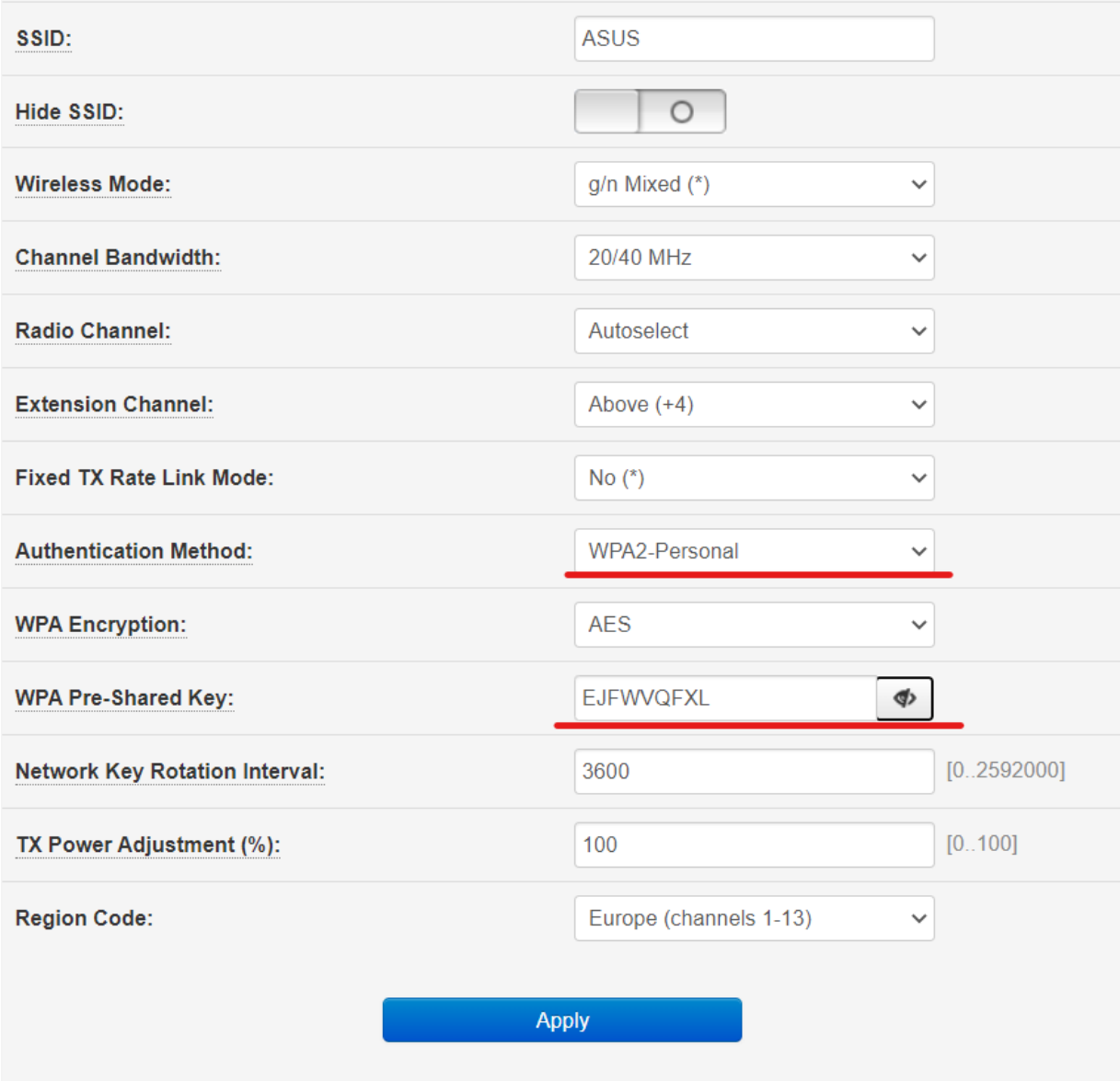

Рисунок 3.9 – Новий пароль встановлений на роутері

Наявність підключення до роутера або те що телефон автоматично виконав перепідключення з використанням нового паролю вже говорить про те, що роутер змінив старий пароль. Але для остаточної впевненості потрібно перевірити поле «WPA Pre-Shared Key» в панелі адміністратора, де тепер показано новій пароль, що повністю відповідає автоматично заданому. Роботоздатість всієї мережі не порушено.

