Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

Безпека інформаційних систем і технологій

«Допущено до захисту»

Зав.кафедрою БІСТ

Сватовський І.І._____

« » червня 2023р.

Пояснювальна записка

до кваліфікаційної роботи бакалавра

спеціальність: 125 Кібербезпека

на тему: «Розробка web-додатку для віддаленої роботи з квантовим генератором випадкових послідовностей»

оцінка « »

Голова ЕК

Лемешко О.В._____

Керівник доц., к.т.н Нарєжній О.П.
(прізвище та ініціали/підпис)

Рецензент ст. викл Осипчук А.В.
(прізвище та ініціали/підпис)

Виконавець студент групи КБ-42

Сорокін К.Ю._____
(прізвище та ініціали/підпис)

Харків – 2023

РЕФЕРАТ

Пояснювальна записка складається з 48 сторінок, 28 малюнків, та 15 джерел за переліком посилань.

Мета роботи: розробити систему яка складається з налаштованої на роботу операційної системи з сервером SSH та встановлених пакетів, web-додатка, який зможе зчитати з порту згенеровані випадкові числа на запити з різних протоколів, та програми, яка симулює генератор випадкових чисел. Система повинна бути розроблена за допомогою перевірених технологій, вибір яких повинен буду обґрунтований. Система повинна бути захищена від атак зсередини локальної мережі та мережі Інтернет. Сценарій атаки перехвату згенерованих випадкових чисел та сценарій атаки обмеження доступу повинні бути перевірені. Також система повинна відповідати функціональним схемам, вибір яких також повинен бути обґрунтовано. Після розробки систему необхідно перевірити на безпеку та перевірити результати роботи під навантаженням.

Результат роботи: система, яка може бути використана для віддаленої роботи з квантовим генератором випадкових чисел з якою можливо працювати за допомогою протокол HTTP, функціональні схеми за якими розроблялася система, результати тестування, за якими були виявлені можливі недоліки та виправлені вразливості безпеки системи та пари ключів з сертифікатом. Результати роботи можуть застосовуватись в майбутніх дослідженнях в ґалузі роботи з квантовими генераторами випадкових чисел та оптимізації роботи системи віддаленої роботи під навантаженням.

Ключові слова: РОЗРОБКА, WEB, PYTHON, HTTP, ШИФРУВАННЯ, ТЕСТУВАННЯ, RASPBERRY, ГЕНЕРАТОР ВИПАДКОВИХ ЧИСЕЛ, ВІДДАЛЕНА РОБОТА.

ABSTRACT

The explanatory memorandum consists of 48 pages, 28 figures, and 15 sources from the list of references.

Objective: to develop a system that consists of: an operating system prepared for work with an SSH-server and installed packages, web application, that could read generated random numbers from a serial port on requests from different protocols, and an application, that simulates quantum random number generator. The system must be developed using approved technologies, choice of which must be justified. The system must be secured from attacks inside local network and from the Internet. A scenario of intercepting generated random numbers and a scenario of denial-of-service attacks must be tested. Also, the system must be developed according to functional schemes, the choice of which must be justified. After development the system must be tested for security and performance.

Result: A system, that can be used to work remotely with quantum random number generator using HTTP protocol, functional schemes that were used to develop the system, tests results that were used to detect probable issues and security vulnerabilities, that were fixed, pairs of asymmetric keys with a certificate. Result can be used in a future research in a field of quantum random number generators and performance optimisation of the system for remote work under stress.

Keywords: DEVELOPMENT, WEB, PYTHON, HTTP, ENCRYPTION, TESTING, RASPBERRY, RANDOM NUMBER GENERATOR, REMOTE WORK.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ, ПОЗНАЧЕНЬ, ТЕРМІНОВ

SSL	–	Secure Socket Layer
TLS	–	Transport Layer Security
TCP/IP	–	Transmission Control Protocol (TCP) і Internet Protocol (IP)
RFC	–	Request for Comments (Запит коментарів)
MAC	–	Message Authentication Code
CA	–	Certificate Authority
HTTP	–	Hyper Text Transfer Protocol (протокол передачі гіпер-текстових документів)
HMAC	–	Hash-based message authentication code (хеш-код аутентифікації повідомлень)
SHA	–	Secure Hash Algorithm
RSA	–	абревіатура від прізвищ Rivest, Shamir та Adleman (криптографічний алгоритм з відкритим ключем, що базується на обчислювальній складності задачі факторизації великих цілих чисел)
USB	–	Universal Serial Bus
HDMI	–	High-Definition Multimedia Interface
Micro-SD	–	формат картки пам’яті
IoT	–	Internet of Things
JSON	–	JavaScript Object Notation
UI	–	User Interface
API	–	Application Programming Interface
SSH	–	Secure Shell
DNS	–	Domain Name Server
Nmap	–	Network Mapper
ICMP	–	IP Control Message Protocol

RNG	–	Random Number Generator
TRNG	–	True RNG
QRNG	–	Quantum RNG
CSRNG	–	Cryptographically Secure RNG
PRNG	–	Pseudo-random Number Generator

ВСТУП

У сучасному цифровому світі, де технології займають центральне місце у багатьох сферах життя, безпека і надійність стають критичними аспектами для розробки та функціонування різноманітних систем. Одним з основних викликів у цьому контексті є забезпечення випадковості - процесу генерації непередбачуваних послідовностей чисел. Класичні методи генерації псевдовипадкових чисел, які базуються на детерміністичних алгоритмах, можуть бути вразливими до атак та передбачення, особливо у контексті криптографічних застосувань та систем, де надійність випадковості є вирішальною. Однак, з появою квантових генераторів випадкових чисел виникає нова перспектива. Квантові генератори використовують непередбачувані квантові події, такі як фотони, для створення повністю випадкових послідовностей чисел. Ці генератори використовують принципи квантової механіки, що гарантує непередбачуваність та незалежність випадкових значень. У зв'язку з цим, розробка web-додатку для віддаленої роботи з квантовим генератором випадкових послідовностей стає актуальною задачею. Такий додаток може забезпечити доступ до надійної випадковості для широкого спектру застосувань, включаючи криптографію, наукові дослідження, симуляції, аналіз даних та інші області, де випадковість відіграє важливу роль. Оскільки доступу до квантового генератора під час виконання роботи не має, то у даній роботі ми досліджуємо можливості розробки web-додатку, замінивши квантовий генератор на пристрій, який також буде генерувати випадкові числа але з тим же інтерфейсом. Ми розглянемо архітектуру та технології, які дозволяють забезпечити безпеку та ефективність передачі випадкових чисел через мережу Інтернет. Розробка такого web-додатку має потенціал прискорити і спростити доступ до надійної випадковості, забезпечуючи високу безпеку та незалежність випадкових

послідовностей. Це може стати кроком у напрямку розвитку більш безпечних та надійних систем у сучасному цифровому світі.

1 РОЗРОБКА ФУНКЦІОНАЛЬНИХ СХЕМ ДЛЯ РОЗУМІННЯ ТОГО ЯК ПОВИННА ФУНКЦІОНУВАТИ СИСТЕМА

1.1 Обґрунтування використаних діаграм

Оскільки для розробки системи потрібно два прилади та два компоненти, можемо використати діаграму розгортання як схему фізичних приладів та компонентів. Компоненти та прилади не мають профілів та не зберігають даних, тому можемо використати одну діаграми розгортання для опису структури системи.

З точки зору поведінки оскільки користувач повинен мати одну роль, діаграма випадків використання буде такою самою як і діаграма послідовності. Система має один профіль та один випадок використання, тому розробка діаграми послідовності розкриє алгоритм роботи системи.

1.2 Діаграма розгортання

Діаграма розгортання є одним з видів діаграм, який використовується в UML, який дозволяє моделювати фізичну структуру системи. Діаграма розгортання дозволяє відображати фізичні складові системи, такі як обладнання та програмні компоненти, та відносини між ними. Компоненти системи зображуються у вигляді прямокутників з назвами, а обладнання - у вигляді прямокутників або кола. Зв'язки між компонентами та обладнанням позначаються лініями з напрямленими стрілками. Діаграма розгортання може використовуватися для планування та опису фізичної архітектури системи. Вона дозволяє показати, які компоненти програмного забезпечення розташовані на обладнанні та як вони взаємодіють між собою та з зовнішніми системами. Діаграма розгортання також може бути використана для опису характеристик обладнання, таких як його розмір, тип, характеристики мережі тощо. Отже, діаграма розгортання є корисним інструментом для архітекторів та розробників програмного забезпечення, який допомагає розуміти фізичну

структуру системи та взаємозв'язки між її компонентами. Система складається з трьох компонентів з точки зору фізичних приладів – Raspberry PI 4, Ноутбук та PL2303TA. Linux використовується як ОС ноутбука. Модель та інші характеристики ноутбука для генерації випадкових чисел не важливі(рис 1.1).

Рисунок 1.1 – Діаграма розгортання

PL2303TA - це USB-адаптер для передачі даних, який дозволяє підключати пристрої з RS-232 портами до комп'ютера через USB-порт. Ці провідники можуть мати різний колір та позначення в залежності від виробника адаптера. Правильне підключення провідників до PL2303TA дозволяє передавати дані між RS-232 пристроями та комп'ютером через USB-порт.

Raspberry Pi 4 Model B - це одно платний комп'ютер, розроблений фондом Raspberry Pi Foundation. Raspberry Pi 4 Model B 1.4 - це оновлена версія попередньої моделі, яка включає в себе наступні характеристики:

- Чіп Broadcom BCM2711 з 4-ядерним процесором Cortex-A72 на швидкості 1,5 ГГц
- Оперативна пам'ять ємністю 2 ГБ
- Підтримка бездротового зв'язку Wi-Fi 802.11ac та Bluetooth 5.0
- Два порти HDMI з підтримкою відеовиходу 4K при 60 кадрах в секунду
- Два порти USB 3.0 та 2 порти USB 2.0
- Порт Gigabit Ethernet для підключення до мережі

- Підтримка мікро-SD карт пам'яті для зберігання операційної системи та інших даних

Raspberry Pi 4 Model B rev 1.4 підтримує різноманітні операційні системи, такі як Linux, FreeBSD та Windows. Його можна використовувати для виконання різноманітних завдань, таких як розробка програмного забезпечення, виконання різноманітних проектів з IoT.

