Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ПОЯСНЮВАЛЬНА ЗАПИСКА
Дипломної роботи
магістра
(назва освітньо-кваліфікаційного рівня)

галузь знань *12 Інформацімайні технології*
(шифр і назва галузі знань)
спеціальність *125 Кібербезпека*
(код і назва спеціальності)
освітній ступень *магістр*
(назва освітньої програми)
освітньо-наукова програма *кібербезпека*

на тему: Засоби підвищення ефективності захисту інформаційної системи «Розумний будинок»

Виконавець: студент ІІ курсу, групи КБм-21
Найденко Ілля Миколайович
(підпис) (прізвище ім'я по-батькові)

	Прізвище, ініціали	Оцінка	Підпис
Науковий керівник	**Наконечний В. С.**		
Рецензент	**Сайко В.Г.**		
Нормоконтроль	**Даков С.Ю.**		

Київ 2022

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри кібербезпеки
та захисту інформації
_____ Н.В. Лукова-Чуйко
«__» _____ 2021 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

студенту КБм-21 *Найденку Іллі Миколайовичу*
(група) (прізвище ім'я по-батькові)

Тема дипломної роботи Засоби підвищення ефективності захисту інформаційної системи «Розумний будинок»

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 5 від 29.10.2021

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	Процес підвищення ефективності захисту системи «Розумний будинок».
Предмет досліджень	Сучасні методи підвищення ефективності захисту розумного будинку та їх можливості.
Мета	Пошук, створення та впровадження засобів підвищення ефективності захисту інформаційної системи «Розумний будинок» для керування пристроями, що можуть бути під'єднані до цієї мережі.
Вихідні дані для проведення роботи	Методи захисту веб-додатків та мобільних лодатків

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна	Полягає у застосуванні архітектури безпеки, що базується на

блокчейн технологіях, а саме на smart-контракті для вдосконалення захисту обраної підсистеми та впровадження змін до вже працюючої інформаційної системи «Розумний будинок».

Практична цінність За допомогою smart-контракту впроваджується ще один шар захисту інформаційної системи «Розумний будинок».

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	**Строки виконання робіт (початок-кінець)**
Розробка плану для досягнення мети роботи	29.10.2021 – 23.01.2022
Аналіз літературних джерел	24.01.2022 – 14.02.2022
Розробка методу захисту від витоку даних платіжних карток через інтернет-браузер	15.02.2022 – 24.04.2022
Оформлення і друк пояснювальної записки	25.04.2022 – 19.05.2022

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект Зниження збитків через викрадення даних

Соціальний ефект Покращення технологій забезпечення захисту інформації як особисто так і на підприємствах.

7. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ _____
(підпис) (прізвище, ініціали)

Завдання прийняв
до виконання _____ _____
(підпис) (прізвище, ініціали)

Дата видачі завдання: _____
Термін подання дипломної роботи до ЕК _____

РЕФЕРАТ

Пояснювальна записка: 42 с., 5 рис., 1 табл., 1 додаток, 22 джерела.

Об'єкт дослідження: процес підвищення ефективності захисту системи «Розумний будинок».

Предмет дослідження: сучасні засоби підвищення ефективності захисту розумного будинку та їх можливості.

Мета роботи (проекту): пошук, створення та впровадження засобів підвищення ефективності захисту інформаційної системи «Розумний будинок» для керування пристроями, що можуть бути під'єднані до цієї мережі.

Методи дослідження: системний підхід, методи порівняння, індексний метод, структурний аналіз, кореляційно-регресійний аналіз.

В роботі проведено аналіз: основних систем захисту та компонентів інформаційної безпеки інформаційної системи «Розумного будинку».

Побудовано: архітектуру підвищення захисту ситеми «Розумний будинок».

Розроблено: основні коди програм, за рахунок яках працює інформаційна система, де HTML файл відповідає за клієнтську частину, JAVA SCRIPT файл за серверну, smart-контракт на допомогою якого забеспечується захист критичних вузлів інфраструктури.

Практичне значення роботи полягає у тому, що за допомогою smart-контракту впроваджується ще один шар захисту інформаційної системи «Розумний будинок».

Результати здійснених у дипломному проекті досліджень можуть бути використані при вдосконаленні будь-якої інтегрованої системи «Розумний будинок», що має можливість реалізувати обрану архітектуру захисту за допомогою блокчену.

Наукова новизна дослідження полягає у застосуванні архітектури безпеки, що базується на блокчейн технологіях, а саме на smart-контракті для вдосконалення захисту обраної підсистеми та впровадження змін до вже працюючої інформаційної системи «Розумний будинок».

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

AES	–	Advanced Encryption Standard
API	–	Application Programming Interface
(D)DoS	–	(Distributed) Denial-of-Service
IEEE	–	Institute of Electrical and Electronics Engineers
IoT	–	Internet-of-Things
IPS	–	Intrusion Prevention System
IaaS	–	Infrastructure as a service
IT	–	Information Technology
PaaS	–	Platform as a service
SSL	–	Secure Sockets Layer
SaaS	–	Software as a service
TLS	–	Transport Layer Security
VPN	–	Virtual Private Network
ДПД	–	Діаграма потоків даних
ІКТ	–	Інформаційно-комунікаційні технології
ПЗ	–	Програмне забезпечення
ЦОД	–	Центр обробки даних

ЗМІСТ

ВСТУП

Будь-який будинок, незалежно від свого призначення, потребує догляду і створення декількох систем для контролю його функціонування.

Особливим попитом стали користуватися технології з віддаленим рівнем доступу. Вони створені з метою економії часу, вчать раціонально ними розпоряджатися. Автоматизовані системи допоможуть виконати ряд нескладних функцій, знаходячись на значній відстані. Яскравим прикладом є система «Розумний будинок», яка реалізує потребу людини в комфорті і домашньому затишку. Зростаюча популярність цих систем пов’язана також із забезпеченням безпеки житла за рахунок впровадження протипожежної системи або системи оповіщення про злом завдяки підключенню звукової сигналізації.

Завдяки вбудованій системі «Розумний будинок» можна не турбуватися про збереження свого житла, перебуваючи далеко від дому, контролювати виконання певних процесів за рахунок їх автоматизації. Ця система ще не досконала, але дуже перспективна, адже приказка «Мій будинок – моя фортеця» з даною системою стає актуальною. Є можливість проаналізувати роботу систем, які об’єднані під поняттям «Розумний будинок» і зробити їх більш досконалими, вивести на вищий рівень розвитку.

Але із розвитком та поширенням технологій інтернету речей, збільшується кількість кібератак на цю галузь. Саме тому більшість компаній наразі тестують та впроваджують різні методи захисту мобільних та веб-додатків.

Актуальність дослідження полягає у застосуванні архітектури безпеки, що базується на блокчейн технологіях, а саме на smart-контрактах для вдосконалення захисту обраної підсистеми та впровадження змін до вже працюючої інформаційної системи «Розумний будинок».

Мета магістерської кваліфікаційної роботи – пошук, створення та впровадження засобів підвищення ефективності захисту інформаційної системи

«Розумний будинок» для керування пристроями, що можуть бути під’єднані до цієї мережі.

Для розв’язання зазначеної мети роботи необхідно виконання відповідних завдань. Їх кілька, але всі вони важливі для розробки кінцевого проекту «Розумний будинок»:

- Дослідження архітектури інтелектуальної системи «Розумний будинок» і основ його функціонування.
- Аналіз наявних засобів забезпечення безпеки додатків.
- Вибір найбільш доцільних, вже працюючих, варіантів для створення власного оптимального ефективного методу захисту інформаційної безпеки розумний будинок.

РОЗДІЛ 1
ПОНЯТТЯ РОЗУМНИЙ БУДИНОК

Розумний будинок – це сучасний житловий будинок, оснащений інтелектуальною системою управління, яка забезпечує комфорт і безпеку своїм власникам за рахунок впровадження сучасних технологій і пристроїв у всіх сферах побуту, і здійснення контролю за їх ефективної і злагодженої роботою [5].

Унікальність даної системи в тому, що всі процеси життєзабезпечення не розрізнені, а навпаки взаємопов'язані між собою за рахунок використання комплексу програмно-апаратних засобів. Він істотно підвищує надійність і ефективність функціонування всіх підключених систем.

Головна особливість системи «розумний дім» - це здатність швидко реагувати на зміну звичного ритму роботи однієї з систем і особливим чином на це реагувати, змінюючи роботу інших систем при виникненні необхідності по заздалегідь прописаним алгоритмам взаємодії. Таким чином, утворюється єдиний керуючий комплекс [5].

Необхідно пам'ятати, що спочатку саме людина задає початкові необхідні параметри роботи шляхом введення певних команд, а автоматична система стежить за коректною роботою електроприладів і контролюючих інженерних систем. Все управління скоординовано в одному центрі. Завдяки цьому відпадає необхідність в декількох пультах, в десятках вимикачів по всьому будинку і окремих комунікаційних блоках для управління системами внутрішнього функціонування, наприклад, опалювальної, вентиляційної, пожежної і т.п.

У систему зазвичай закладено кілька програм з певним сценарієм, розрахованим під час доби, сезону, погоди або настрію. Може бути розбивка в залежності від знаходження в певній частині будинку. Для вибору будь-якого з заздалегідь відомих сценаріїв досить лише натиснути певну кнопку на пульті управління.