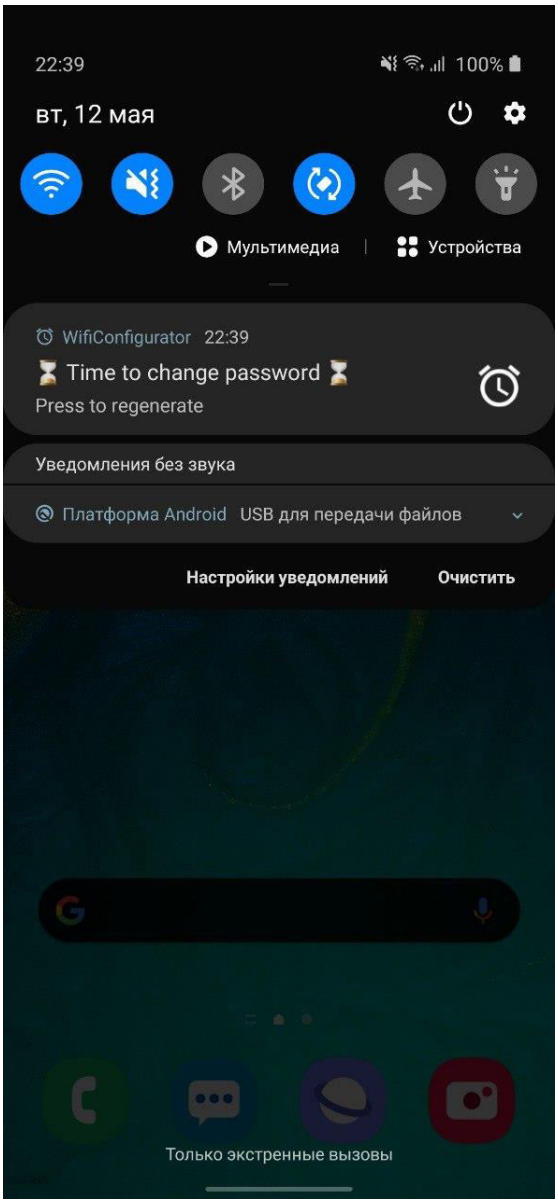

Рисунок 3.10 – Пуш сповіщення про замін пароля

Через годину користувач отримує сповіщення про зміну паролю.

ВИСНОВОК

В ході виконання магістерської наукової роботи було створено мобільний додаток який автоматично добавляє паролю від приладу IoT термін придатності. Що не дає можливості підбору адже за час дешифрування паролю він вже зміниться.

Використання технології динамічного паролю значно зменшує ризики бути атакованим. Симбіоз систем менеджементу паролів, їх генерації та автоматичного переналаштування є найкращим та найзручнішим рішенням для повсякденного користування. Кросплатформність майбутньої системи дозволяє використовувати її для захисту будь-якої техніки інтернету речей.

Серед недоліків додатку можна виділити неможливість роботи з девайсами певних виробників таких як Tp-Link без модифікації. Адже користувач заблокував доступ до термінальної конфігурації свого продукту.

Створене програмне забезпечення має відкритий код, що дозволить модифікувати його та покращувати будь-кому.

Подальші дослідження полягають в тому, щоб поліпшити інтерфейс користувача, додати більше моделей сумісного обладнання, вбудувати підтримку нових алгоритмів шифрування. Необхідно зробити більш широкий вибір налаштувань параметрів генерації паролів, надати користувачу здатність задати правила створення паролів. Додати можливість ротації не тільки вхідних даних а і даних адміністратора на роутері,що в свою чергу додатково збільшить рівень безпеки.

АПРОБАЦІЯ

Інформаційна технологія підвищення рівня безпеки приладів інтернету речей / Д. В. Великодний, С. В. Токмань // МІЖНАРОДНА НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ студентів та молодих вчених (Суми, 20–24 квітня 2020 року)

СПИСОК ЛІТЕРАТУРИ

1. Gupta, A. *IoT Hackers Handbook*; AttifyInc: Sunnyvale, CA, USA, 2017.

2. Feingold, J. Dyn issues analysis of cyberattacks. New Hampshire Business Re-View. 2016.

3. Nastase, L. Security in the Internet of Things: A Survey on Application Layer Protocols. In Proceedings of the 2017 21st International Conference on Control Systems and Computer Science, Bucharest, Romania, 29–31 May 2017.

4. Katsikeas, S.; Fysarakis, K.; Miaoudakis, A.; Bemten, A.V.; Askoxylakis, I.; Papaefsta-thiou, I.; Plemenos, A. Lightweight & Secure Industrial IoT Communications via the MQ Telemetry Transport Protocol. In Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, 3–6 July 2017.

5. Perrazzone, J.B.; Yu, P.L.; Sadler, B.M.; Blum, R.S. Cryptographic Side-Channel Signaling and Authentication via Fingerprint Embedding. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 2216–2225

6. Fischlin, M.; Janson, C.; Mazaheri, S. Backdoored Hash Functions: Immunizing HMAC and HKDF. In Proceedings of the 2018 IEEE 31st Computer Security Foundations Symposium, Oxford, UK, 9–12 July 2018.

7. Hao, Y. *The Boomeraang Attacks on BLAKE and BLAKE2*; Springer: Cham, The Netherlands, 2015.

8. Bogdanov, A.; Khovratovich, D.; Rechberger, C. Biclique Cryptanalysis of the Full AES. In *International Association for Cryptologic Research 2011*; ASIACRYPT 2011, LNCS 7073; Springer: Berlin/Heidelberg, Germany, 2011; pp. 344–371.