1.3 Діаграма послідовності

Діаграма послідовності — це діаграма, яка відображає послідовність виконання об'єктів в рамках якоїсь системи або взаємодії між об'єктами. Вона використовується для опису взаємодії між об'єктами з точки зору послідовності виконання повідомлень. Діаграма послідовності часто використовується для моделювання складних взаємодій між об'єктами та для з'ясування послідовності виконання дій у системі. На діаграмі послідовності зображуються об'єкти, між якими відбувається взаємодія, та повідомлення, які вони обмінюються. Об'єкти зображаються вертикальними лініями, а повідомлення - горизонтальними стрілками, які вказують напрямок передачі повідомлення. На діаграмі можуть бути зображені різні типи повідомлень, такі як сигнали, запити, відповіді та інші. Діаграма послідовності дозволяє відображати часову послідовність виконання повідомлень, вказуючи, який об'єкт їх відправляє і до якого об'єкту вони призначені. Вона також може відображати послідовність виконання операцій у межах окремого об'єкту. Діаграма послідовності є потужним інструментом для аналізу взаємодії між об'єктами та моделювання складних систем. Вона дозволяє відображати не тільки послідовність виконання дій, але і взаємозв'язки між об'єктами та залежності між ними. Якщо проаналізувати процес генерації випадкових чисел, то він поділяється на чотири кроки(рис. 1.2):

Рисунок 1.2 – Діаграма послідовності веб-додатка

1) Запит користувача до веб-додатку за допомогою браузера або іншого клієнта
2) Зчитування байтів з порту веб-додатком, очікування відповіді від порту.
3) Отримання байтів з порту та обробка їх згідно з параметрами
4) Відповідь на запит користувача у форматі протоколу.

2 ВИБІР ТЕХНІЧНОЇ БАЗИ

2.1 Обґрунтування вибору технологій

У виборі мови програмування для проекту, що працює з серійним портом та веб-розробкою, варто враховувати ряд факторів. У даному випадку, кожна з мов Java, C++ або Python може бути використана для реалізації проекту потрібно розглянути бібліотеки та фреймворки для роботи з криптографії, роботи з серійними портами та веб-додатками.

Якщо подивитися бібліотеки для роботи з серійним портом в Java, C++ та Python, то можна виділити наступні бібліотеки. У Java використовуються такі бібліотеки як RXTX, яка не підтримується з 2013, та jSerialComm. У C++ є наступні бібліотеки Boost.Asio та Qt Serial Port. У Python для роботи з серійними портами доступні ці пакети PySerial, Serial. Оскільки бібліотеки усіх трьох мов виконують свою функцію, тому усі мови мають можливість роботи з серійним портом. Але деякі бібліотеки на Java вже не підтримуються. Всі три мови мають велику кількість бібліотек для розробки веб-додатків. Для прискорення роботи ми використаємо «фреймворк» з вбудованим веб-сервером.

У програмуванні, фреймворк - це набір загальних компонентів, бібліотек, правил та протоколів, які дозволяють розробникам зосередитися на розв'язанні конкретних завдань, а не на створенні загальних рішень для базових компонентів. Фреймворк забезпечує основну структуру програми, організовує роботу з даними, відображенням, взаємодією з користувачем, мережевою взаємодією та іншими аспектами розробки програмного забезпечення. Фреймворки зазвичай мають визначену структуру та організацію коду, що дозволяє розробникам швидко та ефективно створювати високоякісне програмне забезпечення з меншими затратами часу та зусиль. Вони також надають засоби для тестування та налагодження програми, що робить процес

розробки менш складним та більш ефективним. Зазвичай фреймворки надають декілька варіантів використання, залежно від потреб та рівня знань розробника. Деякі фреймворки можуть бути спрощені та легкі, щоб допомогти розробникам з нульовим рівнем знань, тоді як інші можуть бути складнішими та потужнішими, для досвідчених розробників, які потребують більшого контролю та настроювання програми. У загальному розумінні, фреймворк є інструментом, який дозволяє розробникам зосередитися на створенні програмного забезпечення, замість витрачання часу на написання базових компонентів програми.

Якщо брати три найпопулярніших веб-фреймворків на Python, відсортованих за рейтингом на даний момент, це:

1) Django - це повнофункціональний веб-фреймворк на Python, який надає все, що потрібно для швидкої розробки веб-додатків. Django має велику кількість готових модулів та інструментів, що дозволяє швидко розробляти веб-додатки.
2) Flask - це легкий та простий веб-фреймворк для Python, який надає базовий функціонал для розробки веб-додатків. Flask є простішим та легшим у використанні фреймворком, що дозволяє розробникам більш точно налаштувати свій додаток.
3) FastAPI - це швидкий та молодий веб-фреймворк на Python, який надає високу продуктивність та ефективність для розробки веб-додатків з API. FastAPI базується на стандартах OpenAPI та JSON Schema.

Нижче перелічені три найпопулярніших веб-фреймворків на Java, відсортовані за рейтингом на даний момент:

1) Spring Framework - це веб-фреймворк на Java, який надає широкий набір функцій для розробки веб-додатків. Spring Framework базується на інверсії керування та введенні залежностей та пропонує механізми для підключення до баз даних.

2) JavaServer Faces - це компонентний веб-фреймворк на Java, який дозволяє розробникам створювати веб-додатки з використанням готових компонентів, таких як таблиці, форми.
3) Play Framework - це веб-фреймворк на Java. Він базується на моделі акторів та має асинхронну архітектуру, що дозволяє працювати з багатьма одночасними з'єднаннями та запитами.

На C++ не так широко використовується для веб-розробки, але тим не менш, є деякі фреймворки для розробки веб-додатків на C++. Можна використати Qt або Wt.

Як можна бачити, Java та Python мають дуже великий вибір різноманітних фреймворків, тоді як C++ має лише фреймворки для розробки графічного інтерфейсу які можуть робити запроси. Тому, оскільки Python має величезний вибір фреймворків та його бібліотеки для роботи з серійним портом досі оновлюються, виберемо саме цю мову для розробки додатку. Для реалізації додатку нам потрібно розробити код обробки лише для одного сценарію. Також система не передбачає бази даних та передбачає одну роль, тому оберемо між фреймворками, які використовують найменшу кількість залежностей. Як результат не будемо використовувати Django. Вибір залишається між Flask та FastAPI:

1) Швидкість: FastAPI був розроблений, щоб мати високу продуктивність та швидкість. FastAPI використовує ASGI-сервер, який дозволяє роботі з багатьма одночасними з'єднаннями та запитами, що робить його швидшим за Flask.
2) Структура проекту: FastAPI має вбудовану підтримку структуризації проекту на основі стандарту OpenAPI, що дозволяє розробникам зосередитися на функціональності додатку, а не на структурі проекту. У Flask, розробник повинен самостійно структурувати свій проект.
3) Схема перевірки даних: FastAPI має вбудовану підтримку схеми перевірки даних заснованої на JSON Schema, що дозволяє розробникам

перевіряти вхідні дані та виводити документацію безпосередньо з коду. У Flask, розробник повинен використовувати сторонні бібліотеки для перевірки вхідних даних.

4) Документація: FastAPI має вбудовану підтримку автоматичної генерації документації на основі стандарту OpenAPI та Swagger UI. Це робить процес документування додатку швидшим та простішим. У Flask, розробник повинен самостійно документувати свій додаток.
5) Розширюваність: Flask має більшу кількість сторонніх бібліотек та розширень, що дозволяє розробникам налаштувати свій додаток згідно з власними потребами та вимогами.

Оскільки розширення нам не потрібні, оберемо FastAPI як фреймворк розробки веб-додатку. Тепер порівняємо pyserial та serial:

1) Підтримка ОС: pyserial підтримує більше ОС, включаючи Windows, macOS та Linux. serial, з іншого боку, підтримує лише Windows та Linux.
2) API: pyserial має більше функцій та можливостей, що дозволяє розробникам взаємодіяти зі з'єднанням серійного порту на більш високому рівні. serial, з іншого боку, має меншу кількість функцій та є більш простим у використанні.
3) Підтримка різних протоколів: pyserial підтримує більше протоколів взаємодії зі з'єднанням серійного порту, таких як RS-485 та Modbus. serial, з іншого боку, підтримує лише базовий протокол RS-232.
4) Швидкість: Обидві бібліотеки можуть досягати однакової швидкості передачі даних через з'єднання серійного порту, але pyserial дозволяє налаштовувати різні параметри передачі даних, такі як швидкість передачі, розмір буфера та інше.

Оберемо бібліотеку pyserial для роботи з серійним портом.

2.2 Обґрунтування вибору ОС приладу

Образ Ubuntu 20.10 можна завантажити з офіційного сайту Ubuntu. Для Raspberry Pi доступні три версії Ubuntu 20.10: Desktop, Core та Server. Також потрібно обрати між x86 та x64 версіями ОС:

- Desktop версія є повнофункціональною операційною системою з графічним інтерфейсом користувача, що містить різноманітні програми та інструменти для повсякденного використання. Це може бути найкращий вибір, якщо використовувати Raspberry Pi як персональний комп'ютер.
- Server версія призначена для використання в якості сервера. Це операційна система без графічного інтерфейсу користувача, яка включає в себе лише необхідні компоненти для роботи.
- Core - це спеціально розроблена операційна система для вбудованих пристроїв та IoT, яка має дуже обмежену кількість пакетів та програм, що містяться в її базовій інсталяції.