Творці системи «Розумний будинок» ставили перед собою виконання кількох першочергових завдань [6]:

1. Об’єднання всіх інженерних систем, які відповідають за різні параметри забезпечення нормального функціонування будинку (вентиляцію, опалення, кондиціонування, водопостачання, освітлення та інші) в єдину інтегровану систему управління будівлею.

2. Заміщення великого числа штатних співробітників, які відповідали за роботу строго визначеної системи, і передача контролю, аналізу та прийняття рішень по коригуванню роботи підвідомчих підсистем загальній системі управління або так званому «інтелекту» будівлі. Саме злагоджена робота цих систем реагує на будь-яку зміну стандартних параметрів роботи датчиків і відхилення роботи приладів від звичайного ритму.

3. Можливість в будь-який момент відключити всю систему або певну її підсистему для передачі під контроль людини. Створення простого і зрозумілого доступу і алгоритму роботи до системи управління «Розумний будинок».

4. Відокремлене нормальне функціонування підконтрольних підсистем при виході з ладу однієї з них або загальної для всіх контролюючої системи.

5. Зведення до мінімуму витрат на процеси вдосконалення вже існуючих підсистем та їх технічне обслуговування шляхом використання загальноприйнятих стандартів при їх побудові, наявність автоматичних налаштувань, і закладення можливості внесення змін у вже існуючу систему. При цьому не повинна страждати якість і швидкість виконання команд.

6. Використання раніше закладених комунікаційних шляхів для підключення пристроїв і модулів, якщо система встановлюється в уже збудоване приміщення. При неможливості проведення цих операцій використання інших додаткових фізичних засобів, наприклад, радіоканалів, силових і слабкострумових ліній.

1.1 Загальний огляд вразливостей «розумних» будинків

Кілька років тому в університеті Мічігану провели дослідження модельного «розумного» будинку, у ході якого було встановлено і підключено до інтернету 18 різних пристроїв: ліжко, світильники, замки, телевізор, кавоварка, зубна щітка та інше. Однією з головних цілей дослідження було виявлення основних уразливостей інтелектуальних систем управління домом. Зокрема, тестували продукцію компанії з назвою SmartThings [7].

Після проведення безлічі різнорідних атак на пристрої цього «розумного» будинку, фахівці зафіксували два основних типи вразливості: надмірні дозволи та небезпечні повідомлення.

Щодо надлишкових дозволів або прав з'ясувалися досить дивні та неприпустимі речі: близько половини встановлених додатків мають доступ до набагато більшої кількості даних та можливостей, ніж це необхідно. Крім того, при взаємодії з фізичними пристроями програми обмінювалися повідомленнями, які містили конфіденційні відомості [7].

Так, додаток контролю рівня заряду автоматичного замку отримувало ще й PIN-код для його розблокування. Програмне забезпечення (ПЗ) деяких «розумних» пристроїв генерувало повідомлення, подібні до реальних сигналів від фізичних пристроїв. Такий підхід давав зловмисникам можливість передавати до мережі недостовірну інформацію. В результаті користувач, наприклад, міг бути впевнений, що двері заблоковані, а вони насправді відчинені.

Крім надмірних дозволів і небезпечних повідомлень, розкрилася ще одна істотна проблема - передача конфіденційної інформації на сервери компаній, що займаються технічною підтримкою цих пристроїв. Тобто гаджети «стежили» за своїми господарями, щохвилини надсилаючи інформацію про їхню взаємодію з пристроями на сервер. Завдяки цій інформації можна відновити точний розпорядок дня мешканців — коли вони прокинулися, почистили зуби, скільки телевізійних каналів дивилися. За два місяці досліджень «розумного» будинку в цифровому ефірі не було жодної хвилини тиші.

Часто розробники залишають собі бекдор, який дозволяє отримати повний доступ або контроль над пристроєм. Виробники виправдовуються необхідністю надавати технічну підтримку користувачам, проте створення таких навмисно створених уразливостей суперечать практикам захисту інформації і є вразливістю. Те, що практично всі виробники програмного забезпечення цим грішать, підтверджується наступним фактом – на конференції Hope X експерт з IT безпеки Джонатан Здзярскі (Jonathan Zdziarski) повідомив про присутність бекдора в операційній системі iOS, існування якого визнала і сама Apple, але назвала його «діагностичним інструментом » [7].

1.2 Аналіз існуючих вразливостей систем «Розумний будинок» Атаки на «розумні» замки

На хакерській конференції DEF CON 24 дослідники Ентоні Роуз (Anthony Rose) та Бен Ремсі (Ben Ramsey) із Merculite Security розповіли, як у рамках експерименту проводили атаки на шістнадцять моделей смарт-замків. Результат виявився досить невтішним: лише чотири змогли встояти перед зломом [8].

Замки одних вендорів передавали паролі відкрито, у незашифрованому вигляді. Так що зловмисники могли легко перехопити їх, використовуючи Bluetooth-сніффер. Декілька замків попалися на методі повторного відтворення: дверима можна було маніпулювати за допомогою заздалегідь записаних сигналів відповідних команд. У світлі поширення голосових помічників дедалі актуальнішим стає і злом розумного замку через голосові команди. Декілька років тому з'ясувалося, наприклад, що якщо господарський гаджет лежить досить близько до зачинених дверей, то досить голосно промовивши через двері «Привіт, Сирі, відчини двері», вас можуть впустити [8].

Поширеним сценарієм злому більшості «розумних» замків є наступний: при отриманні сторонньою особою фізичного доступу до замку натисканням кнопок на ньому можна зробити авторизацію будь-яких гаджетів. Інший цікавий експеримент

дослідників із Pen Test Partners був присвячений перевірці захищеності замків марки Tapplock. Як з'ясувалося, їх можна розблокувати без відбитка пальця власника.

Справа в тому, що коди розблокування генеруються на підставі MAC-адреси пристрою в мережі BLE. А оскільки адреса перетворюється з використанням застарілого алгоритму MD5, він легко може бути з'ясований. Оскільки Bluetooth-замки мають властивість розголошувати свої MAC-адреси по BLE, то зловмисник здатний дізнатися адресу, зламати її з використанням вразливості MD5 та отримати хеш для розблокування замка.

Атаки на відеокамери

Причин уразливості у відеокамер кілька [8]:

- занадто простий чи застарілий механізм захисту;
- стандартні паролі, які часто знаходяться у відкритому доступі в інтернеті;
- при підключенні до камер через «хмару» клієнтські програми надсилають дані в незашифрованому вигляді;
- незмінний майстер-пароль від виробника.

Часто камери атакують методом man-in-the-middle, вбудовуючись між клієнтом та сервером. У такий спосіб можна не тільки читати та змінювати повідомлення, а й підміняти відеопотік. Особливо у тих системах, де не підтримується протокол HTTPS.

Атаки на розетки та лампочки

Те, що лампочки та розетки знаходяться в одній локальній мережі з іншими пристроями, дає хакерам доступ до секретної інформації. Допустимо, ваш будинок висвітлюють "розумні" лампочки Philips Hue. Це досить поширена модель. Однак у мосту Hue Bridge, через який лампочки спілкуються один з одним, існувала вразливість. І траплялися випадки, коли через цю вразливість зловмисники могли дистанційно перехопити контроль над роботою ламп [9].

Атакам схильні і «розумні» розетки. Наприклад, у моделі SP-1101W компанії Edimax для захисту сторінки з налаштуваннями застосовувався лише логін та пароль, причому виробник ніяк не пропонував змінити дані за умовчанням. Це наводить на підозри, що ті самі паролі використовувалися на переважній більшості

пристроїв цієї компанії. До того ж ще й шифрування під час обміну даними між сервером виробника та клієнтською програмою відсутнє. Це може призвести до того, що зловмисник зможе прочитати будь-які повідомлення або навіть перехопити керування пристроєм, наприклад, для підключення до DDoS-атак [9].

Атаки на смарт-ТВ

Ще одна загроза безпеці персональних даних стосується «розумних» телевізорів. ПО телевізора набагато складніше, ніж у камер або замків. Отже, варіантів для зламу системи більше.

Вбудовані в ТВ браузери зазвичай слабо захищені, і можна здійснити атаку користувачеві підроблені сторінки, збираючи паролі, інформацію про банківські картки та інші конфіденційні дані.

Відома історія з телевізорами Samsung: користувачі скаржилися на те, що вбудована система розпізнавання голосу дозволяє стежити за їх розмовами. Виробник навіть вказав у угоді користувача, що слова, сказані в присутності телевізора, можуть бути передані третій стороні [9].

Висновки за розділом 1

При створенні системи «розумного» будинку варто приділяти увагу до компонентів та їх вразливостей. Підключені до системи пристрої так чи інакше схильні до ризику злому.

Адміністраторам та інсталяторам, а також просунутим користувачам таких систем можна порадити наступне [10]:

- уважно вивчіть повний функціонал пристрою: що він здатен робити, які має дозволи, яку інформацію отримує та надсилає, необхідно відключити все непотрібне;
- регулярно оновлюйте версію прошивки та вбудованого ПЗ;
- використовуйте паролі, що містять цифри, літери та спеціальні знаки;
- всюди, де це можливо, вмикайте двофакторну автентифікацію;

- всі порти мають бути закритими, якщо вони не використовуються, а відкриті потрібно захищати стандартними методами авторизації через налаштування операційної системи;
- захищати за допомогою SSL вхід через інтерфейс користувача, у тому числі з веб-доступом;
- потрібно виключити фізичний доступ сторонніх до "розумного" пристрою.

РОЗДІЛ 2
ЗАХИСТ МОБІЛЬНИХ ТА ВЕБ-ДОДАТКІВ

2.1. Вразливості Web-додатків.