9. Siddavaatam, P.; Sedaghat, R.; Cheng, M.H. An Adaptive Security Framework with Extensible Computational Complexity for Cipher Systems. In Proceedings of the 11th International Conference for Internet Technology and Secured Transactions, Barcelona, Spain, 5–7 December 2016.

10. B. Herzberg, D. Bekerman and I. Zeifman, Breaking Down Mirai: An IoT DDoS Botnet Analysis, Imperva Incapsula, Oct. 2016, [online] Available: www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html.

11. Postel, J. and J. Reynolds, "Telnet Protocol Specification", STD 8, RFC 854, May 1983.

12. Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH)Transport Layer Protocol", RFC 4253, January 2006.

13. В.А. Ворона, В.А. Тихонов «Системы контроля и управления доступом»

ДОДАТОК

Додаток А

QRCodeGen.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using ZXing;
using ZXing.QrCode;

public class QrCodeGen : MonoBehaviour
{
   public const string template = "WIFI:T:WPA;S:mynetwork;P:mypass;;";
   public static Texture2D GenerateQr(string ssid, string pass)
   {
      return generateQR($"WIFI:T:WPA;S:{ssid};P:{pass};;");
   }

   private static Color32[] Encode(string textForEncoding, int width, int height) {
      var writer = new BarcodeWriter {
         Format = BarcodeFormat.QR_CODE,
         Options = new QrCodeEncodingOptions {
            Height = height,
            Width = width
         }
      };
      return writer.Write(textForEncoding);
   }

   private static Texture2D generateQR(string text) {
      var encoded = new Texture2D (256, 256);
      var color32 = Encode(text, encoded.width, encoded.height);
```

```
        encoded.SetPixels32(color32);
        encoded.Apply();
        return encoded;
    }
}
```

StringGenrator.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using UnityEngine;
using Random = UnityEngine.Random;
//simple version for showing to all
public class StringGenerator : MonoBehaviour
{
    public static string GenerateString(int length)
    {
        // creating a StringBuilder object()
        StringBuilder str_build = new StringBuilder();
        System.Random random = new System.Random();

        char letter;

        for (int i = 0; i < length; i++)
        {
            double flt = random.NextDouble();
            int shift = Convert.ToInt32(Math.Floor(25 * flt));
            letter = Convert.ToChar(shift + 65);
            str_build.Append(letter);
        }
```

```
        return str_build.ToString();
    }
}
```

Monosingleton.cs

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

/// <summary>
/// Mono singleton Class. Extend this class to make singleton component.
/// Example:
/// <code>
/// public class Foo : MonoSingleton<Foo>
/// </code>. To get the instance of Foo class, use <code>Foo.instance</code>
/// Override <code>Init()</code> method instead of using <code>Awake()</code>
/// from this class.
/// </summary>
public abstract class MonoSingleton<T> : MonoBehaviour where T : MonoSingleton<T>
{
    private static T m_Instance = null;
    public static T instance
    {
        get
        {
            // Instance requiered for the first time, we look for it
            if( m_Instance == null )
            {
                m_Instance = GameObject.FindObjectOfType(typeof(T)) as T;

                // Object not found, we create a temporary one
                if( m_Instance == null )
```

```
            {
                                    Debug.LogWarning("No instance of " + typeof(T).ToString() + ", a
temporary one is created.");

                                        isTemporaryInstance = true;
                m_Instance = new GameObject("Temp Instance of " + typeof(T).ToString(),
typeof(T)).GetComponent<T>();

                // Problem during the creation, this should not happen
                if( m_Instance == null )
                {
                   Debug.LogError("Problem during the creation of " + typeof(T).ToString());
                }
            }
                                    if (!_isInitialized){
                                        _isInitialized = true;
                                        m_Instance.Init();
                                    }
        }
        return m_Instance;
    }
}

        public static bool isTemporaryInstance { private set; get; }

        private static bool _isInitialized;

// If no other monobehaviour request the instance in an awake function
// executing before this one, no need to search the object.
private void Awake()
{
    if (m_Instance == null) {
                        m_Instance = this as T;
```

```
                } else if (m_Instance != this) {
                        Debug.LogError ("Another instance of " + GetType () + " is already exist!
Destroying self...");
                        DestroyImmediate (this);
                        return;
                }
                if (!_isInitialized) {
                        DontDestroyOnLoad(gameObject);
                        _isInitialized = true;
                        m_Instance.Init ();
                }
    }


    /// <summary>
    /// This function is called when the instance is used the first time
    /// Put all the initializations you need here, as you would do in Awake
    /// </summary>
        public virtual void Init(){}