Для виконання мети потрібно використати Server версію, бо нам не потрібна графічна середа та інструменти для повсякденного використання та недостатньо функціонала з Core версії. Оскільки прилад має менш ніж 4 ГБ оперативної пам’яті, потрібно використати x86 версію ОС

3 РОЗРОБКА WEB-ДОДАТКУ ДЛЯ ВЗАЄМОДІЇ КОМП'ЮТЕРА КОРИСТУВАЧА З КВАНТОВИМ ГЕНЕРАТОРОМ ВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ

3.1 Налаштування Raspberry PI4

Для того, щоб почати розробку веб додатку потрібно підготовити середу Raspberry Pi 4. Потрібно встановити Ubuntu 20.10, забезпечити зв’язок з інтернетом для завантаження пакетів, додати прилад до локальної мережі та встановити необхідні пакети.

3.1.1 Встановлення програмного забезпечення для розпаковки ОС

Для встановлення ОС на карту пам’яті спочатку треба обрати програму для розпакування файлів та додаткового налаштування. Використаємо найпопулярнішу програму, яку рекомендує офіційне тренування – Raspberry Pi Imager. Після встановлення програми треба запустити її та вибрати завантажений образ Ubuntu 20.10 Server x86 та розділи microSD карти в якості пристрою для запису.

3.1.2 Налаштування ОС перед розпаковкою файлів

Оскільки у прилада нема монітора для взаємодії з Raspberry PI, то один з найбільш простих способів працювати з пристроєм це протокол SSH. SSH - це криптографічний мережевий протокол, розроблений для забезпечення безпечного зв'язку, віддаленого виконання команд та інших мережевих послуг через незахищену мережу. Він широко використовується для віддаленого адміністрування, безпечної передачі файлів та інших завдань, які вимагають безпечного зв'язку між клієнтом та сервером. Протокол SSH поділяється на кілька рівней та компонентів:

1) Транспортний рівень (RFC 4253) - транспортний рівень забезпечує конфіденційність, цілісність та автентифікацію сервера. Він встановлює захищений, зашифрований канал між клієнтом та сервером за

допомогою комбінації асиметричних (з відкритим ключем) та симетричних алгоритмів шифрування. Клієнт і сервер під час встановлення з'єднання домовляються про алгоритми шифрування, обміну ключами та захисту цілісності. Транспортний шар також включає механізм для повторного обміну ключами, який дозволяє періодично оновлювати ключі сеансу.

2) Рівень автентифікації (RFC 4252) - цей рівень відповідає за автентифікацію клієнта на сервері. Протокол автентифікації SSH підтримує кілька методів, таких як пароль, автентифікація за допомогою відкритого ключа та автентифікація на основі хоста. Клієнт і сервер також можуть використовувати додаткові або спеціальні методи автентифікації. Процес автентифікації зазвичай включає представлення клієнтом облікових даних, таких як ім'я користувача та пароль або відкритий ключ, які сервер перевіряє.
3) Рівень з'єднання (RFC 4254) - рівень з'єднання передає кілька логічних каналів через безпечний транспортний рівень. Кожен канал може використовуватися для окремої послуги, такої як сеанс оболонки, передача файлів або переадресація портів. Канали можуть відкриватися та закриватися незалежно один від одного, дозволяючи одночасно використовувати кілька послуг через одне з'єднання SSH.

Протокол SSH також має декілька додаткових компонентів та функцій:

1) Призначені числа та простори імен (RFC 4250): Цей документ визначає схеми нумерації та іменування, що використовуються в протоколі SSH, такі як номери повідомлень, назви алгоритмів та назви служб.
2) Безпечна публікація відбитків ключів SSH на основі DNS (RFC 4255): Цей RFC описує метод зберігання та отримання відбитків відкритих ключів SSH в системі доменних імен (DNS) безпечно, використовуючи розширення безпеки DNS (DNSSEC) для забезпечення автентичності та цілісності.

SSH широко прийнятий і вважається безпечним та надійним протоколом для віддаленого зв'язку. Його модульний дизайн, підтримка кількох методів автентифікації та можливість передавати кілька послуг роблять його універсальним та надійним рішенням для різноманітних потреб у безпечному зв'язку. SSH широко використовується у різних галузяї і додатках, забезпечуючи конфіденційність, цілісність та автентифікацію при взаємодії між клієнтами та серверами. Його гнучкість та безпека забезпечують популярність протоколу SSH як стандарту для віддаленого керування та передачі даних через незахищені мережі.

У якості реалізація SSH можна обрати вже встановлене у образ системи програмне забезпечення OpenSSH. Він складається з двох компонентів: серверного та клієнтського. Серверний компонент дозволяє віддаленим користувачам підключатися до системи та керувати нею, тоді як клієнтський компонент дозволяє вам підключатися до віддалених систем та виконувати команди на них. OpenSSH використовує криптографічні методи для шифрування трафіку та забезпечення захисту від зламу. Це забезпечує безпечний та захищений спосіб підключення до віддалених систем, особливо якщо вони знаходяться в мережі Інтернет.

На даному етапі використаємо аутентифікацію з паролем (рис. 3.1).

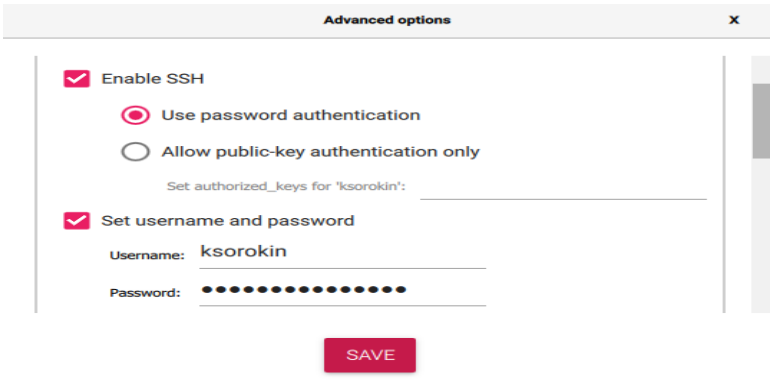

Рисунок 3.1 - Налаштувати сервера OpenSSH за допомогою ПО Raspberry Pi Imager.

Наступним кроком буде налаштування бездротової мережі для пристрою. Під’єднаний до мережі Raspberry PI зможе завантажувати додаткові пакети для роботи веб-додатку (рис. 3.2).

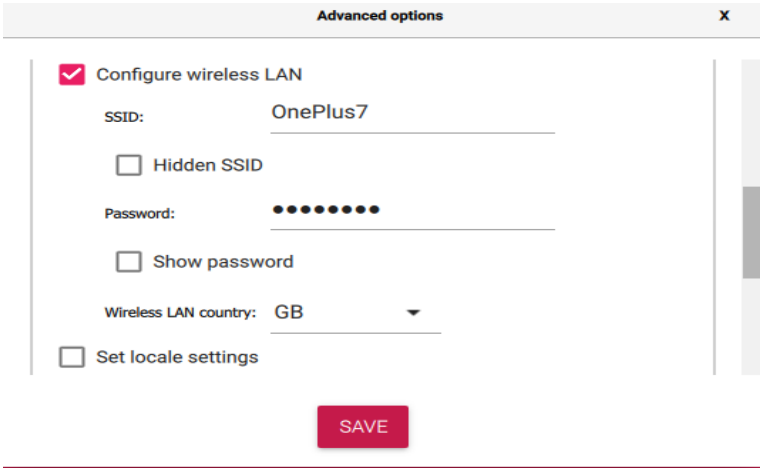

Рисунок 3.2 - Налаштувати параметри бездротової мережі для під’єднання до пристрою за допомогою SSH

Після розпаковки налаштованого на роботу у мережі образу ОС, можна тестувати зв'язок з пристроєм та його спроможність виходу у мережу Інтернет.

3.1.3 Встановлення та тестування зв’язку з пристроєм

Через відсутність маршрутизатора побудуємо локальну мережу за допомогою смартфону з операційною системою Android. Після увімкнення мобільного інтернету, можемо перетворити телефон на маршрутизатор увімкнувши функцію Hotspot, з можливістю виходу у мережу Інтернет. Така конфігурація не дає функції подивитися на LAN IP адресу для кожного пристрою, але ми можемо отримати її просканувавши бездротову мережу за допомогою програмного забезпечення nmap.

Nmap - це відкрите програмне забезпечення, що використовується для сканування мереж та виявлення підключених пристроїв, а також для визначення конфігурації мережі та служб, які доступні на підключених комп'ютерах. Nmap працює за допомогою низки технік сканування, таких як сканування портів, сканування версій програм та опитування мережі.

Найбільш поширеним видом сканування, який використовує Nmap, є сканування портів. Воно використовується для визначення того, які порти на пристрої відкриті та доступні для з'єднання. Nmap також може використовуватися для виявлення вразливостей в мережах, які можуть бути використані зловмисниками для злому систем та викрадення даних. Для цього в Nmap вбудовано більше 150 типів сканування, які можуть використовуватися для виявлення вразливостей та інших проблем в мережі. Nmap є потужним інструментом для адміністраторів мереж, які хочуть досліджувати та вдосконалювати безпеку своїх мереж. Проте, варто використовувати його з обережністю, оскільки некоректне використання може призвести до порушення закону або завдати шкоди мережі. Для сканування мережі потрібно знати маску та її адрес. Для цього ми можемо подивитися конфігурацію мережі у операційній системі Linux.

Для сканування мережі на активні пристрої nmap потребує знати діапазон IP-адрес, які можливо використовувати в мережі. Цей діапазон можна вказати вручну, або ж nmap може скористатися інструментами для визначення активних IP-адресів в мережі.

Для цього можна скористатися предвстановленним пакетом ip. Зі списку інтерфейсів треба знайти інтерфейс з бездротової мережі який містить IPv4 адресу та маску мережі.