Вразливості безпеки веб-застосунків залишаються однією з найпоширеніших слабкостей безпеки. Інші поширені проблеми включають низьку поінформованість в галузі інформаційної безпеки (ІБ), слабкі політики паролів або їх повсюдне недотримання, недоліки в процесах управління оновленням програмного забезпечення, використання небезпечних конфігурацій та, як не парадоксально, неефективне підключення до Інтернету. обмеження доступу. Незважаючи на те, що вразливості веб-застосунків неодноразово описувалися в сучасній науковій та спеціальній літературі, механізми превентивного захисту, що знижують ризик шкоди, зустрічаються рідко. [19, 20].

Проблема безпеки веб-додатків ускладнюється тим, що питання, пов'язані із захистом цих систем від внутрішніх та зовнішніх загроз, часто не беруться до уваги або їм приділяється занадто мало уваги при розробці. Це, у свою чергу, створює ситуацію, коли проблеми з інформаційними системами стають відомі власнику системи вже після завершення проекту, а усунення вразливостей у вже розробленому веб-додатку виявляється більш витратною статтею бюджету, ніж його розробка та впровадження. Недооцінка важливості поінформованості про ризики інформаційних систем, що використовують веб-програми, доступні через Інтернет, є одним з основних факторів низького рівня захисту від більшості з них.

Існують специфічні ризики, пов'язані з використанням інформаційних систем та технологій. Оцінка ризиків необхідна для моніторингу ефективності національної діяльності в галузі ІБ, вжиття відповідних захисних заходів та побудови ефективних систем захисту. Потенційні вразливості та загрози національній безпеці становлять основу потенційних чи актуальних загроз, тому їх своєчасне виявлення – порушення конфіденційності та цілісності інформації, поширення шкідливих програм та

фінансове шахрайство, класифікація загроз для інформаційних систем – є основним завданням захисту інтернет-додатків. Ризики повинні постійно відстежуватися та періодично переоцінюватися. Важливо пам'ятати, що сумлінно проведена та ретельно задокументована перша оцінка може значно полегшити подальші дії. Управління ризиками, як і будь-яка інша діяльність у галузі ІБ, має бути інтегровано у життєвий цикл ІТ-системи.

2.2. Тестування мобільних додатків.

Сьогодні кожна людина нерозривно пов'язана з використанням мобільних пристроїв. Вони потрібні всім поколінням, від школярів до людей похилого віку, оскільки область застосування смартфонів стала надзвичайно широкою. Якоюсь мірою смартфони практично замінили телефони, електронні книги, персональні комп'ютери та ноутбуки у повсякденному використанні. Така широка сфера застосування смартфонів обумовлена їхньою широкою функціональністю та, відповідно, кількістю програмного забезпечення, яке робить використання цього гаджета зручним та необхідним.

Будь-яке програмне забезпечення, перш ніж використовувати його, має бути протестоване. Фахівці, які займаються забезпеченням якості програмного забезпечення, особливо мобільних додатків, у зв'язку з бурхливим розвитком технологій у галузі інформаційних технологій, регулярно порушують питання про вдосконалення процесу тестування, який враховуватиме весь процес раціонального розподілу ресурсів. в рамках життєвого циклу програмного забезпечення, а також раціонального опису початкового етапу та критеріїв завершення тестування. [18].

Необхідність оптимального розподілу ресурсів, обсягу робіт та тривалості всього процесу тестування виникає у двох аспектах [18]:

- бажання клієнта отримати продукт найвищої якості в найкоротші терміни та за найкращою ціною;
- бажання підрядника своєчасно виконати роботу високої якості в обмін на вигідну винагороду.

Таким чином, концепція "залізного трикутника" має місце не тільки у розробці програмного забезпечення загалом, але й у тестуванні програмного забезпечення зокрема.

Тестування мобільних програм має безліч особливостей. Це пов'язано з такими аспектами, як [18]:

- неймовірно велика кількість постачальників, що, своєю чергою, вимагає різних конфігурацій компонентів;
- специфіка та різноманітність операційних систем для мобільних платформ;
- характеристики екрана (порівняно з невеликим ПК, сенсорний інтерфейс, зміна орієнтації);
- функціональність устрою як комунікатора і т.д.

Тестування програмного забезпечення можна класифікувати за кількістю характеристик, і таких класифікацій існує безліч. Більшість видів тестування програмного забезпечення та мобільного тестування однакові, але деякі різняться [17].

Тестування оновлення - воно має бути простим, без додаткових зусиль користувача, з урахуванням різних способів (Wi-Fi, 3G, ПК, SD).

Тестування інтернаціоналізації можуть виникнути проблеми, характерні для мобільної платформи, наприклад, брак вільного простору на екрані.

Тестування простоти використання - це, мабуть, найважливіше за умов жорсткої конкуренції. Досить часто таке тестування проводиться у формі бета-тестування.

Випадкове тестування - програма має правильно реагувати на хаотичні події.

Тести на мультиплатформенності та мультипристрою - програма повинна коректно працювати на всіх конфігураціях всіх пристроїв, для яких вона була розроблена.

Лабораторні випробування – імітація реальних умов якості зв'язку та навколишнього середовища. Зазвичай невідомо, як поведеться програма, наприклад, при нестабільному сигналі Wi-Fi або нульовому рахунку 3G. Цей тип тестування дозволяє переконатися, що таких випадків не станеться.

Сертифікаційні тести використовуються для підтвердження відповідності програми стандартам, ліцензійним угодам та умовам використання.

Тестування мобільних додатків належить до класифікації за природою додатка. Проте мобільні додатки можна класифікувати за належністю до розробки на [17]:

– мобільні веб-додатки – вебсторінки, які відкриваються через браузер мобільних пристроїв, що адаптовані до мобільних пристроїв;

– native apps – це програми, розроблені для однієї конкретної платформи (ios, android, windows та ін.);

– гібрид – це об'єднання мобільних веб-додатків та native app.

Покроковий алгоритм тестування мобільних додатків

Крок 1: Планування

Коли етап розробки програми майже завершений, вам потрібно знову запитати себе, чого ви хочете досягти, створюючи цю програму і які ваші обмеження.

Вам необхідно відповісти на такі запитання:

- Чи взаємодіє ваша програма з іншими?

- Наскільки функціональними є всі функції програми?

- Чи є тестований мобільний додаток нативним, веб-додатком або гібридним?

- Чи обмежується завдання тестування програми тестуванням інтерфейсу?

- Чи потрібно також тестувати бекенд?

- Яка має бути сумісність із різними бездротовими мережами?

- Скільки даних та вільного місця займає програма залежно від характеру її використання?

- Як швидко завантажується програма та як швидко відображаються меню та функції програми?

- Як здійснюватиметься збільшення навантаження на програму?

- Чи впливають різні зміни у статусі та стані телефону на продуктивність мобільного додатка?

Переконайтеся, що ви узгодили з командою тестувальників ролі кожного та ваші очікування від процесу тестування. Зрештою, спілкування - це ключ до підтримки правильної робочої атмосфери у колективі.

Правильне визначення ролей та завдань також застосовується для призначення списку тестових випадків. Уся команда QA повинна підтримувати та оновлювати цей документ, який містить звіти про тестування всіх функціональних можливостей, реалізованих у процесі розробки.

Крок 2. Визначте типи необхідного тестування мобільних додатків

Перед тестуванням будь-якого мобільного додатка важливо визначити, що саме ви хочете в ньому протестувати: набір функцій, зручність використання, сумісність, продуктивність, безпека та багато іншого. На цьому етапі доцільно вибрати методи тестування мобільного додатка.

Визначте, для яких цільових пристроїв призначено програму та які вимоги до функціональності мають бути перевірені. Ви також повинні визначити, які цільові пристрої повинні бути включені до списку тестування.

Це можна зробити так:

- Визначте, які пристрої підтримуватимуться програмою;
- Визначте, яка версія операційної системи підтримуватиметься програмою в першу чергу;
- Визначте найпопулярніші моделі мобільних пристроїв у цільовій групі;
- Визначте набір додаткових пристроїв із різними розмірами екрана, які потенційно можуть підтримуватись програмою;
- Вирішіть, чи використовуватимете ви для тестування фізичні пристрої або емулятори.

Крок 3: Тестові приклади та розробка сценаріїв тестування додатків

Підготуйте документ, який описує тестові випадки для кожної функції та функціональності, що підлягає тестуванню.

Крім тестових випадків функціональності слід також включити деякі специфічні моменти (випадки):

- застосування для конкретного акумулятора;
- швидкість нанесення;
- вимоги до даних;

Крок 4: Ручне та автоматизоване тестування

Тепер настав час провести ручне та автоматизоване тестування.

На попередніх етапах ви вже вирішили, які тести та сценарії використовувати, та підготували їх. На цьому етапі запускаються тести для перевірки основних функціональних механізмів, щоб переконатись у відсутності помилок.

Автоматизоване тестування мобільних програм заощаджує час та інші ресурси тестувальника.

Крок 5: Сертифікація та безпека програми тестування

Безпека та конфіденційність даних сьогодні надзвичайно важливі. Користувачі вимагають, щоб вся їхня інформація була безпечною та конфіденційною.

Переконайтеся, що програма, яку ви тестуєте, є безпечною. Ви повинні перевірити наявність SQL-ін'єкцій, крадіжки сеансів, аналізу дампи даних, аналізу пакетів та SSL-трафіку.