    /// Make sure the instance isn't referenced anymore when the user quit, just in case.
    private void OnApplicationQuit()
    {
        m_Instance = null;
    }
}
```

Controller.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using EasyMobile;
```

```
using FSG.Android.Wifi;
using UnityEngine;
using UnityEngine.UI;
using ZXing;
using ZXing.QrCode;



// nvram show | grep <password>
//nvram set <variable-name>="value" // for asus rt_wpa_psk="";
//nvram commit
//reboot
public class Controller : MonoSingleton<Controller>
{
   [SerializeField] private Button connectButton;
   [SerializeField] private InputField _inputField;

   [SerializeField] private Notification Notification;
   private string _host = "192.168.1.1";
   private string _username = "admin";
   private string _password = "passwd";

   private string _ssid="ASUS";
   private string _pass = "dorfswifi";

   [SerializeField] private Button ConnectBtn;

   private void Awake()
   {
      //connectButton.onClick.AddListener(delegate { AndroidWifiManager.Connect(_ssid,_pass); });

      GetAdminData();
      connectButton.onClick.AddListener(ConnectSSH);
```

```
    ConnectBtn.onClick.AddListener(delegate { AndroidWifiManager.Connect(_ssid,
GetCurrentPassword()); });


  }


  private void OnDestroy()
  {
    //connectButton.onClick.RemoveListener(delegate { AndroidWifiManager.Connect(_ssid,_pass); });
    connectButton.onClick.RemoveListener(ConnectSSH);


    ConnectBtn.onClick.RemoveListener(delegate { AndroidWifiManager.Connect(_ssid,
GetCurrentPassword()); });
  }


  public void OnEnable()
  {
    EventManager.StartListening(Variables.E_UpdateAdminInfo,GetAdminData);


  }


  private void Start()
  {
    Notifications.Init();
    Notification.ScheduleLocalNotification(new TimeSpan(0,0,20));
  }


  public void OnDisable()
  {
    EventManager.StopListening(Variables.E_UpdateAdminInfo,GetAdminData);
  }


  public const string PASSWORD ="PASSWORD";
```

```
public void RegeneratePassword()
{

}

private void GetAdminData(object[] args=null)
{
   _username = Cloud.GetFromCloud(WindowSettings.ADMIN_LOGIN,String.Empty);
   _password = HashCode.Dehash(Cloud.GetFromCloud(WindowSettings.ADMIN_PASS,String.Empty));
}


private string ChangePasswordCommand()
{
   var pass = StringGenerator.GenerateString(9);
   PlayerPrefs.SetString(PASSWORD,pass);
   return $"/usr/sbin/nvram set rt_wpa_psk=\"{pass}\"";
}

private string CommitCommand()
{
   return "/usr/sbin/nvram commit";
}

private string RebootCommand()
{
   return "/sbin/reboot";
}

public string GetCurrentPassword()
{
```

```
    return HashCode.Dehash(Cloud.GetFromCloud(PASSWORD,String.Empty));
}

public string GetCurrentSsid()
{
    return _ssid;
}
public void ConnectSSH()
{


    Connector connector=new Connector(_host,_username,_password);
    gameObject.DoLateAction(() =>
    {
        connector.ExecuteCommand(ChangePasswordCommand());
        Debug.Log("Connected");

        connector.ExecuteCommand(CommitCommand());
        Debug.Log("Commited");

        connector.ExecuteCommand(RebootCommand());
        Debug.Log("Rebooted");

        UIWindows.instance.Open(WindowType.Alert);

    },5);



    gameObject.DoLateAction(() =>
    {
        AndroidWifiManager.Connect(_ssid, GetCurrentPassword());
```

```
            Debug.Log("Reconnected" + GetCurrentPassword());
        },35f);
        gameObject.DoLateAction(() =>
        {
            AndroidWifiManager.Connect(_ssid, GetCurrentPassword());
            Debug.Log("Reconnected");
        },60f);
        gameObject.DoLateAction(() =>
        {
            AndroidWifiManager.Connect(_ssid, GetCurrentPassword());
            Debug.Log("Reconnected");
        },80f);

    }

    public void Connect()
    {
        AndroidWifiManager.Connect(_ssid, GetCurrentPassword());
    }

    public void ForgotWifi(string ssid)
    {

            AndroidWifiManager.RemoveNetwork(AndroidWifiManager.GetConfiguredNetworks().Find(q =>
q.SSID == ssid).networkId);
            Debug.Log("Removed");

    }


}
```