Крім того, для сканування мережі на активні пристрої, nmap використовує протоколи ICMP і TCP для відправки запитів до кожного з можливих IP-адрес у визначеному діапазоні та отримання відповіді від активних пристроїв (рис. 3.3).

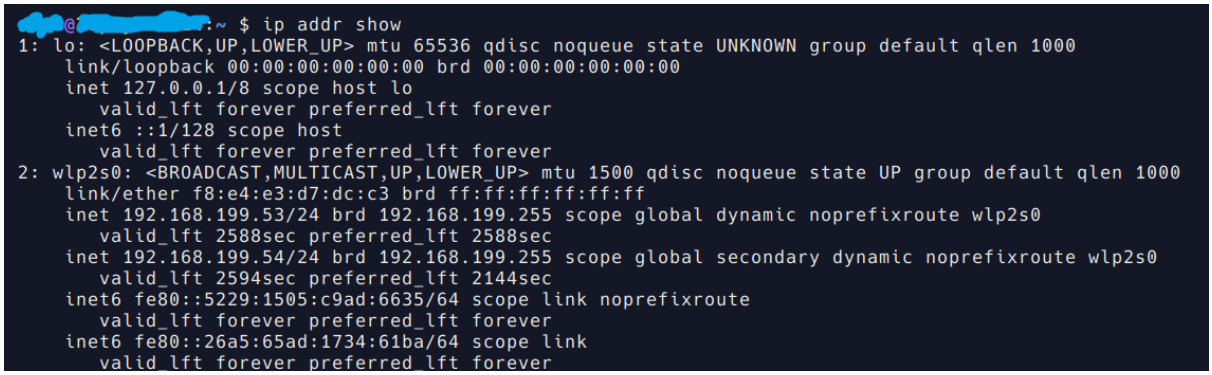

Рисунок 3.3 - Результат виконання команди ip – список інтерфейсів.

Для ефективного сканування мережі, nmap може використовувати різні опції та налаштування, наприклад, змінювати швидкість сканування, використовувати різні типи сканувань, зберігати результати у різних форматах (рис 3.4).

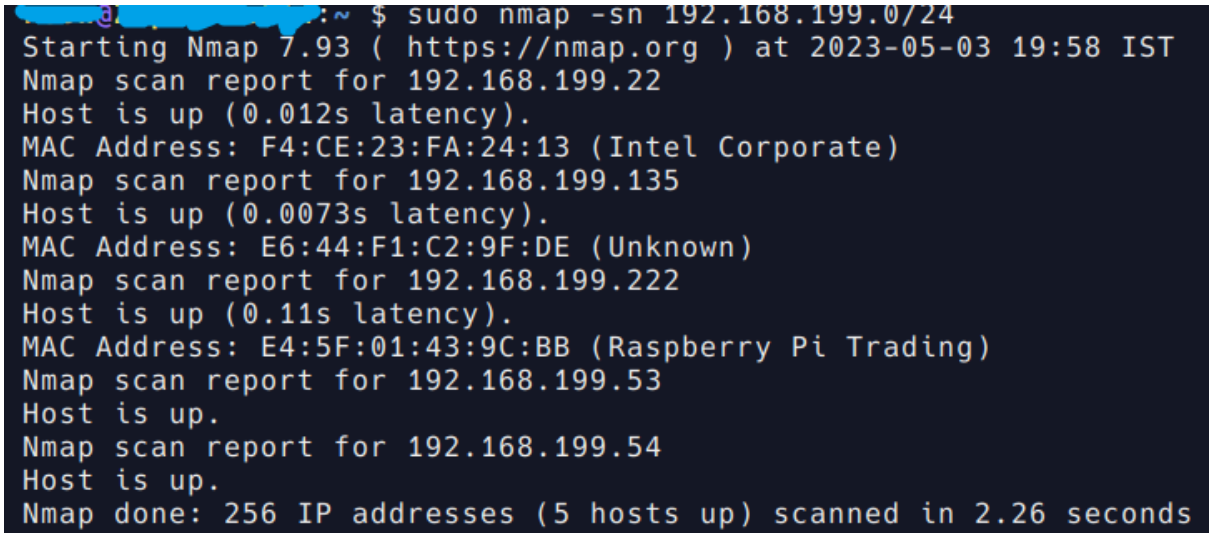

Рисунок 3.4 - Результат виконання команди nmap – список пристроїв приєднаних до междротової локальної мережі.

За допомогою цієї команди отримаємо список пристроїв, які під'єднані до мережі та їх IPv4 адреса. Nmap визначив наш Raspberry PI через його MAC-адресу по її vendor порції. IPv4 адреса пристрою це 192.168.199.222. Розберемо, що означають параметри цієї команди:

- sudo - виконує команду в режимі адміністратора, що дозволяє отримати доступ до додаткових ресурсів, таких як права доступу до мережевих інтерфейсів.

- nmap - викликає утиліту Nmap.
- -sn - параметр, що вказує Nmap виконувати ping-сканування, тобто відправляти запити на кожну IP-адресу в зазначеному діапазоні і чекати на відповідь. Якщо відповідь отримано, то Nmap вважає, що пристрій з цієї IP-адреси активний.
- 192.168.199.0/24 - адреса мережі, яку необхідно сканувати. У цьому випадку, 192.168.199.0 - це мережа, а /24 вказує Nmap на те, що необхідно сканувати всі можливі IP-адреси в мережі, що починаються з «192.168.199.» і закінчуються від 1 до 254.

Отже, команда sudo nmap -sn 192.168.199.0/24 виконує сканування мережі 192.168.199.0/24 на активні пристрої, відправляючи запити на кожну можливу IP-адресу в діапазоні і чекаючи на відповідь. Після завершення сканування, Nmap поверне список активних пристроїв у вказаній мережі.

Тепер перевіримо SSH з'єднання з пристроєм. Для цього потрібно використати клієнтську частину OpenSSH – програму ssh. Оскільки був встановлений вхід по паролю, потрібно лише вказати користувача та пароль. Після цього зможемо користуватися Raspberry PI без дисплея лише увімкнувши точку доступу (рис. 3.5).

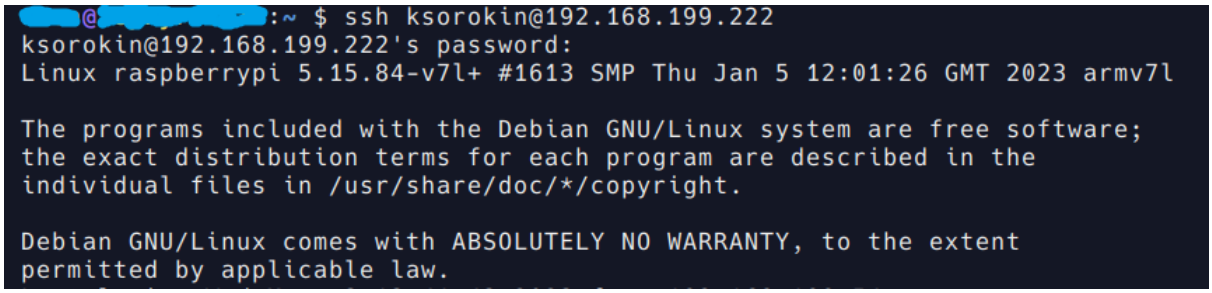

Рисунок. 3.5 - Результат виконання команди ssh – доступ до Raspberry PI під користувачем, якого ми завдали при налаштуванні образа ОС.

3.1.4 Встановлення програмного забезпечення на Raspberry PI та перевірка доступу до мережі Інтернет.

Для початку роботи потрібно обновити систему за допомогою використання команди apt-get:

1) Відкрити сесію SSH з пристроєм.
2) Ввести наступну команду, щоб оновити списки пакетів - sudo apt-get update.
3) Після завершення оновлення списків пакетів ввести наступну команду, щоб оновити всі пакети в системі - sudo apt-get upgrade.
4) Після завершення оновлення треба перезавантажити систему, командою - sudo reboot.

Для встановлення Python 3 на Ubuntu можна використати наступні кроки в терміналі:

1) Відкрити термінал з SSH сесією.
2) Потім виконати команду для встановлення Python 3 та його додаткових пакетів – sudo apt install python3 python3-pip. Ця команда встановить менеджер пакетів Python 3 - pip.
3) Після успішного завершення встановлення, перевіримо версію Python 3 за допомогою команди - python3 –version

Ця команда виведе версію Python 3, яка була встановлена. Тепер Python 3 повинен бути успішно встановлений на системі Ubuntu. Існує кілька способів підготувати середовище операційної системи під розробку веб-додатку на Python, але найбільш популярні і рекомендовані способи включають встановлення Python та управління пакетами за допомогою віртуальних середовищ. Оскільки сам Python та менеджер управління пакетів pip ми вже встановили, то нам залишається тільки розібратися з віртуальним середовищем.

1) Спершу потрібно створити віртуальне середовище для веб-додатку. Віртуальне середовище - це ізольоване середовище Python, яке дозволяє вам встановлювати пакети та їх залежності без впливу на глобальну конфігурацію системи. Для створення віртуального середовища можна використовувати засоби, такі як virtualenv або venv (який є стандартним модулем Python).

2) Далі потрібно активувати віртуальне середовище. Це можна зробити за допомогою команди activate, яку можна знайти в папці Scripts віртуального середовища.
3) Останній крок це встановлення необхідних пакетів для веб-додатку. Найпростішим рішенням є встановлення пакетів за допомогою pip. Потрібно встановити FastAPI та pyserial. Зробити це можна за допомогою команди pip install “fastapi[all]” pyserial

3.1.5 Налагодження зв’язку між пристроєм та ноутбуком

Для забезпечення зв’язку потрібно під’єднати кабель до USB-порту ноутбука та під’єднати чотири контакту до Raspberry PI. Для цього треба з’ясувати схеми під’єднання контактів. PL2303TA має 4 провідника та складається з наступних елементів:

1) Красний контакт: 5V - живлення адаптера.
2) Чорний контакт: заземлення адаптера.
3) Зелений контакт: передача даних від пристрою до комп'ютера.
4) Білий контакт: передача даних від комп'ютера до пристрою.