Дуже важливо перевірити безпеку сховища конфіденційності мобільного додатка та те, як воно поводиться залежно від різних схем дозволів пристрою.

Окрім перевірки безумовного шифрування імен користувачів та паролів, задайте собі наступні питання:

- Чи є у додатку сертифікати безпеки?

- Чи використовує програму безпечні мережеві протоколи?

- Чи існують обмеження, наприклад, кількість спроб входу, після якого користувач блокується?

2.3. Види загроз мобільних та WEB – додатків.

Кожна мобільна ОС відрізняється, і в кожній можна знайти велику кількість як нових, так і добре відомих уразливостей.

Підтвердилася тенденція, яку експерти відзначають у традиційних щорічних звітах щодо безпеки RBS. Розробники мобільного банкінгу не приділяють достатньої уваги безпеці додатків і не дотримуються інструкцій щодо безпечної розробки. Процеси розробки безпечного коду та архітектури часто відсутні. Було

виявлено, що всі досліджені програми містять принаймні одну вразливість, яка може або перехоплювати дані, що передаються між клієнтом і сервером, або безпосередньо використовувати вразливості пристрою та самого мобільного додатка.

Наприклад, 35% мобільних банків iOS і 15% мобільних банків Android містять уразливості, пов’язані з несправністю SSL, а це означає, що критичні платіжні дані можуть бути перехоплені за допомогою атаки «людина посередині». 22% програм iOS потенційно вразливі до ін’єкції SQL, що створює ризик крадіжки всієї платіжної інформації за допомогою кількох простих запитів. 70% додатків iOS і 20% додатків для Android потенційно вразливі до XSS, однієї з найпопулярніших атак для введення в оману користувачів мобільного банкінгу і, наприклад, крадіжки їхніх даних аутентифікації. 45% додатків iOS потенційно уразливі до атак XXE, особливо небезпечних для пристроїв, які пройшли популярну в Україні операцію джейлбрейка. Близько 22% додатків Android зловживають механізмами міжпроцессного зв’язку, тим самим ефективно дозволяючи стороннім програмам отримувати доступ до критичних банківських даних.

Класифікація мобільних додатків

Мобільні програми можна класифікувати за різними критеріями, але в контексті безпеки додатків нас цікавить наступне: за місцем розташування програми та за типом використовуваної технології передачі даних.

За місцем застосування:

- SIM-програми - програма на SIM-картці, написана за стандартом SIM Application Toolkit (STK);
- Web-додатки - спеціальна версія веб-сайту;
- мобільні додатки - програми, розроблені для конкретної мобільної ОС за допомогою спеціального API, встановленого в смартфоні.

За типом технології, яка використовується для зв'язку з сервером:

- мережеві програми - використовувати власний протокол зв'язку через TCP/IP, наприклад HTTP;
- SMS-додатки - програми на основі SMS (Short Messaging Service).

- Додаток спілкується з сервером за допомогою коротких текстових повідомлень;
- USSD-додатки - програми, засновані на USSD (Unstructured Supplementary Service Data). Послуга заснована на передачі коротких повідомлень, схожих на SMS, але з низкою відмінностей;
- IVR-додатки - програми, засновані на технології IVR (Interactive Voice Response). Система заснована на попередньо записаних голосових повідомленнях і тоновому набору.

Додатки, розроблені для певної мобільної ОС за допомогою спеціалізованого API, встановленого в смартфоні для взаємодії з відповідним банківським сервісом, зараз є найпоширенішими, оскільки повністю використовують можливості мобільного пристрою та мають найзручніший інтерфейс користувача. Це ті, які ми розглядали в цьому дослідженні.

Методи оцінки безпеки клієнтської програми:

- Динамічний аналіз:

1) налагодження запущеної програми (на емуляторі або пристрої);
2) фазування;
3) аналіз мережевого трафіку;
4) аналіз взаємодії з файловою системою;
5) аналіз пам'яті програми.

- Статичний аналіз: Аналіз вихідного коду (за наявності);

1) Реверс-інжиніринг;
2) Розбирання;
3) Декомпіляції;
4) Аналіз отриманого представлення на наявність недоліків у коді.

Моделі зловмисників

1. Зловмисник з фізичним доступом до клієнтського пристрою. На пристрої не ввімкнено блокування екрана та не використовується шифрування.

2. Зловмисник, який не має доступу до пристрою, знаходиться поруч із жертвою і здатний здійснити атаку «людина посередині».

3. Зловмисник, який завантажив власну шкідливу програму на клієнтський пристрій за допомогою офіційних магазинів програм або іншим способом.

Атаки на стороні сервера нічим не відрізняються від атак на звичайні системи RBS.

Атаки на стороні клієнта можливі, якщо є:

- фізичний доступ до пристрою;
- шкідлива програма на пристрої;
- можливість керувати каналом, напр. в результаті нападу «людина посередині».

Маючи фізичний доступ, зловмисник може отримати доступ до файлової системи. Якщо програма зберігає дані аутентифікації або інші критичні дані в відкритому доступі або критичні дані «витікають» у відкритому доступі, зловмиснику не важко отримати ці дані та вкрасти гроші.

Захист додатків

Використовуйте криптографічні можливості пристрою, шифрування критичних даних і можливість дистанційного очищення даних, якщо це необхідно, а також проведіть аналіз безпеки програми, щоб допомогти виявити можливі витоки критичних даних і неправильне використання шифрування.

Атака зловмисного програмного забезпечення вимагає встановлення шкідливої програми з використанням методів соціальної інженерії або атаки Drive-by-Download.

Після встановлення шкідливої програми зловмисник може підвищити свої привілеї в системі, використовуючи експлойт для вразливості ОС смартфона, і отримати віддалений доступ до пристрою з повними дозволами, що призводить до повної скомпрометації пристрою: зловмисник може вкрасти. важливі дані користувачів мобільного банкінгу або дані підробних платіжних транзакцій.

Захист: оновити програмне забезпечення на пристрої, використовуйте засоби захисту програмного забезпечення та підвищте обізнаність користувачів про проблеми ІБ.

Атаки на посилання: класична атака «людина посередині» перехоплює дані між пристроєм клієнта та сервером. Для цього потрібно перебувати в тій самій мережі,

що й жертва, наприклад у загальнодоступній мережі Wi-Fi, або використовувати підроблені точки бездротового доступу та підроблені базові станції. Уразливість у мобільному додатку потрібна через неправильну обробку шифрування переданих даних або не шифрування даних взагалі. Найпоширенішим прикладом є неправильна обробка SSL. В результаті зловмисник може підслухати та підробити передані дані, що в кінцевому підсумку може призвести до крадіжки коштів з рахунку клієнта.

Захист: правильна реалізація обробки SSL. Також при підключенні до сервера рекомендується довіряти лише сертифікату SSL банку в мобільному додатку. Це допоможе у випадку, якщо кореневий ЦС зламано.

Також варто відзначити, що джейлбрейк пристрою (iOS) або наявність root-доступу на пристрої користувача (Android) значно знижує рівень безпеки пристрою та полегшує атаку зловмисника.

Інші види атак:

1. Міжсайтовий скриптинг - XSS (атака на користувача спрямована на виконання довільного скрипта в його браузері) - використання шкідливого JavaScript-коду в сторінку атакованої веб-системи. Коли сторінка завантажується, браузер автоматично виконує код JavaScript, який збирає реєстраційну інформацію та надсилає її на сайт зловмисника [17].

(2) SQL-ін'єкція - вставлення шкідливого SQL-коду в тіло HTTP-запиту до веб-додатку. Потім рядок URL можна виконати, підставивши SQL-запит у базу даних та дізнавшись пароль адміністратора. Тип SQL-ін'єкції: "http://localhost/path/user.php ? id = -2 + UNION + SELECT + 1,2,3,4,5, concat (user_email, 0x3e, user_passwd), 7,8, 9, 10,11 + від + users-"[17].

3. CRLF-атака – техніка модифікації заголовків HTTP-запитів.

Існує два типи [17]:

- Ін'єкція CRLF - використовує ASCII уявлення CR + LF (переклад каретки + новий рядок) для генерації "шкідливих" URL.

- пакет запиту HTTP. Використовуючи це, зловмисник може згенерувати URL-адресу для заміни відповіді сервера і, спровокувавши внутрішню помилку, побачити інформацію як про сервер веб-програми, так і про службу.

4 - Атака XXE (XML eXternal Entity). Під час перевірки XML-документа парсером (об'єктно-орієнтована мова сценаріїв, призначена для генерації HTML-сторінок на веб-сервері з підтримкою CGI) з використанням схеми DTD всі її директиви повинні бути виконані. Якщо в правилах зазначено, що у вхідному документі дозволено спеціальні символи XML, ризик атаки дуже високий [17].

5. CSRF (Cross Site Request Forgery). Це дозволяє зловмиснику використовувати облікові дані людини (cookies) та виконувати будь-які шкідливі операції від його імені [17].

6. DDoS-атака - велика кількість зовнішніх запитів (часто безглуздих або неправильно сформульованих) з різним трафіком (на різних портах), що викликають переповнення пам'яті, а потім відмову в обслуговуванні системи, під час якої можна легко отримати доступ до ресурсів та отримати root [11].