AndroidWifiManager.cs

```
//#define EDITING

#if EDITING || (UNITY_ANDROID && !UNITY_EDITOR)
#define PLATFORM_SUPPORTED
#endif

using System.Collections.Generic;
using UnityEngine;
using System;

namespace FSG.Android.Wifi
{
   /// <summary>
   /// Handles interop between Unity and the WifiManager Android class
   /// </summary>
   public static class AndroidWifiManager
   {
      public enum ConnectResult
      {
         SUCCESS,
         ADD_NETWORK_FAILED,
         DISCONNECT_FAILED,
         ENABLE_NETWORK_FAILED,
         CONNECT_FAILED,
         UNSUPPORTED_PLATFORM,
      }
#if UNITY_EDITOR
      private static string s_debugConnectedNetwork = string.Empty;
      private static bool s_debugWifiEnabled = true;
#endif
```

```
        #region Private Methods
#if PLATFORM_SUPPORTED
        /// <summary>
        /// Returns the Unity applications activity
        /// </summary>
        private static AndroidJavaObject GetActivity()
        {
            try
            {
                return new
AndroidJavaClass("com.unity3d.player.UnityPlayer").GetStatic<AndroidJavaObject>("currentActivity");
            }
            catch (Exception ex)
            {
                Debug.LogException(ex);
                Debug.LogError("Error getting currentActivity, are you sure you're on Android?");
                return null;
            }
        }

        /// <summary>
        /// Returns the WifiManager object from the Unity activity
        /// </summary>
        private static AndroidJavaObject GetWiFiManager(AndroidJavaObject activity)
        {
            try
            {
                CheckPermissions();
                return activity.Call<AndroidJavaObject>("getSystemService", "wifi");
            }
            catch (Exception ex)
            {
```

```
            Debug.LogException(ex);
            Debug.LogError("Error getting wifi service, are you sure you're on Android?");
            return null;
        }
    }
#if UNITY_2019_2_OR_NEWER
    private static string[] s_requiredPermissions = new string[]
    {
        UnityEngine.Android.Permission.CoarseLocation,
        UnityEngine.Android.Permission.FineLocation
    };
    /// <summary>
    /// Ensure the required permissions are granted
    /// </summary>
    private static void CheckPermissions()
    {
        for (int i = 0; i < s_requiredPermissions.Length; i++)
        {
            if (!UnityEngine.Android.Permission.HasUserAuthorizedPermission(s_requiredPermissions[i]))
            {
                UnityEngine.Android.Permission.RequestUserPermission(s_requiredPermissions[i]);
            }
        }
    }
#else
    private static void CheckPermissions() { }
#endif

    /// <summary>
    /// Converts a C# string array to a java.lang.String array
    /// </summary>
    /// <param name="values">Array to be converted</param>
```

```
    /// <returns>The java array</returns>
    private static AndroidJavaObject StringArrayToJavaArray(string[] values)
    {
        AndroidJavaClass arrayClass = new AndroidJavaClass("java.lang.reflect.Array");
        AndroidJavaObject arrayObject = arrayClass.CallStatic<AndroidJavaObject>("newInstance", new
AndroidJavaClass("java.lang.String"), values.Length);
        for (int i = 0; i < values.Length; i++)
        {
            arrayClass.CallStatic("set", arrayObject, i, new AndroidJavaObject("java.lang.String", values[i]));
        }
        return arrayObject;
    }
#endif
    #endregion

    #region Public Methods
    /// <summary>
    /// Gets the built SDK version
    /// </summary>
    /// <returns>Version code of the current Android SDK</returns>
    public static int GetAndroidBuildVersion()
    {
        try
        {
            using (var version = new AndroidJavaClass("android.os.Build$VERSION"))
            {
                return version.GetStatic<int>("SDK_INT");
            }
        }
        catch (Exception ex)
        {
            Debug.LogException(ex);
```

```
        }
        return -1;
    }

    /// <summary>
    /// Enable or disable Wi-Fi.
    /// </summary>
    /// <param name="enabled">True to enable, false to disable.</param>
    /// <returns>False if the request cannot be satisfied; true indicates that wifi is either already in the requested state, or in progress toward the requested state.</returns>
    public static bool SetWifiEnabled(bool enabled)
    {
#if PLATFORM_SUPPORTED
        try
        {
            using (var activity = GetActivity())
            using (var wifiManager = GetWiFiManager(activity))
            {
                return wifiManager.Call<bool>("setWifiEnabled", enabled);
            }
        }
        catch (Exception e)
        {
            Debug.LogException(e);
        }
#elif UNITY_EDITOR
        s_debugWifiEnabled = enabled;
        if (!enabled)
        {
            s_debugConnectedNetwork = string.Empty;
        }
#endif
```

```
        return false;
    }

    /// <summary>
    /// Return whether Wi-Fi is enabled or disabled.
    /// </summary>
    public static bool IsWifiEnabled()
    {
#if PLATFORM_SUPPORTED
        try
        {
            using (var activity = GetActivity())
            using (var wifiManager = GetWiFiManager(activity))
            {
                return wifiManager.Call<bool>("isWifiEnabled");
            }
        }
        catch (Exception e)
        {
            Debug.LogException(e);
        }
        return false;
#elif UNITY_EDITOR
        return s_debugWifiEnabled;
#else
        return false;
#endif
    }

    /// <summary>
    /// Begins a scan to find available wifi networks. Limited to 4 times per 2 minutes while the app is
open.
```

```
        /// </summary>
        /// <returns>True if the operation succeeded, i.e., the scan was initiated.</returns>
        public static bool StartScan()
        {
#if PLATFORM_SUPPORTED
            try
            {
                using (var activity = GetActivity())
                using (var wifiManager = GetWiFiManager(activity))
                {
                    return wifiManager.Call<bool>("startScan");
                }
            }
            catch (Exception e)
            {
                Debug.LogException(e);
            }
#endif
            return false;
        }


        /// <summary>
        /// Return the results of the latest access point scan.
        /// </summary>
        /// <returns>The list of access points found in the most recent scan. An app must hold
ACCESS_COARSE_LOCATION or ACCESS_FINE_LOCATION permission in order to get valid
results.</returns>
        public static List<AndroidWifiScanResults> GetScanResults()
        {
            List<AndroidWifiScanResults> results = new List<AndroidWifiScanResults>();
#if PLATFORM_SUPPORTED
            try
            {
```

```
            using (var activity = GetActivity())
            using (var wifiManager = GetWiFiManager(activity))
            {
                using (var androidList = wifiManager.Call<AndroidJavaObject>("getScanResults"))
                {
                    int size = androidList.Call<int>("size");
                    for (int i = 0; i < size; i++)
                    {
                        var result =
AndroidWifiScanResults.FromAndroidObject(androidList.Call<AndroidJavaObject>("get", i));
                        results.Add(result);
                    }
                }
            }
        }
        catch (Exception e)
        {
            Debug.LogException(e);
        }
#elif UNITY_EDITOR
        if (s_debugWifiEnabled)
        {
            results.Add(new AndroidWifiScanResults()
            {
                SSID = "Editor Test Open",
                securityType = AndroidWifiSecurityType.OPEN,
                level = 100,
            });
            results.Add(new AndroidWifiScanResults()
            {
                SSID = "Editor Test WEP",
                securityType = AndroidWifiSecurityType.WEP,
```

```
            level = 100,
        });
        results.Add(new AndroidWifiScanResults()
        {
            SSID = "Editor Test WPA",
            securityType = AndroidWifiSecurityType.WPA,
            level = 100,
        });
    }
#endif
    return results;
}

/// <summary>
/// Return a list of all the networks configured for the current foreground user.
/// </summary>
/// <returns>A list of network configurations</returns>
public static List<AndroidWifiConfiguration> GetConfiguredNetworks()
{
    List<AndroidWifiConfiguration> results = new List<AndroidWifiConfiguration>();
#if PLATFORM_SUPPORTED
    try
    {
        using (var activity = GetActivity())
        using (var wifiManager = GetWiFiManager(activity))
        {
            using (var androidList = wifiManager.Call<AndroidJavaObject>("getConfiguredNetworks"))
            {
                int size = androidList.Call<int>("size");
                for (int i = 0; i < size; i++)
                {
```

```
                    var result =
AndroidWifiConfiguration.FromAndroidObject(androidList.Call<AndroidJavaObject>("get", i));
                    results.Add(result);
                }
            }
        }
        catch (Exception e)
        {
            Debug.LogException(e);
        }
#elif UNITY_EDITOR
        if (!string.IsNullOrEmpty(s_debugConnectedNetwork))
        {
            results.Add(new AndroidWifiConfiguration()
            {
                SSID = s_debugConnectedNetwork,
                status = AndroidWifiConfiguration.Status.CURRENT,
            });
        }
#endif
        return results;
    }

    /// <summary>
    /// Gets the Wi-Fi enabled state.
    /// </summary>
    public static AndroidWifiState GetWifiState()
    {
#if PLATFORM_SUPPORTED
        try
        {
```

```