Тепер, для того щоб під’єднати контакти до пристрою, треба визначити які контакти відповідають за ввід та вивід UART, заземлення та живлення. Зробити це можна використавши команду pinout (рис. 3.6).

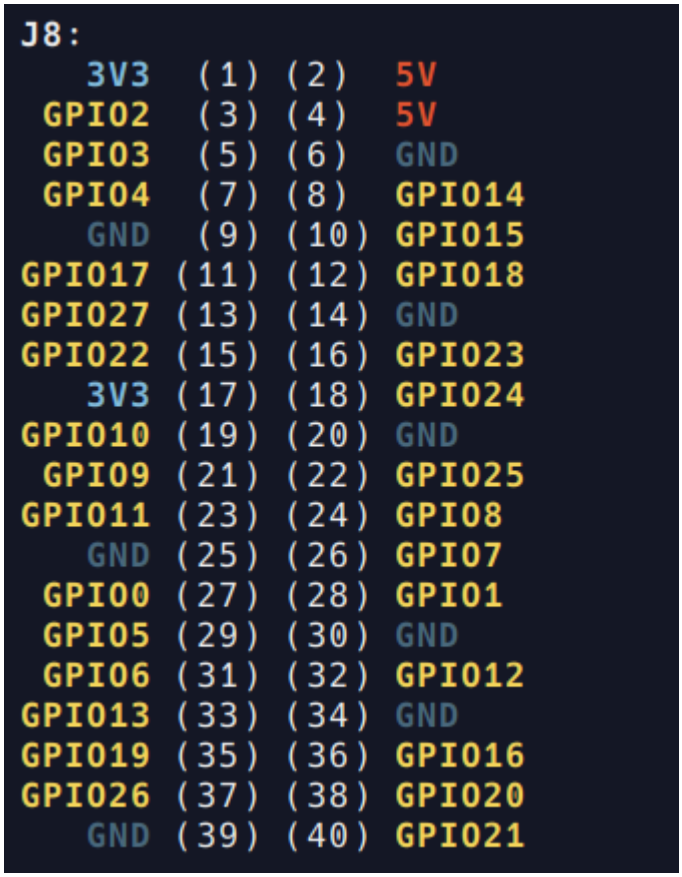

Рисунок 3.6 – Схема плати пристрою

Відповідно цій схемі контакти треба підключити так:

1) Красний контакт (живлення) до четвертого контакту
2) Чорний контакт (заземлення) до шостого контакту
3) Зелений контакт (передача даних від пристрою до комп’ютера) до восьмого контакту
4) Білий контакт (передача даних від комп’ютера до пристрою) до десятого контакт

3.2 Розробка інтерфейсу між генератором випадкових байтів та серійний портом

Для того щоб налаштувати поток випадкових байтів для роботи з серійним портом потрібно:

- Розробити інтерфейс який міг би записати байти до порту
- Вибрати генератор випадкових байтів та обґрунтувати вибір
- Реалізувати генератор випадкових байтів або знайти надійну реалізацію
- Реалізувати інтерфейс

3.2.1 Створення об’єкту потоку порту

Запис байтів до серійного порту виконує пакет pyserial. Для здійснення запису потрібно створити об’єкт класу Serial та надати йому характеристики потоку як параметри (рис. 3.7).

```
ser = serial.Serial(
    port='/dev/ttyUSB0',
    baudrate=9600,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=1
)
```

Рисунок 3.7 – Створення об’єкту

Параметри, які передаються при ініціалізації:

- port: цей параметр вказує на ім'я послідовного порту, до якого підключений пристрій. У даному випадку це `/dev/ttyUSB0`, але залежно від конфігурації пристрою може бути іншим.
- baudrate: цей параметр встановлює швидкість передачі даних в бодах (біти в секнду). У даному випадку швидкість передачі даних встановлюється на рівні 9600 бодів.

- parity: цей параметр встановлює тип контролю парності (parity) для передачі даних. У даному випадку встановлюється значення PARITY_NONE, тобто жодного контролю парності не використовується.
- stopbits: цей параметр встановлює кількість стоп-бітів (stop bits) для передачі даних. У даному випадку встановлюється значення STOPBITS_ONE, тобто використовується один стоп-біт.
- bytesize: цей параметр встановлює розмір даних (data bits), які передаються через послідовний порт. У даному випадку встановлюється значення EIGHTBITS, тобто передаються 8 біт даних.
- timeout: цей параметр встановлює максимальний час очікування (в секундах) на отримання даних від пристрою через послідовний порт. У даному випадку час очікування встановлюється на рівні 1 секунди.

Новостворений об’єкт зможе писати байти до потоку за допомогою метода write.

3.2.2 Вибір та обґрунтування генератора випадкових байтів

Генератор випадкових чисел - це пристрій або програма, які створюють послідовність чисел, які здаються випадковими в певному статистичному сенсі. Генератори випадкових чисел використовуються у багатьох галузях, таких як криптографія, статистика, моделювання та інші.

Існують два основні типи генераторів випадкових чисел: апаратні та програмні. Апаратні генератори випадкових чисел базуються на фізичних процесах, таких як радіоактивний розпад або шум радіоелектронних схем. Вони забезпечують високу ентропію і є найбільш безпечними генераторами випадкових чисел.

Програмні генератори випадкових чисел генерують послідовність чисел на основі початкового числа, використовуючи математичні формули. Вони забезпечують меншу ентропію, але є більш ефективними та менш складними у реалізації. Для безпечного застосування програмних генераторів випадкових

чисел необхідно використовувати високоякісні алгоритми та забезпечувати достатній рівень ентропії.

CSPRNG - це програмний генератор випадкових чисел, який забезпечує високий рівень безпеки шляхом використання криптографічних алгоритмів та засобів захисту. У відміну від звичайних програмних генераторів випадкових чисел, CSPRNG має властивості, що забезпечують, що послідовність випадкових чисел, яку він генерує, не може бути передбаченою чи відтворена незалежно від того, чи знані проміжні значення. CSPRNG використовуються в криптографії, де потрібно забезпечити випадковість при генерації ключів, підписів та інших криптографічних параметрів. Вони також використовуються в інших областях, де важливо забезпечити випадковість, таких як статистика та моделювання.

Виберемо ChaCha20 через його надійність та розповсюдженість реалізацій. ChaCha20 - це симетричний потоковий шифр, розроблений Даніелем Дж. Бернштейном у 2008 році. Він належить до родини потокових шифрів Salsa20, які відомі своєю простотою, швидкістю та безпекою. ChaCha20 є розвитком оригінального шифру Salsa20 з модифікованою перестановочною функцією для покращення дифузійних властивостей. Хоча ChaCha20 не включає безпосередньо концепцій теорії груп, як RSA чи криптографія на еліптичних кривих, його принципи дизайну ґрунтуються на глибокому розумінні сучасної криптографії. Безпека ChaCha20 перш за все залежить від його основної перестановочної функції, яка призначена для забезпечення замішання та розповсюдження. Замішання досягається через нелінійні операції, а розповсюдження - шляхом розподілу впливу вхідних бітів по всьому стану. Основна перестановочна функція ChaCha20 складається з набору простих арифметичних та бітових операцій, таких як додавання за модулем 2^32, бітове XOR та поворот. Опис алгоритму ChaCha20:

1) Генерація ключа: генерується 256-бітний секретний ключ, який безпечно передається між відправником та отримувачем. Обидві

сторони повинні використовувати один і той же ключ для шифрування та розшифрування.

2) Вибір мітки часу: обирається 96-бітна мітка для кожного повідомлення. Мітка часу повинна бути унікальною для кожного повідомлення, зашифрованого з однаковим ключем, щоб забезпечити безпеку.
3) Створення лічильника: 32-бітний лічильник створюється зі значенням нуль і збільшується для кожного наступного блоку ключового потоку.
4) Створення стану: ChaCha20 використовує матрицю 4x4 з 32-бітних слів, яка називається станом. Стан створюється за допомогою чотирьох констант, 256-бітним секретного ключа (розділеним на вісім 32-бітних слів), 32-бітним лічильником та 96-бітної частки (розділеним на три 32-бітних слова).
5) Оновлення стану: стан ChaCha20 оновлюється за допомогою 20 раундів, організованих як 10 "подвійних раундів". Кожен подвійний раунд складається з "колонного раунду" та "діагонального раунду". Раунди включають прості операції, такі як додавання за модулем 2^32, бітове XOR та поворот.
6) Серіалізація стану: після завершення раундів оновлений стан додається елементарно до початкового стану за модулем 2^32. Отримана матриця 4x4 серіалізується, щоб створити 512-бітний блок ключового потоку.
7) Генерація ключового потоку: кроки 4-6 повторюються з значенням лічильника, яке на одиницю більше минулого стану, щоб створити наступні блоки ключового потоку, які потрібні для повідомлення.
8) Шифрування: відкритий текстовий повідомлення об'єднується за допомогою операції XOR зі створеним ключовим потоком, щоб створити зашифрований текст.
9) Розшифрування: отримувач, який має той же секретний ключ, генерує такий же ключовий потік та об'єднує його за допомогою операції XOR з

зашифрованим текстом, щоб відновити відкритий текстовий повідомлення.

Дизайн ChaCha20 з його простими та ефективними операціями робить його високо стійким до різних атак, таких як атаки на основі часу та диференційна криптоаналіз. Більше того, велика кількість раундів (20) забезпечує значний запас безпеки, роблячи шифр стійкішим до потенційних атак.