Таблиця 2.1

Web-атаки та способи запобігання Види загроз

Тип загрози	**Способи запобігання**
XSS Міжсайтовий скриптинг	Заборона вбудовування HTML, захист спеціальних символів, передача всіх cookies з прапором HttpOnly.
SQL Injection	Використання ORM у веб-додатках, перевірка достовірності даних клієнта та сервера, тестування веб-застосунків, перевірка на "дозволені" параметри, що отримуються в неперевірених SQL-запитах, тестування програм за допомогою інструментів для пошуку потенційних SQL-ін'єкцій.
Атака CRLF	Обов'язкове кодування послідовностей CRLF перед їх надсиланням у заголовках HTTP, а також повне шифрування даних, що передаються.
XXE атаки	Атаки, які включають ретельне налаштування аналізаторів XML та використання схеми XML замість DTD для перевірки аналізатора.

Тип загрози	Способи запобігання
CSRF (Підробка запитів)	Встановлення прапорця cookie HttpOnly, використання одноразових токенів сесії та відправлення прихованої форми, щоб токен не можна було підробити, перевірка реферерів для HTTP-запитів до веб-додатку.
DDoS-атака	Уникнути цієї ситуації практично неможливо, але наслідки DDoS-атак та їх ефективність можна значно знизити шляхом правильного налаштування маршрутизаторів та брандмауерів та постійного аналізу аномалій у мережевому трафіку.

У таблиці 2.1 наведені види загроз та методи запобігання цим атакам.

Висновки за розділом 2

Вразливості веб-додатків залишаються однією з найпоширеніших слабких сторін інформаційної безпеки. Проблема безпеки веб-додатків посилюється тим, що при розробці веб-додатків часто не враховуються питання, пов'язані із безпекою цих систем від внутрішніх і зовнішніх загроз, або приділяється недостатньо уваги цьому процесу.

Це, у свою чергу, створює ситуацію, коли проблеми ІС приходять до уваги власника системи після завершення проекту. А виправлення вразливостей у вже створеному веб-додатку є більш дорогим бюджетним елементом, ніж його проектування та впровадження.

РОЗДІЛ 3
ЗАСОБИ ПІДВИЩЕННЯ ЗАХИСТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ДЛЯ КЕРУВАННЯ ОСВІТЛЕНІСТЮ «РОЗУМНИЙ БУДИНОК»

3.1. Використання блокчейнів у безпеці додатків

Блокчейн – це сукупність взаємозалежних транзакцій. Кожна сторона, що бере участь у ланцюжку, підписує свої транзакції сильними ключами, які гарантують, що кожна транзакція була автентифікована [12].

Потім блок вставляються два хеша, один з яких захищає блок від несанкціонованого доступу, а інший захищає транзакцію даних, яка була виконана до додавання хеша. Ці хеші допомагають записувати дані про всі події, що відбулися в блоці, що гарантує, що жодна із сторін, залучених до блоку, не зможе проникнути в нього, не попередивши інших сторін [14].

Коли ми говоримо про захист системи або програми, ми маємо на увазі захист даних, файлів, документів або контрактів, що зберігаються у мобільному додатку.

При збереженні документа або файлу блокчейн створює хеш. Хеш - це функція блокчейна, яка перетворює дані на унікальний результат фіксованої довжини для кожної транзакції [15]. Зокрема, у випадку хеш, кожен блок містить хеш попереднього блоку, який має посилання на хеш, на основі якого буде побудований наступний блок. У блоці зберігається хеш поточної транзакції, який потім об'єднується з наступним хеш при додаванні нового блоку. Справжність блоку можна визначити шляхом перевірки його хеша.

Залучаючи до реєстру групу користувачів, кожен з яких має доступ до перегляду та додавання інформації про кожну транзакцію в режимі реального часу, блокчейн забезпечує прозоре керування реєстром та захист від несанкціонованого доступу.

Для забезпечення внутрішнього зв'язку всередині мобільного додатка Blockchain використовує метадані зв'язку, які розповсюджуються по розподіленому реєстру і не можуть бути зібрані в будь-якій центральній точці, що запобігає злому.

Блокчейн використовує інфраструктуру безпеки без ключів для зберігання всіх паролів даних та запуску алгоритму перевірки паролів для їх підтвердження. Це дозволяє в режимі реального часу виявити будь-які маніпуляції з даними, оскільки вихідний хеш завжди доступний в інших блоках системи/ланцюжка [13].

Основною технологією є використання пов'язаних списків. Зв'язковий список - це організація даних, у якій кожен блок даних (за винятком, можливо, першого та останнього) посилається на один чи два інші. У цьому полягає головна відмінність блокчейна від більшості сучасних сховищ даних, оскільки останні переважно використовують масиви.

Масив - це колекція змінних одного типу із загальним ім'ям, що використовується для їх позначення. У JS масиви може бути як одномірними, і багатовимірними. Масиви служать різних цілей, оскільки вони забезпечують зручний спосіб зв'язування пов'язаних змінних разом.

Основною перевагою масивів є структурна гнучкість: порядок елементів зв'язаного списку може відповідати порядку елементів даних у пам'яті комп'ютера, а порядок проходження списку завжди чітко визначається його внутрішніми зв'язками.

Блокчейн - це розширення пов'язаних списків з одним суттєвим обмеженням: зв'язки між блоками даних та самі дані залишаються незмінними. Найпростіша аналогія – стопка книг.

- книги лежать одна на іншій;

- додавання нової книги без перегину стопки можливе лише у верхній частині стопки;

- Не можна знімати книгу зі стопки, інакше вона розсиплеться.

Справді, блокчейн – це нова організаційна парадигма для координації всіх видів людської діяльності. Можливо це навіть наше майбутнє, про яке варто знати вже сьогодні. [23]

Незмінність з'єднань між блоками даних забезпечується прив'язкою як до блоку, до його даних. Дані кожного блоку є входом хеш-функції. Результатом його виконання є числовий рядок фіксованої довжини.

У принципі, не існує однозначної відповідності між вихідними (вхідними) даними та хеш-кодом (вихідними даними). Значення, що повертаються хеш-функцією (вихідні дані), менш різноманітні, ніж значення вхідного масиву (вхідні дані). Випадок, коли хеш-функція перетворює кілька різних повідомлень на однакові результати, називається "колізією". [23]

Хешування використовується для побудови асоціативних масивів, пошуку дублікатів у рядах даних, створення досить унікальних ідентифікаторів для наборів даних та перевірки агрегатів для виявлення випадкових чи навмисних помилок під час зберігання або передачі даних. [23]

При зміні даних змінюється результат хеш-функції (на прикладі SHA256), що забезпечує безпеку даних за рахунок відстеження змін. Шифрування - це оборотне перетворення інформації з метою приховати її від неавторизованих осіб, дозволяючи у своїй авторизованим користувачам отримати доступ до неї. Основна мета шифрування - захист конфіденційності інформації, що передається. Важливою особливістю будь-якого алгоритму шифрування є використання ключа, що підтверджує вибір конкретного перетворення з безлічі можливих даного алгоритму. [23]

Важливо розуміти, що хешування не еквівалентне шифрування. Шифр створюється так, щоб його можна було використовувати для відновлення вихідного набору даних. На відміну від цього, у разі хешування ця умова є необов'язковою, а в нашому випадку навіть небажаною. Hash = SHA256 (дані);

Незмінність зв'язків між блоками забезпечується тим, кожен наступний блок при хешуванні своїх даних використовує дані попереднього блоку, наприклад, його хеш. Найбільш поширеною причиною є те, що один хеш відповідає нескінченній кількості вхідних даних.

Дуже корисно мати можливість уявити ланцюжок блоків у вигляді таблиці. Це дозволяє використовувати реляційні бази даних для зберігання даних і відповідно

застосовувати всі існуючі методи шифрування, стиснення та захисту даних, які застосовуються до цих баз даних.

На сьогодні ця технологія широко використовується на криптовалютному ринку. Технологія найбільш відома завдяки використанню платіжної платформи Bitcoin. Однак ця платформа використовує не тільки блокчейн, але й багато інших сучасних та перспективних технологій, наприклад, технологію цифрового підпису.

Вже існує безліч розширень для створення бізнес-додатків на основі блокчейну, які забезпечують:

- Безпечне адміністрування мережі, що виключає хакерські атаки MIM ("посередник") та усуває проблему "одного адміністратора";

- Сховище цифрових сертифікатів, що забезпечує повну безпеку доступу користувачів до сайтів (зокрема, що виключає крадіжку паролів);

- можливість проведення двосторонніх угод без участі третьої сторони, що гарантує (юридичної фірми, нотаріуса, банку тощо);

- проставлення тимчасових штампів на документах, що дозволяє вирішувати проблеми, пов'язані з патентами, авторськими правами та ін;

- підтвердження справжності товарів (активів) за допомогою захищеного сертифікату;

- підтвердження прав на якесь майно;

- створення публічних електронних візиток, інформація на яких автоматично оновлюється навіть після "роздачі" інтернет-ресурсами;

- система DNS, стійка до DDOS-атак;

Біткойн - це набір концепцій та технологій, які разом становлять основу екосистеми цифрових грошей. Валюта під назвою біткоїн використовується для зберігання та передачі вартості між учасниками мережі [23].

Переведення коштів у однорангових мережах від відправника до отримувача здійснюється без участі третіх осіб. Це дозволяє уникнути ефекту подвійної оплати та повністю виключити повноваження будь-якого користувача [23].

3.2. Переваги додатків з використанням блокчейну.

Захист даних

Використовуючи криптографію для призначення закритих ключів користувачам, ваші користувачі зможуть зберігати, переглядати та керувати всіма даними та інформацією, що стосується транзакцій, в одному місці.