            using (var activity = GetActivity())
            using (var wifiManager = GetWiFiManager(activity))
            {
                return (AndroidWifiState)wifiManager.Call<int>("getWifiState");
            }
        }
        catch (Exception e)
        {
            Debug.LogException(e);
        }
        return AndroidWifiState.UNKNOWN;
#elif UNITY_EDITOR
        return s_debugWifiEnabled ? AndroidWifiState.ENABLED : AndroidWifiState.DISABLED;
#else
        return AndroidWifiState.UNKNOWN;
#endif
    }

    /// <summary>
    /// Add a new network description to the set of configured networks. The networkId field of the supplied configuration object is ignored.
    ///  The new network will be marked DISABLED by default. To enable it, called enableNetwork
    /// </summary>
    /// <param name="wifiConfiguration">The set of variables that describe the configuration</param>
    /// <returns>The ID of the newly created network description. This is used in other operations to specified the network to be acted upon. Returns -1 on failure.</returns>
    public static int AddNetwork(AndroidWifiConfiguration wifiConfiguration)
    {
#if PLATFORM_SUPPORTED
        try
        {
            using (var activity = GetActivity())
            using (var wifiManager = GetWiFiManager(activity))
```

```
            {
                var configurations = GetConfiguredNetworks();
                int index = 0;
                for (int i = 0; i < configurations.Count; i++)
                {
                    if (configurations[i].networkId == wifiConfiguration.networkId)
                    {
                        index = i;
                        break;
                    }
                }
                using (var androidList = wifiManager.Call<AndroidJavaObject>("getConfiguredNetworks"))
                {
                    return wifiManager.Call<int>("addNetwork", androidList.Call<AndroidJavaObject>("get",
index));
                }
            }
        }
        catch (Exception e)
        {
            Debug.LogException(e);
        }
#endif
        return -1;
    }


    /// <summary>
    /// Remove the specified network from the list of configured networks
    /// </summary>
    /// <param name="networkId">The ID of the network as returned by AddNetwork() or
GetConfiguredNetworks().</param>
    /// <returns>True if the operation succeeded</returns>
    public static bool RemoveNetwork(int networkId)
```

```
    {
#if PLATFORM_SUPPORTED
        try
        {
            using (var activity = GetActivity())
            using (var wifiManager = GetWiFiManager(activity))
            {
                wifiManager.Call<bool>("disconnect");
                return wifiManager.Call<bool>("removeNetwork", networkId);
            }
        }
        catch (Exception e)
        {
            Debug.LogException(e);
        }
#endif
        return false;
    }


    /// <summary>
    /// Allow a previously configured network to be associated with. If attemptConnect is true, an attempt to connect to the selected network is initiated.
    /// </summary>
    /// <param name="networkId">The ID of the network as returned by AddNetwork() or GetConfiguredNetworks().</param>
    /// <param name="attemptConnect">Initate a connection to the network</param>
    /// <returns>True if the operation succeeded</returns>
    public static bool EnableNetwork(int networkId, bool attemptConnect)
    {
#if PLATFORM_SUPPORTED
        try
        {
            using (var activity = GetActivity())
```

```
            using (var wifiManager = GetWiFiManager(activity))
            {
                return wifiManager.Call<bool>("enableNetwork", networkId, attemptConnect);
            }
        }
        catch (Exception e)
        {
            Debug.LogException(e);
        }
#endif
        return false;
    }

    /// <summary>
    /// Disables a network based off of it's networkId
    /// </summary>
    /// <param name="networkId">The ID of the network as returned by AddNetwork() or
GetConfiguredNetworks().</param>
    /// <returns>True if the operation suceeded</returns>
    public static bool DisableNetwork(int networkId)
    {
#if PLATFORM_SUPPORTED
        try
        {
            using (var activity = GetActivity())
            using (var wifiManager = GetWiFiManager(activity))
            {
                return wifiManager.Call<bool>("disableNetwork", networkId);
            }
        }
        catch (Exception e)
        {
```

```
            Debug.LogException(e);
        }
#endif
        return false;
    }

    /// <summary>
    /// Disconnects from the currently connected to network
    /// </summary>
    /// <returns>True if the operation succeeded</returns>
    public static bool Disconnect()
    {
#if PLATFORM_SUPPORTED
        try
        {
            using (var activity = GetActivity())
            using (var wifiManager = GetWiFiManager(activity))
            {
                return wifiManager.Call<bool>("disconnect");
            }
        }
        catch (Exception e)
        {
            Debug.LogException(e);
        }
        return false;
#elif UNITY_EDITOR
        s_debugConnectedNetwork = string.Empty;
        return true;
#else
        return false;
#endif
```

```
        }

        /// <summary>
        /// Connect to the specified SSID using the supplied password.
        /// </summary>
        /// <param name="ssid">The network SSID obtained through GetScanResults or
GetConfiguredNetworks</param>
        /// <param name="password"></param>
        /// <returns></returns>
        public static ConnectResult Connect(string ssid, string password = "")
        {
#if PLATFORM_SUPPORTED
            try
            {
                AndroidWifiConfiguration existing = GetConfiguredNetworks().Find(q => q.SSID == ssid);
                AndroidWifiScanResults scanResult = GetScanResults().Find(q => q.SSID == ssid);
                using (var activity = GetActivity())
                using (var wifiManager = GetWiFiManager(activity))
                {
                    using (var wifiConfig = new AndroidJavaObject("android.net.wifi.WifiConfiguration"))
                    {
                        wifiConfig.Set("SSID", string.Format("\"{0}\"", ssid));
                        // determine the security type of the network
                        AndroidWifiSecurityType securityType = AndroidWifiSecurityType.WPA2;
                        if (scanResult != null)
                        {
                            securityType = scanResult.securityType;
                        }
                        else if (string.IsNullOrEmpty(password))
                        {
                            securityType = AndroidWifiSecurityType.OPEN;
                        }
```

```
                switch (securityType)
                {
                   case AndroidWifiSecurityType.WPA:
                   case AndroidWifiSecurityType.WPA2:
                      {
                         wifiConfig.Set("preSharedKey", string.Format("\"{0}\"", password));
                         break;
                      }
                   case AndroidWifiSecurityType.WEP:
                      {
                         try
                         {
                            password = string.Format("\"{0}\"", password);
                            wifiConfig.Set("wepTxKeyIndex", 0);
                            wifiConfig.Set("wepKeys", StringArrayToJavaArray(new string[] { password }));
                            using (var groupCipher = new AndroidJavaObject("java.util.BitSet"))
                            {
                               groupCipher.Call("set", 0);
                               wifiConfig.Set("allowedGroupCiphers", groupCipher);
                            }
                            using (var allowedKey = new AndroidJavaObject("java.util.BitSet"))
                            {
                               allowedKey.Call("set", 0);
                               wifiConfig.Set("allowedKeyManagement", allowedKey);
                            }
                         }
                         catch (Exception ex)
                         {
                            Debug.LogError("Error connecting to WEP network. It's possible it was
deprectated in Android API 28.");
                            Debug.LogException(ex);
                            return ConnectResult.UNSUPPORTED_PLATFORM;
```

```
            }
            break;
        }
    case AndroidWifiSecurityType.OPEN:
        {
            using (var allowedKey = new AndroidJavaObject("java.util.BitSet"))
            {
                allowedKey.Call("set", 0);
                wifiConfig.Set("allowedKeyManagement", allowedKey);
            }
            break;
        }
}
if (existing != null)
{
    wifiManager.Call<bool>("removeNetwork", existing.networkId);
}
int networkId = wifiManager.Call<int>("addNetwork", wifiConfig);
if (networkId < 0)
{
    return ConnectResult.ADD_NETWORK_FAILED;
}
if (!wifiManager.Call<bool>("disconnect"))
{
    return ConnectResult.DISCONNECT_FAILED;
}
if (!wifiManager.Call<bool>("enableNetwork", networkId, true))
{
    return ConnectResult.ENABLE_NETWORK_FAILED;
}
if (!wifiManager.Call<bool>("reconnect"))
{
```

```
                    return ConnectResult.CONNECT_FAILED;
                }
                return ConnectResult.SUCCESS;
            }
        }
    }
    catch (Exception e)
    {
        Debug.LogException(e);
        return ConnectResult.CONNECT_FAILED;
    }
#elif UNITY_EDITOR
    s_debugConnectedNetwork = ssid;
    return ConnectResult.SUCCESS;
#else
    return ConnectResult.UNSUPPORTED_PLATFORM;
#endif
    }
    #endregion
  }
}
```