Важливо підкреслити, що ChaCha20, як симетричний шифр, залежить від безпечного розподілу та керування ключами. На практиці ChaCha20 часто комбінується з алгоритмом аутентифікації повідомлень, таким як Poly1305, щоб забезпечити як конфіденційність, так і цілісність. Комбінована конструкція ChaCha20-Poly1305 використовується у ряді протоколів безпечного зв'язку, включаючи TLS та SSH.

3.2.3 Вибір реалізації

Оскільки генератор випадкових байтів знаходиться на системі Linux, можемо використати потік з /dev/random який використовує саме ChaCha20. У системах Linux і Unix, /dev/urandom є спеціальним файлом, який забезпечує інтерфейс для генерації випадкових чисел. Він використовує різні джерела ентропії на комп'ютері, такі як введення клавіатури, миші, дискової активності та інших процесів, щоб збирати випадкові дані. Коли програма звертається до /dev/urandom, вона отримує блок випадкових даних, які можна використовувати як ключі шифрування, паролі або для генерації випадкових чисел для криптографічних операцій. /dev/urandom запам'ятовує, скільки ентропії було використано при останньому запиті, і використовує це значення, щоб забезпечити високу якість випадкових чисел для майбутніх запитів. /dev/urandom генерує випадкові дані без затримки, використовуючи резервуар випадкових чисел, який постійно оновлюється в процесі збору ентропії. Для цього причини у системах Linux також існує /dev/random, який є дуже потужним інструментом для генерації випадкових чисел, але може бути

повільним у випадку, якщо комп'ютер не має достатньої кількості ентропії. У системі ентропія може з'являтися з різних джерел. Ось декілька прикладів:

- Генератор випадкових подій: система може зібрати випадкові дані з фізичних подій, таких як тепловий шум, радіоактивний розпад, атмосферні шуми, мерехтіння світла, час старту системи та інших випадкових подій.
- Введення користувача: система може отримувати випадкові дані від користувача, такі як клавіатурний ввід, клацання мишки, голосові команди та інші.
- Аналіз даних: система може отримувати випадкові дані з аналізу даних, таких як зображення, звук, відео, текстові дані та інші.

Ці джерела випадковості можуть бути оброблені та використані для створення послідовності випадкових бітів, які можуть бути використані в якості ключів шифрування, паролів та інших криптографічних об'єктів.

3.2.4 Реалізація інтерфейсу

Для реалізації інтерфейсу між серійним портом та генератором випадкових байтів потрібно розробити клас, який може зчитувати, писати випадкові байти та закривати потік серійного порту. Цей клас має чотири методи та конструктор:

__init__ - це конструктор класу, який приймає об'єкт типу `serial.Serial` як вхідний параметр. Це об'єкт, який представляє серійний порт, з яким будемо працювати.

```
class QRNGSerialPort:
    def __init__(self, serial_port: serial.Serial):
        self.serial_port = serial_port
```

Рисунок 3.8– Конструктор классу інтерфейса

write_random_bytes_to_serial_port - цей метод генерує випадковий байтовий рядок довжиною 256 байтів за допомогою функції `secrets.token_bytes(256)` та записує цей рядок в серійний порт. Для

зчитування з /dev/urandom використовується пакет secrets, який зчитує потік та відразу закриває його.

```
def write_random_bytes_to_serial_port(self):
    self.serial_port.write(secrets.token_bytes(256))
```

Рисунок 3.9 – метод запису випадкових байтів до серійного порту

read_random_bytes_to_serial_port - цей метод читає випадкові байти з серійного порту. Довжина байтів, що читаються, визначається за допомогою параметра `length` та розміру типу даних `dtype`. Прочитані байти конвертуються в масив NumPy відповідного типу даних. Після читання, серійний порт закривається.

```
def read_random_bytes_to_serial_port(self, length, dtype):
    byte_length = length * np.dtype(dtype).itemsize
    random_bytes = self.serial_port.read(byte_length)
    self.serial_port.close()
    return np.frombuffer(random_bytes, dtype=dtype)
```

Рисунок 3.10 – Зчитування та конвертація байтів

close_port - цей метод закриває серійний порт, відключаючи від нього.

data_type_is_valid - цей статичний метод приймає параметр `dtype` та перевіряє, чи є це валідним типом даних. Це відбувається за допомогою спроби конвертувати `dtype` в об'єкт `np.dtype`. Якщо конвертація проходить без помилок, метод повертає `True`, в іншому випадку - `False`.

```
@staticmethod
def data_type_is_valid(dtype):
    return dtype in allowed_dtypes
```

Рисунок 3.11 – Перевірка чи можна обробити вказаний тип даних

Далі нам потрібно написати нескінчений цикл для постійного запису байтів у буфер (рис 3.12).

```
16      qrng_ser_port = QRNGSerialPort(ser)
17      atexit.register(qrng_ser_port.close_port)
18      while True:
19          qrng_ser_port.write_random_bytes_to_serial_port()
20
```

Рисунок 3.12 – Неперервний запис байтів до серійного порту

Після розробки нам треба зберегти назви та версії встановлених пакетів щоб мати змогу встановити їх на пристрої. Для цього потрібно перечислити усі встановлені пакети та записати їх у файл (рис. 3.13).

```
(venv) [redacted]:~/diplom $ pip freeze > requirements.txt
(venv) [redacted]:~/diplom $ cat requirements.txt
anyio==3.6.2
click==8.1.3
fastapi==0.95.1
h11==0.14.0
httptools==0.5.0
idna==3.4
pydantic==1.10.7
python-dotenv==1.0.0
PyYAML==6.0
sniffio==1.3.0
starlette==0.26.1
typing_extensions==4.5.0
uvicorn==0.22.0
uvloop==0.17.0
watchfiles==0.19.0
websockets==11.0.2
```

Рисунок 3.13 – Запис встановлених пакетів та їх версій у файл

Команда `pip freeze > requirements.txt` створює файл `requirements.txt`, в якому містяться назви пакетів Python, необхідних для роботи додатка. Цей файл можна використовувати для автоматичної встановлення необхідних пакетів на іншому комп'ютері або сервері. Конкретніше, команда `pip freeze` виводить список всіх встановлених пакетів Python разом з їх версіями, а символ `>` перенаправляє цей вивід у файл `requirements.txt`. Отже, після запуску цієї команди, у файлі `requirements.txt` будуть збережені назви пакетів і їх версії, розділені символом нового рядка. Цей файл можна потім використовувати для автоматичного встановлення необхідних пакетів на іншому комп'ютері або сервері. Для цього достатньо скопіювати файл `requirements.txt` на інший комп'ютер і використати команду `pip install -r

requirements.txt`, що автоматично встановить всі необхідні пакети Python з їх версіями, вказаними у файлі.

3.3 Розробка веб-додатку

Наш веб-додаток повинен приймати HTTP запити методом GET з параметрами для типу чисел та кількістю чисел які плануємо сгенерувати.

HTTP - це протокол передачі гіпертексту, що використовується для передачі даних, особливо документів HTML, в Інтернеті. У 2015 році було представлено HTTP/2. HTTP - це протокол клієнт-сервер на основі запитів та відповідей. Клієнт (зазвичай веб-браузер) надсилає запит до сервера, а сервер повертає відповідь, яка містить статус виконання запиту та потрібні дані. HTTP працює зверху на базі протоколу TCP/IP та використовує порт 80. Основні аспекти протоколу HTTP включають:

1) Методи запиту: HTTP визначає кілька методів запиту, таких як GET, POST, PUT, DELETE, HEAD, PATCH та інші, щоб вказати, яка дія вимагається від сервера. Наприклад, метод GET використовується для отримання ресурсу, а метод POST - для надсилання даних на сервер.
2) Коди статусу: HTTP використовує коди статусу для вказівки результатів виконання запиту. Наприклад, код статусу 200 означає "OK", а код статусу 404 означає "Не знайдено".
3) Заголовки: Заголовки HTTP дозволяють передавати додаткову інформацію про запит або відповідь, таку як тип вмісту, кодування, довжину вмісту тощо. Заголовки можуть використовуватися для керування кешуванням, перенаправленням, безпекою та іншими аспектами.
4) Структура повідомлення: HTTP-повідомлення складається зі стартового рядка, заголовків та тіла повідомлення. Стартовий рядок містить інформацію про метод запиту, URI ресурсу та версію протоколу для запитів, а для відповідей - версію протоколу, код статусу та текстове пояснення статусу. Заголовки містять рядки з ключем та значенням,

розділеними двокрапкою. Тіло повідомлення містить дані для відправки, якщо такі є.

5) Управління з'єднанням: HTTP/1.1 використовує постійні з'єднання за замовчуванням, що дозволяє передавати кілька HTTP-повідомлень через одне TCP-з'єднання без повторного встановлення з'єднання для кожного запиту. Це покращує продуктивність та швидкість передачі даних.
6) Безпека: Для безпечної передачі даних протокол HTTP може використовувати HTTPS, що використовує шифрування на основі TLS або SSL для захисту даних від перехоплення та змін.

HTTP/2 вносить декілька поліпшень до протоколу HTTP/1.1, таких як мультиплексування потоків, стиснення заголовків, пріоритет запитів та інші оптимізації для підвищення продуктивності та швидкодії. HTTP - це один з основних протоколів Інтернету, який лежить в основі веб-перегляду, API, передачі даних та багатьох інших онлайн-сервісів. Він продовжує розвиватися та покращуватися для забезпечення надійного, швидкого та безпечного обміну інформацією у всесвітньому мережі.