Прозорість та перевірка даних

Виводячи ідентифікацію даних на новий рівень, Blockchain дозволяє користувачам робити свої дані доступними у децентралізованій системі, до якої кожна із залучених сторін має доступ у режимі реального часу. Блокчейн унеможливлює для будь-якої однієї сторони підробку інформації в блоці та залишення її непоміченим, тим самим підтримуючи цілісність даних і роблячи їх прозорими для всіх, тим самим усуваючи необхідність у посередниках [16].

Захист інфраструктури

Зберігаючи запис DNS у блокчейні, власники мобільних програм можуть перенести ризик злому на захищену платформу блокчейна. Розподілена, прозора DNS, яку пропонує технологія, унеможливлює доступ навіть для уряду без дозволу залучених сторін.

Кінець паролів

Блокчейн дозволяє автентифікувати користувачів, їх пристрої та транзакції, які вони здійснюють, без введення пароля. Його функція децентралізації мережі допомагає досягти консенсусу між залученими сторонами для автентифікації SSL сертифікатів на основі блокчейна.

3.3. Використання блокчейну Solana для забезпечення захисту додатку

Solana як блокчейн має кілька різних рішень, які допоможуть зробити його одним із найпродуктивніших Layer 1s у світі. Команда Solana запропонувала низку

революційних інновацій, кожне з яких пов'язане з безпекою, пропускною здатністю та децентралізацією.

Переваги блокчейну Solana [21]:

- Має перевірку годинника в ланцюжку.
- Потокова передача транзакцій, не чекаючи глобального консенсусу (без шкоди для безпеки).
- Зменшує розмір пакетів даних та передає їх за допомогою UDP (менші вимоги до пам'яті).
- Видаляє мемпул (перенаправляючи непідтверджені транзакції майбутнім валідаторам).

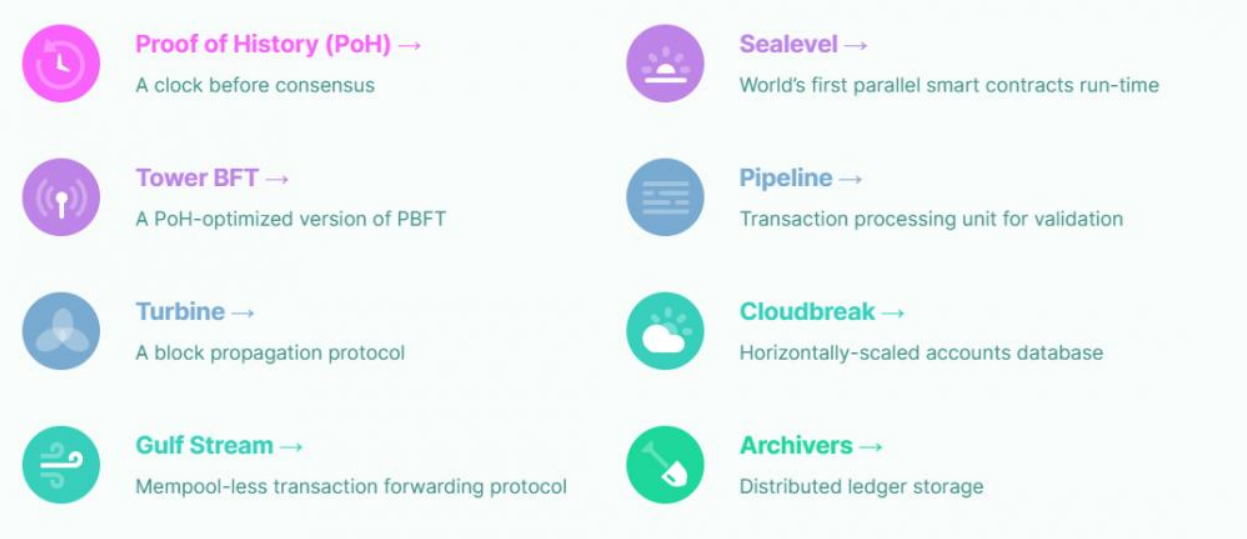

Рисунок 3. 1 – Переваги блокчейну Solana

Однією з ключових рис Solana є система консенсусу Proof of Stake (PoS), яка підтримується так званим Tower Consensus. Це різновид системи, відомої як Practical Byzantine Resilience (PBFT), яка дозволяє розподіленим мережам досягати консенсусу, незважаючи на атаки з боку шкідливих вузлів [21].

Реалізація PBFT компанією Solana забезпечує глобальне джерело часу блокчейну, використовуючи другий новий протокол, відомий як Proof of History (PoH). По суті, він забезпечує хроніку попередніх подій у блокчейні, гарантуючи наявність глобального запису у тому, що коли відбулося [21].

Tower Consensus використовує цей синхронізований годинник для зниження обчислювальної потужності, необхідної для перевірки транзакцій, оскільки більше не потрібно обчислювати тимчасові мітки попередніх транзакцій. Це дозволяє Solana досягти пропускної спроможності, яка перевершує більшість конкурентів [21].

Крім того, Solana включає ряд інших інновацій, які дозволяють їй виділятися на тлі своїх конкурентів [21]. Серед них - технологія розпаралелювання транзакцій, відома як Sealevel. Вона забезпечує паралельне середовище виконання смарт-контрактів, яке оптимізує ресурси та надає можливість горизонтального масштабування Solana між процесорами та твердотілими накопичувачами.

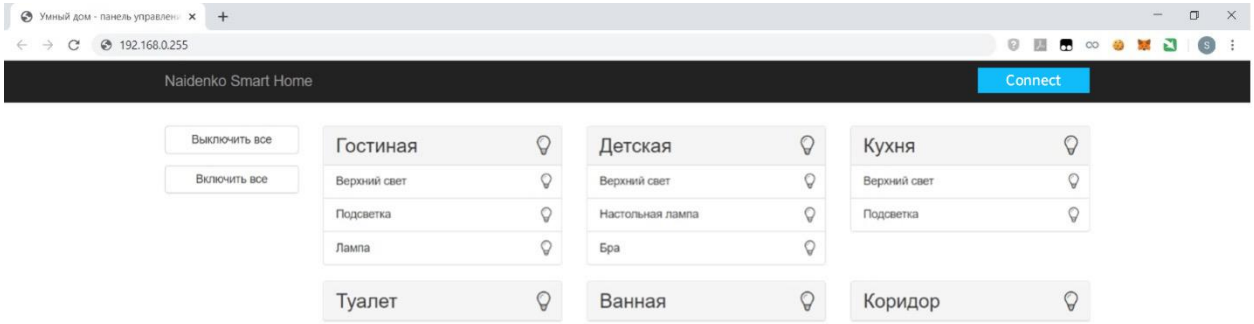

Рисунок 3. 2 – Інтерфейс у вимкненому стані

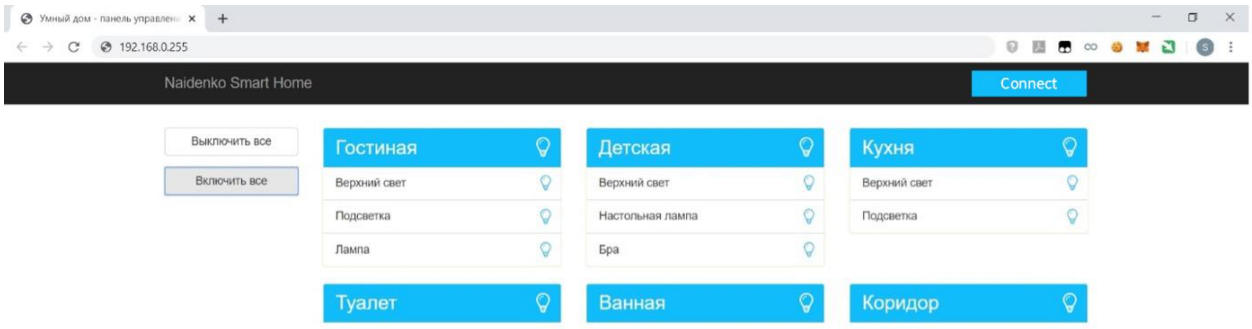

Рисунок 3. 3 – Інтерфейс у ввімкненому стані

Функції додатку працюють лише якщо користувач авторизований в систему та аутинтифікований за адрессою свого Phantom Wallet. Для цього використовується Phantom Wallet, який функціонує на блокчейні Solana.

При натисканні кнопки Connect у веб-додатку відкрівається розширення Phantom Wallet веб-браузеру, а на смартфоні – у мобільному додатку.