AndroidWifiConfigurator.cs

```
using UnityEngine;

namespace FSG.Android.Wifi
{
  /// <summary>
  /// C# representation of the WifiConfiguration Android class
  /// </summary>
  public class AndroidWifiConfiguration
  {
    /// <summary>
```

```
    /// Possible status of a network configuration.
    /// </summary>
    public enum Status
    {
        CURRENT = 0,
        DISABLED = 1,
        ENABLED = 2
    }

    // When set, this network configuration entry should only be used when associating with the AP having the specified BSSID.
    public string BSSID;
    // Fully qualified domain name of a Passpoint configuration
    public string FQDN;
    // The network's SSID.
    public string SSID;
    // This is a network that does not broadcast its SSID, so an SSID-specific probe request must be used for scans.
    public bool hiddenSSID;
    // Flag indicating if this network is provided by a home Passpoint provider or a roaming Passpoint provider.
    public int networkId;
    // Pre-shared key for use with WPA-PSK.
    public string preSharedKey;
    // Name of Passpoint credential provider
    public string providerFriendlyName;
    // The current status of this network configuration entry.
    public Status status;

    public static AndroidWifiConfiguration FromAndroidObject(AndroidJavaObject javaObject)
    {
        return new AndroidWifiConfiguration()
        {
```

```
            BSSID = javaObject.GetFieldSafe<string>("BSSID"),
            FQDN = javaObject.GetFieldSafe<string>("FQDN"),
            SSID = javaObject.GetFieldSafe<string>("SSID"),
            hiddenSSID = javaObject.GetFieldSafe<bool>("hiddenSSID"),
            networkId = javaObject.GetFieldSafe<int>("networkId"),
            preSharedKey = javaObject.GetFieldSafe<string>("preSharedKey"),
            providerFriendlyName = javaObject.GetFieldSafe<string>("providerFriendlyName"),
            status = (Status)javaObject.GetFieldSafe<int>("status"),
        };
    }
  }
}
```