Метод GET є одним з основних методів HTTP-протоколу і використовується для запиту ресурсів з веб-сервера. У методі GET, клієнт надсилає запит на сервер з запитом отримати певний ресурс, наприклад веб-сторінку, зображення або інший файл. Запит містить адресу ресурсу, який клієнт хоче отримати. При отриманні запиту методу GET, веб-сервер оброблює запит і шукає запрошений ресурс. Якщо запитуваний ресурс знайдено, веб-сервер надсилає його клієнту у відповіді на запит. Якщо запитуваний ресурс не знайдено, веб-сервер може повернути код помилки HTTP, такий як 404 (ресурс не знайдено). Метод GET передає параметри запиту через URL-адресу в рядку запит. Параметри запиту можуть бути використані для передачі додаткової інформації про запитуваний ресурс. Параметри передаються у вигляді пар ключ-значення, розділених символом "&". Метод GET також може використовуватися для отримання статусу

ресурсу, не вимагаючи повернення контенту. Наприклад, клієнт може надіслати запит методом GET для перевірки того, чи є ресурс доступним, і отримати відповідь з кодом статусу HTTP, таким як 200 (OK) або 404 (файл не знайдено). Загалом, метод GET є простим і ефективним способом отримання ресурсів з веб-сервера, і він широко використовується в інтернет-додатках та веб-сервісах.

Тому наш запит на веб-сервер можна описати як “GET /api/qrng?dtype=[dtype]&length=[int]”, де length описує кількість чисел на виході і відображається від 1 до певного ліміту, а dtype це тип (або класс з точки зору Python) чисел, який впливає на їх максимальне значення, можливість бути від’ємними та мати дрібну частину.

Для написання веб-додатку нам потрібно:

- розробити код, який буде відповідати на вище зазначений запит у контексті FastAPI
- під’єднати фреймворк до серверу uvicorn, який буде обслуговувати запроси
- запустити сервер на окремому порті TCP

Для того щоб обробляти запити розробленим кодом на Python потрібно використати ASGI. ASGI - це інтерфейс між веб-серверами і фреймворками, який дозволяє використовувати асинхронні фреймворки Python для створення веб-додатків з підтримкою асинхронного ввід/вивід. ASGI замінює старий WSGI, який не має підтримки асинхронної роботи і забезпечує зв'язок між веб-сервером і фреймворком. ASGI дозволяє асинхронному фреймворку отримувати запити та відповідати на них асинхронно, що дозволяє підтримувати більше запитів на одиницю часу, в порівнянні з традиційним фреймворком, що працює за синхронними правилами. ASGI можна використовувати для побудови сучасних веб-додатків, які використовують WebSocket, Server-Sent Events і HTTP/2, які вимагають асинхронного ввід/вивід. Розберемо код ініціалізації веб-додатку (рис. 3.14):

```
app = FastAPI()

qrng_ser_port = QRNGSerialPort(ser)

@app.on_event("shutdown")
def shutdown_event():
    qrng_ser_port.close_port()
```

Рисунок 3.14 – Код ініціалізації веб-додатку

Ініціалізація об’єкта app використовуючи клас FastAPI для того що цей об’єкт після присвоєння властивостей пішов аргументом у ASGI для серверної обробки запитів та ініціалізація класу інтерфейсу серійного порту. Для коректної роботи з портом при завершенні додатку треба закрити потоку.

Розберемо код обробки запитів на генерацію випадкових байтів (рис. 3.15). Після того, як за допомогою декоратора був присвоєний путь /api/generate-numbers до функції generate_numbers з параметрами «тип даних» та «кількість згенерованих чисел » усі запити по цьому путі до сервера будуть оброблятися цією функцією. У разі некоректного запиту користувачу буде повернута відповідь зі статусом 400, що означає поганий запит.

```
@app.get("/api/generate-numbers")
async def generate_numbers(dtype: str = "int32", length: int = 1):
    if not qrng_ser_port.data_type_is_valid(dtype):
        raise HTTPException(status_code=400, detail="Data type is invalid")
    res = qrng_ser_port.read_random_bytes_to_serial_port(length, dtype)
    return {"numbers": res.tolist()}
```

Рисунок 3.15 – Код обробки запитів на генерацію

Після копіювання файлів на пристрій потрібно3 запустити веб-сервер uvicorn. Для того щоб це зробити треба створити та активувати віртуальну середу, встановити пакета з файлу, який перенесли раніше, та ввести команду (рис 3.16).

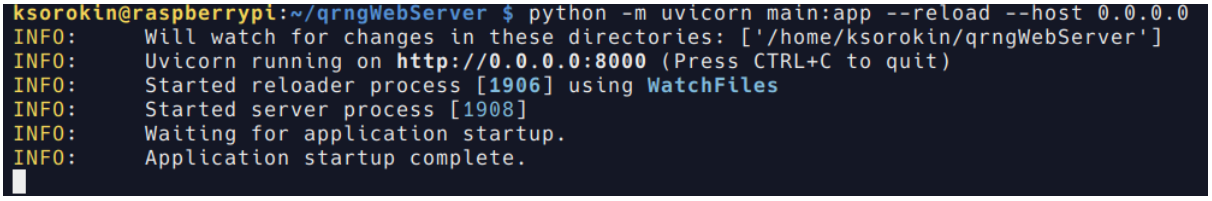

Рисунок 3.16 – Команда запуску веб-сервера

Ця команда запускає веб-сервер Uvicorn у режимі розробки та налаштовує його для обробки вхідних запитів на локальному комп'ютері за допомогою Python-файлу main.py та об'єкту app. Крім того, параметр reload дозволяє автоматично перезавантажувати сервер при зміні коду в додатку, а параметр host 0.0.0.0 налаштовує сервер на прослуховування всіх вхідних з'єднань на будь-якій доступній мережевій інтерфейсі.

Отже, ця команда встановлює з'єднання між веб-додатком і веб-сервером, який обробляє вхідні HTTP-запити і надсилає відповіді клієнту. Коли запускається веб-сервер, він починає прослуховувати вхідні з'єднання на вказаному хості та порту, оброблює запити, які отримує, та надсилає відповіді клієнту через відповідне з'єднання. У випадку Uvicorn, він працює як ASGI-сервер і може обробляти запити, які використовують ASGI-протокол.

Для тестування треба зробити запит на веб-сервер (рис. 3.17) використовуючи путь та параметри.

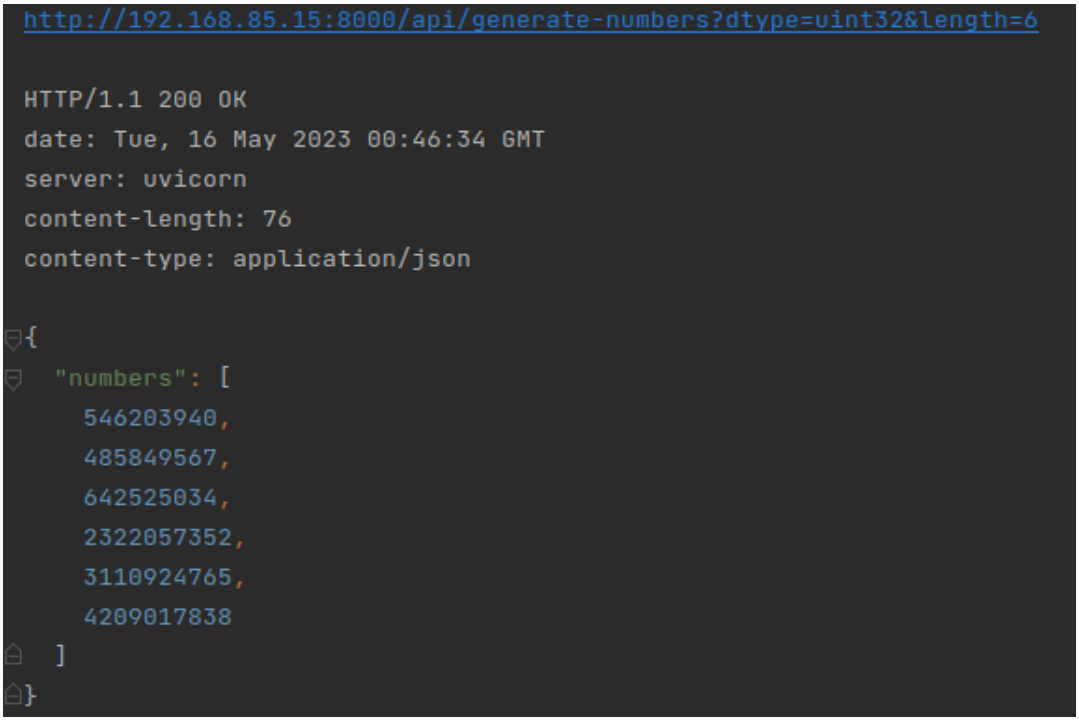

Рисунок 3.17 – Результат запиту на генерацію випадкових чисел

Результат цього запиту повернув результат виконання функції generate_numbers з параметрами «кількість чисел», яка дорівнює 6 та «тип даних» uint32, що означає відсутність від'ємних чисел. Також з заголовків протоколу HTTP можна побачити дату, веб-сервер, тип даних результату та його довжину. Усі запити на веб-сервер повернули інші числа, які були більше нуля. Запити на генерацію чисел, які допускають присутність від'ємних, повертали у тому числі і від'ємні числа (рис. 3.18).