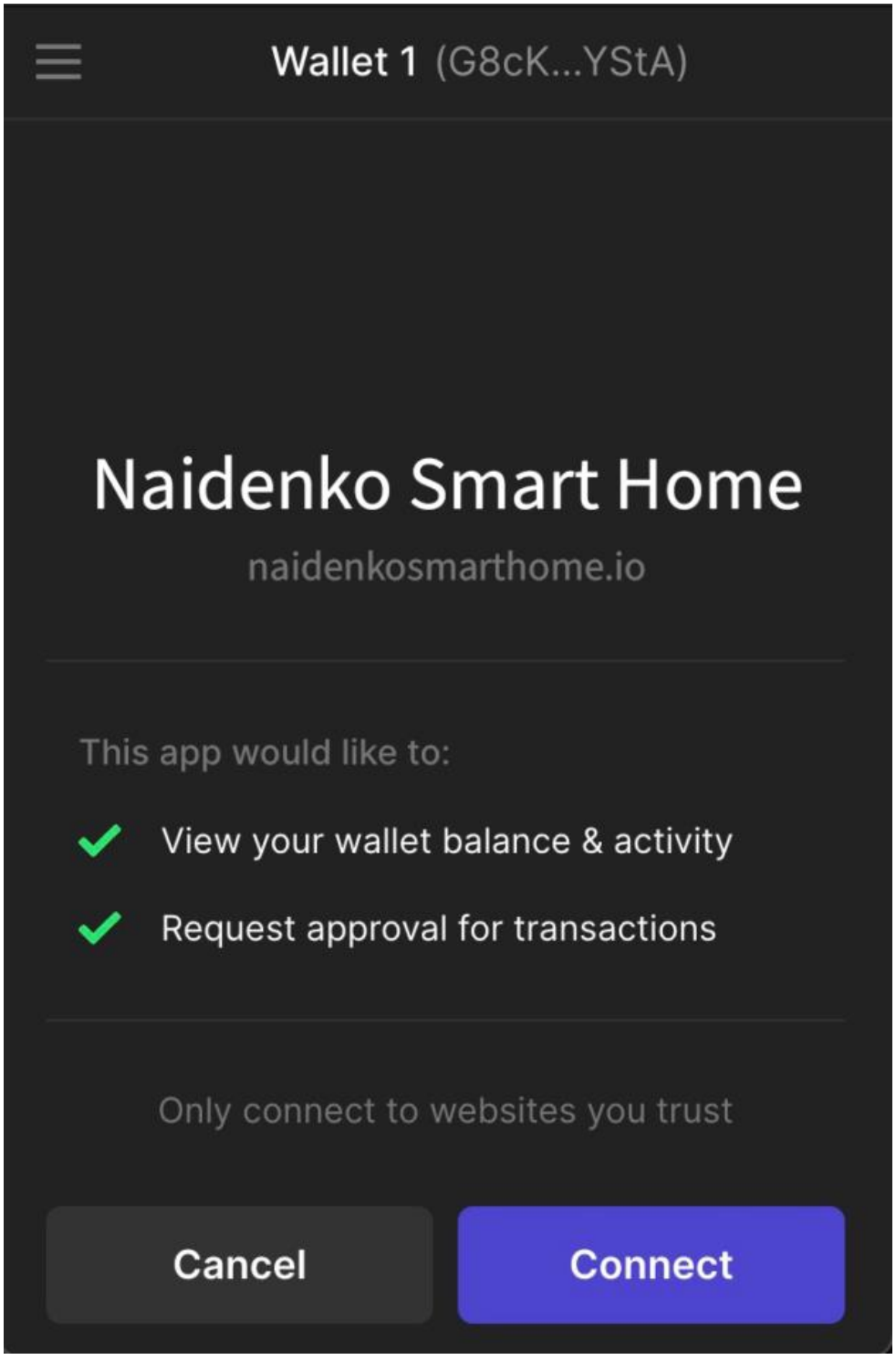

Рисунок 3. 4 – Інтерфейс Phantom Wallet

Натискаючи кнопку Connect, користувач авторизується в системі та вважається аутинтифікованим у тому випадку, якщо адреса користувача співпадає з адресою адміністратора (тобто першого авторизованого користувача у конкретній системі) або з адресою, якій адміністратор надав права на доступ.

Для додавання нових пристроїв, користувачів, змін у всіх критичних вузлах інфраструктури додатку необхідне додаткове підтвердження, що відбувається за допомогою транзакції у блокчейні Solana.

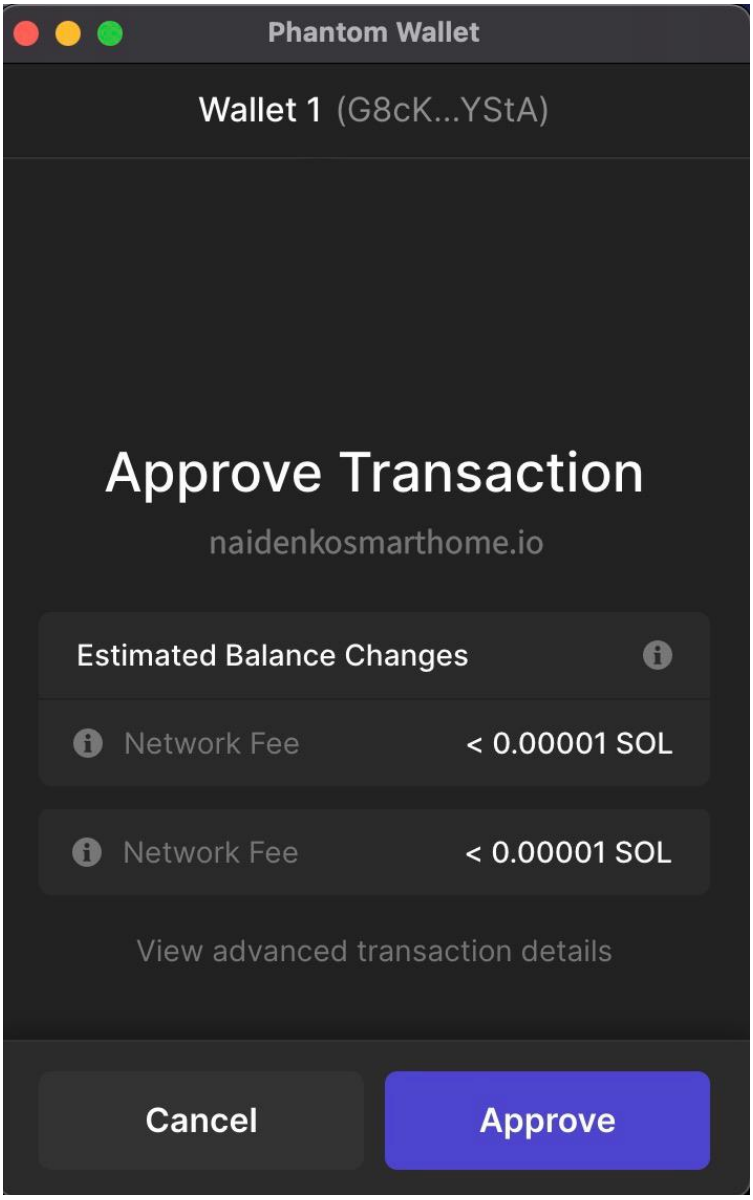

Рисунок 3. 5 – Підтвердження транзакції

HTML файл додатку відповідає за клієнтську частину, а JS файл за серверну. Повний код програми знаходиться у додатку А.

Висновки за розділом 3

Виходячи з наведеного в розділі 3 можна стверджувати, що захищенність додатку підвищилась за рахунок впровадження блокчейн технологій, а саме авторизації та аутинтифікації за допомогою Phantom Wallet.

ВИСНОВКИ

Основною метою даної роботи є вивчення та аналіз можливостей підвищення ефективності безпеки інформаційної системи, якою можна керувати віддалено.

У ході цієї роботи формулюються основні вимоги до системи, а також засоби реалізації та впровадження.

Порівнюючи поточні можливості безпеки систем управління розумним будинком, можна зробити висновок, що найбільш перспективними є децентралізовані системи на основі блокчейна, які не вразливі до атак грубої сили та DDOS.

Після ряду відповідних тестів та численних перевірок можна зробити висновок, що запропонована система з віддаленим доступом працює досить добре.

У ході цієї роботи було отримано такі результати:

Після ретельного аналізу сучасних систем захисту програм був обраний інструмент у вигляді програмного забезпечення, розробленого автором даної дисертації.

Було розроблено архітектуру проекту, включаючи апаратні та програмні компоненти.

Розроблено програмне та апаратне забезпечення, яке може бути використане для підвищення безпеки системи "розумного будинку", в якій реалізовано вибрану архітектуру. Тому можна сказати, що система стала цілком безпечною.

Алгоритми, розроблені в процесі роботи системи управління та моніторингу, здатні не лише захищати, а й ефективно керувати системою безпеки інтелектуальних програм у режимі реального часу.

За результатами випробувань зроблено висновок, що розроблена система відповідає технічним вимогам захисту додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. С. Грингард. Интернет вещей. Будущее уже здесь — 2012. — Т. 8. — №— С. 12-15.

2. Перри Ли. Архитектура интернета вещей, 2018 – 538 с. — 2003. — №5 — с. 90-96.

3. Т. О. Кричевська. Технологія розподіленого реєстру: теоретико-інституційні засади, потенціал, фактичні досягнення та соціально-економічне значення, 2018 – 12 с.

4. IoT Inc: How Your Company Can Use the Internet of Things to Win in the Outcome Economy 2016 – С. 20–21.

5. В. Архіпов «Системи для «інтеллектуального» будинку» - "СтройМаркет", № 45 1999 г.

6. Mike Riley «Programming Your Home Automate and Your Computer» - « The Pragmatic Bookshelf Dallas, Texas • Raleigh, North Carolina », 2012 г.

7. Сравнительный обзор средств предотвращения утечек данных (DLP) [Електронний ресурс]. – Режим доступу: https://safesurf.com/specialists/article/5233/609990/ – Заголовок з екрана. 10.

8. Внедрение DLP-систем [Електронний ресурс]. – Режим доступу: https://techexpert.ua/our-services/implementation-of-dlp-systems/ – Заголовок з екрана. 11.

9. D. K. Barman, G. Khataniar, (2012) “Design Of Intrusion Detection System Based On Artificial Neural Network And Application Of Rough Set”, International Journal of Computer Science and Communication Networks, Vol. 2, No. 4, pp. 548-552.