AndroidExtention.cs

```
using UnityEngine;

namespace FSG.Android.Wifi
{
  public static class AndroidExtensions
  {
    /// <summary>
    /// Calls Get<T>() safely on an AndroidJavaObject. This is to safeguard against an outstanding bug in Unity 2018.2.x
    /// which crashes Unity if the Get call returns a null string
    /// See here: https://issuetracker.unity3d.com/issues/android-application-crashes-when-native-function-returns-null
    ///
    /// Also some versions of Android don't have all the properties we are looking for
    /// </summary>
    public static T GetFieldSafe<T>(this AndroidJavaObject javaObject, string fieldName, bool isStatic = false, params object[] args)
    {
      try
```

```
        {
            if (typeof(T) == typeof(string))
            {
#if UNITY_2018_2
                Debug.LogErrorFormat("This version of Unity is currently not supported. See {0}",
                    "https://issuetracker.unity3d.com/issues/android-application-crashes-when-native-function-returns-null");
                return (T)System.Convert.ChangeType("UNITY 2018.2 NOT SUPPORTED", typeof(T));
#endif
            }
            return javaObject.Get<T>(fieldName);
        }
        catch (System.Exception ex)
        {
            Debug.LogException(ex);
            Debug.LogWarningFormat("Error getting java field \"{0}\". This version of Android might not have it.", fieldName);
            return default(T);
        }
    }
  }
}
```