10. John P. M. Data Breaches Affected Nearly 6 Billion Accounts in 2021 [Електронний ресурс], Mello Jr John P. Tech News World. – 2022. – Режим доступу до ресурсу: https://www.technewsworld.com/story/data-breaches-affected-nearly-6-billion-accounts-in-2021-87392.html.

11. E. Tyugu, (2021) “Artificial intelligence in cyber defense”, 3rd International Conference on Cyber Conflict (ICCC 2011), pp. 1–11

12. Jibilian I. SolarWinds cyber attack [Електронний ресурс], I. Jibilian, K. Canales Business Insider. – 2020. – Режим доступу до ресурсу: https://www.businessinsider.com/solarwinds-hack-explained-government-agencies-cyber-security-2020-12.

13. Сайт компанії Gartner [Електронний ресурс]. – Режим доступу: https://www.gartner.com/. 15.

14. DLP-системы: защита от утечки информации [Електронний ресурс]. – Режим доступу: http://pro-spo.com/personal-data-security/3738-dlp-sistemy-zashhitaot-utechki-informaczii – Заголовок з екрана. 17.

15. Романюков М.Г. Категоріювання інформації у сучасній структурі кібербезпеки держави з використанням матриць цінностей, М.Г. Романюков. – Харків: Критичні комп'ютерні технології та системи: науково-технічний семінар. 23 травня 2019 року. Тема семінару – Безсерверні архітектури, хмарні технології та кібербезпека. 18.

16. Johansen G. Digital forensics and incident response: an intelligent way to respond to attacks. – 2020.

17. Enterprise Data Loss Prevention (DLP) Reviews and Ratings [Електронний ресурс], Gartner Peer Inside. – 2021. – Режим доступу до ресурсу: https://www.gartner.com/reviews/market/enterprise-data-loss-prevention.

18. Ушаков В. Проблеми оперативного виявлення і реагування на інциденти інформаційної безпеки, В. Ушаков, О. Сєвєрінов, GLOBAL CYBER 69 SECURITY FORUM. Матеріали першого міжнародного науково-практичного форуму – Х.: ХНУРЕ, 2020. – С. 104-105.

19. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville. (2017). Improved training of wasserstein GANs. In Proc. of the 31st International Conference on Neural Information Processing Systems, (pp.5769-5779).

20. Глибоке занурення в екосистему Solana. https://coinmarketcap.com/alexandria/ru/article/solana

21. Основні етапи тестування мобільних додатків, [Режим доступу], https://itvdn.com/ru/blog/article/mob-test-blog

22. Використання технології блокчейн для захисту даних. Режим доступу: https://cyberleninka.ua/article/n/ispolzovanie-tehnologii-blokcheyn-dlya-zaschity-dannyh

ДОДАТОК А

Код smart-контракту

Файл SolAuth

```
contract SolAuth {
mapping (string => address) authAddr;
mapping (string => address) recoveryAddr;

event Create (string login);
event AuthChange (string login, address from, address to);
event RecoveryChange (string login, address from, address to);
event Drop (string login, address by);

function createAccount (string _login) public {
require (bytes (_login) .length <= 32);
require (bytes (_login) .length> 2);
require (authAddr [_login] == address (0));
authAddr [_login] = msg.sender;
recoveryAddr [_login] = msg.sender;
// emit Create (bytes32ToString (_login));
emit Create (_login);
}

function authAddress (string _login) view public returns (address) {
return authAddr [_login];
}

function setAuthAddress (string _login, address _addr) public {
require (authAddr [_login] == msg.sender || recoveryAddr [_login] == msg.sender);
broadcast AuthChange (_login, authAddr [_login], _addr);
authAddr [_login] = _addr;
```

```
}

function recoveryAddress (string _login) view public returns (address) {
return recoveryAddr [_login];
}

function setRecoveryAddress (string _login, address _addr) public {
require (recoveryAddr [_login] == msg.sender);
emit RecoveryChange (_login, authAddr [_login], _addr);
recoveryAddr [_login] = _addr;
}

function dropAccount (string _login) public {
require (recoveryAddr [_login] == msg.sender);
delete authAddr [_login];
delete recoveryAddr [_login];
emit Drop (_login, msg.sender);
}

// function bytes32ToString (bytes32 data) pure internal returns (string) {
// uint256 len = 0;
// uint256 j;
// byte char;
// for (j = 0; j <32; j ++) {
// char = byte (bytes32 (uint (data) * 2 ** (8 * j)));
// if (char == 0) {break; }
// len ++;
//}
// bytes memory bytesString = new bytes (len);
// for (j = 0; j <len; j ++) {
```

```
// char = byte (bytes32 (uint (data) * 2 ** (8 * j)));
// bytesString [j] = char;
//}
// return string (bytesString);
//}

function signerAddress (bytes32 data, uint8 v, bytes32 r, bytes32 s) pure public returns (address) {
bytes memory prefix = "\ x19Ethereum Signed Message: \ n32";
bytes32 prefixed = keccak256 (prefix, data);
return ecrecover (prefixed, v, r, s);
}
function signerAddressRaw (bytes32 data, uint8 v, bytes32 r, bytes32 s) pure public returns (address) {
return ecrecover (data, v, r, s);
}
}
```

Файл migrations.sol

```
pragma solidity ^ 0.4.8;

contract Migrations {
  address public owner;

  // A function with the signature `last_completed_migration ()`, returning a uint, is required.
  uint public last_completed_migration;

  modify restricted () {
    if (msg.sender == owner) _;
  }
```

```
  constructor () public {
    owner = msg.sender;
  }

  // A function with the signature `setCompleted (uint)` is required.
  function setCompleted (uint completed) restricted public {
    last_completed_migration = completed;
  }

  function upgrade (address new_address) restricted public {
    Migrations upgraded = Migrations (new_address);
    upgraded.setCompleted (last_completed_migration);
  }
}
```

Файл contracts.txt

[{"constant": true, "inputs": [{"name": "_ login", "type": "bytes32"}], "name": "recoveryAddress", "outputs": [{"name": "", "type": "address"}], "payable": false, "stateMutability": "view", "type": "function"}, {"constant": true, "inputs": [{" name ":" _ login "," type ":" bytes32 "}]," name ":" authAddress "," outputs ": [{" name ":" "," type ":" address "}]," payable ": false," stateMutability ":" view "," type ":" function "}, {" constant ": false," inputs ": [{" name ":" _ login "," type ":" bytes32 "} , {"name": "_ addr", "type": "address"}], "name": "setRecoveryAddress", "outputs": [], "payable": false, "stateMutability": "nonpayable", " type ":" function "}, {" constant ": false," inputs ": [{" name ":" _ login "," type ":" bytes32 "}, {" name ":" _ addr "," type " : "address"}], "name": "setAuthAddress", "outputs": [], "payable": false, "stateMutability": "nonpayable", "type": "function"}, {"constant": false, "inputs": [{"name": "_ login", "type": "bytes32"}], "name": "dropAccount", "outputs": [], "payable": false, "stateMutability" : "nonpayable", "type": "function"}, {"constant": false, "inputs":

[{"name": "_ login", "type": "by tes32 "}]," name ":" createAccount "," outputs ": []," payable ": false," stateMutability ":" nonpayable "," type ":" function "}]

Файл config.json

```
{
 "extends": [
  "tslint: recommended",
  "tslint-react"
 ],
 "rules": {
  "use-isnan": true,
  "indent": [
   true,
   "spaces",
   2
  ],
  "arrow-return-shorthand": false,
  "space-before-function-pair": [
   true,
   {
    "anonymous": "always",
    "named": "never",
    "asyncArrow": "always"
   }
  ],
  "no-bitwise": false,
  "ban-types": [
   true,
   [
    "Object",
    "Use 'object' instead."
```

```
    ],
    [
      "Boolean",
      "Use 'boolean' instead."
    ],
    [
      "Number",
      "Use 'number' instead."
    ],
    [
      "String",
      "Use 'string' instead."
    ]
  ],
  "object-literal-key-quotes": [
    false
  ],
  "no-string-literal": false,
  "jsx-no-lambda": false,
  "jsx-no-multiline-js": false,
  "no-shadowed-variable": false,
  "prefer-const": false,
  "comment format": [
    false
  ],
  "no-namespace": false,
  "array-type": [
    true,
    "array"
  ],
```

```
"max-line-length": [
  true,
  180
],
"new-parens": true,
"no-arg": true,
"no-conditional-assignment": false,
"no-consecutive-blank-lines": [
  true,
  2
],
"quotemark": [
  true,
  "single",
  "jsx-double"
],
"no-var-requires": false,
"trailing-comma": [
  true,
  {
    "multiline": "always",
    "singleline": "never"
  }
],
"unified-signatures": false,
"prefer-for-of": false,
"curly": false,
"max-classes-per-file": [
  false
],
```

```
"object-literal-sort-keys": false,
"no-empty-interface": false,
"ordered-imports": [
  false
],
"only-arrow-functions": [
  false
],
"no-console": [
  false
],
"callable-types": false,
"member-ordering": [
  true,
  {
    "order": [
      "private-static-field",
      "protected-static-field",
      "public-static-field",
      "private-static-method",
      "protected-static-method",
      "public-static-method",
      "private-instance-field",
      "protected-instance-field",
      "public-instance-field",
      "constructor",
      "private-instance-method",
      "protected-instance-method",
      "public-instance-method"
    ]
```

```
      }
    ]
  },
  "jsRules": {
    "max-line-length": [
      true,
      180
    ]
  }
}
```