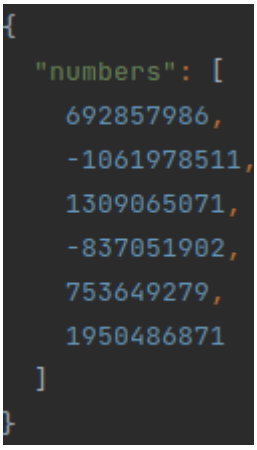

Рисунок 3.18 – приклад відповіді з від'ємними числами

Тепер необхідно згенерувати сертифікати для використання TLS з HTTP для уникнення атак типу MITM. Протокол TLS - це криптографічний протокол, що надає захищене з'єднання між двома кінцевими точками в мережі, зазвичай між клієнтським застосунком (наприклад, веб-браузером) і сервером. Основний RFC, що регулює TLS, - це RFC 8446 для TLS 1.3 і RFC 5246 для TLS1.2. TLS також забезпечує цілісність даних за допомогою коду аутентифікації повідомлень, або в TLS 1.3 - за допомогою аутентифікованого шифрування з додатковими даним. Основні кроки роботи TLS:

1) Угода про параметри з'єднання: клієнт і сервер обмінюються повідомленнями, щоб узгодити версію TLS, набори шифрів та інші параметри, необхідні для встановлення безпечного з'єднання. Також вони обмінюються випадковими значеннями, які будуть використовуватися під час встановлення ключів.
2) Аутентифікація: Сервер надсилає клієнту свій цифровий сертифікат, який містить відкритий ключ сервера і підписаний довіреним центром сертифікації. Клієнт перевіряє цей сертифікат, щоб переконатися, що він спілкується із справжнім сервером. В TLS 1.3 взаємна аутентифікація також можлива, коли клієнт також надсилає свій сертифікат серверу.
3) Створення секретного ключа: клієнт і сервер створюють спільний секретний ключ. Він буде використовуватися для шифрування та дешифрування даних, які передаються між ними.
4) Шифрування та передача даних: Після встановлення секретного ключа, дані можуть бути безпечно передані. Дані шифруються за допомогою секретного ключа перед відправленням і дешифруються після отримання. Це гарантує конфіденційність даних.

Цей процес надає безпеку, автентичність та цілісність даних, що передаються між клієнтом і сервером. Для генерації TLS сертифіката знадобиться програмне забезпечення для створення сертифікатів. Однією з популярних утиліт для цього є OpenSSL. Після встановлення програмного

забезпечення потрібно згенерувати приватний ключ та використовуючи ключ створити сертифікат. Під час виконання цієї команди OpenSSL запитає ввести інформацію, яка буде включена в сертифікат, таку як назва країни, назва організації, ім'я веб-сайту тощо. Ці кроки згенерують сертифікат, який можна використовувати для тестування або внутрішнього використання, але він не буде довіряти більшості клієнтських браузерів. Для отримання сертифіката, який буде довіряти браузери, потрібно подати запит на сертифікат до сертифікаційного центру .

4 ТЕСТУВАННЯ СИСТЕМИ

4.1 Тестування під навантаженням

Тестування під навантаженням - це вид тестування, який вимірює роботу системи під значним навантаженням для перевірки її відповідності та ефективності. Його метою є визначення, як система впорається з великими об'ємами даних, користувачів, запитів. Для тестування використаємо Jmeter.
Процес тестування під навантаженням складається з наступних основних кроків:

1) Планування – визначається цілі тестування, сценарії та метрики, які потрібно виміряти. Цілі нашого тестування це виявлення максимальної кількості запитів у секунду та виявлення вузьких місць.
2) Створення тесту - створюються сценарії тестування, які симулюють різні типи навантаження на систему. Наприклад, можна створити сценарій, який симулює велику кількість користувачів, які одночасно використовують API. Сценарій, який потрібно симулювати, це велика кількість запитів на протязі одної хвилини.
3) Виконання тесту - запускається тест та моніторинг систему. Вимірюються різні метрики, такі як час відповіді, пропускна здатність, помилки тощо. Конфігурація, яка була використана під час тесту(рис 4.1) використовує 8 потоків, не зупиняється на помилках та триває 60 секунд.
4) Аналіз результатів - після завершення тесту аналізуються результати, щоб визначити, чи впоралася система з навантаженням. Також визначаються можливі проблеми, такі як вузькі місця в процесі обробки, та виробляєте рекомендації щодо поліпшення процесу.

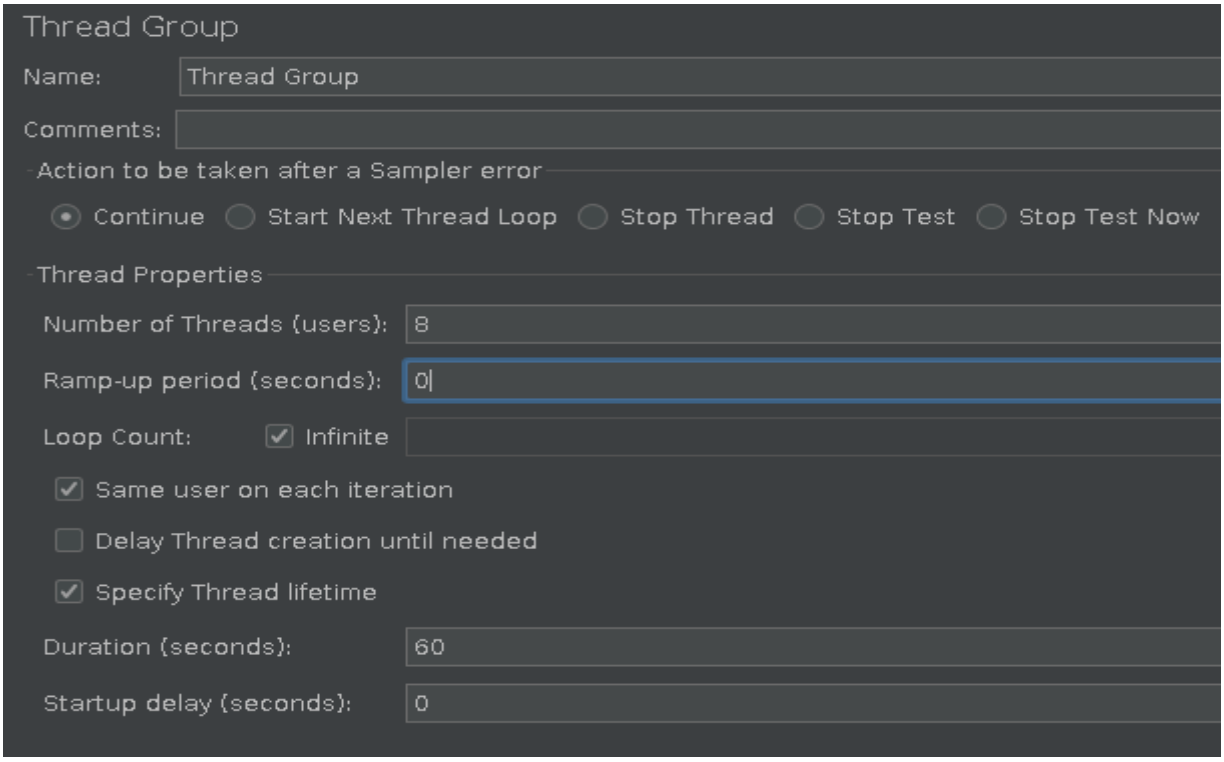

Рисунок 4.1 – конфігурація кількості потоків тесту

Після проведення тесту та генерації відліку за його результатами, отримаємо статистику помилок, пропускної спроможності, та часу відповіді на запит. Статистика помилок показує (рис. 4.2), що 0.03 % усіх запитів отримали помилку у якості відповіді.

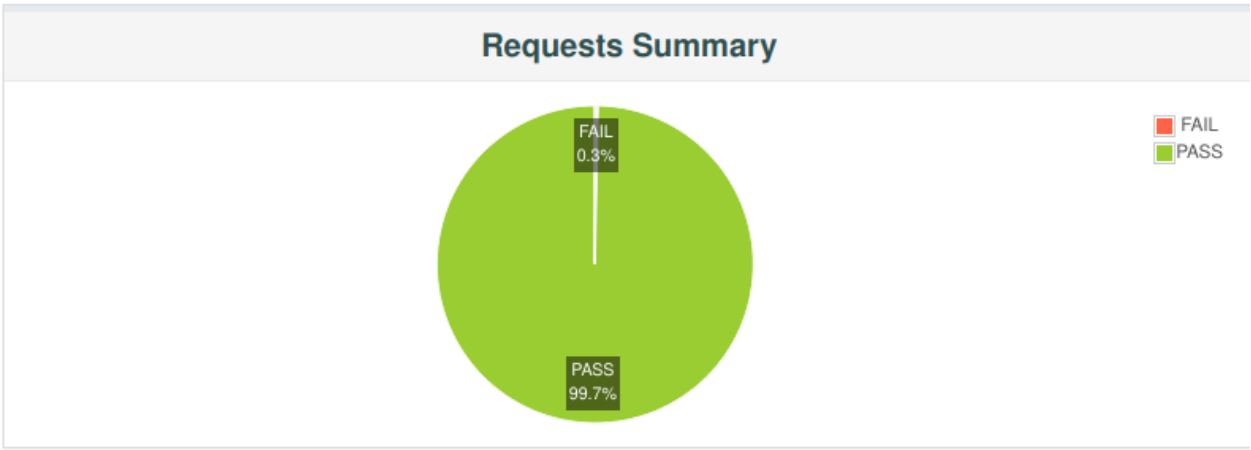

Рисунок 4.2 – відношення помилок до успішних запитів

У результаті аналізу статистики (рис. 4.3) можна отримати дві найчастіших помилки: не відповідь на запит (перенавантаження) та помилка на сервері.

Errors

Type of error	Number of errors	% in errors	% in all samples
Non HTTP response code: org.apache.http.NoHttpResponseException/Non HTTP response message: 192.168.85.15:8000 failed to respond	41	50.00%	0.15%
500/Internal Server Error	41	50.00%	0.15%

Рисунок 4.3 – Типи помилок

Скористаємося історією помилок веб-додатку для аналізу «вузького місця» або потенційної проблеми. У історії є одна помилка (рис 4.4), яка вказує на те, що UART не має достатньої швидкості передачі.

```
serial.serialutil.SerialException: device reports readiness to read but returned n
o data (device disconnected or multiple access on port?)
```

Рисунок 4.4 – Помилка з історії веб-додатку

Розберемо наступну помилку використовуючи дані за зміною в часі. Проаналізувавши зміну інтервалу між відповідями за часом (рис 4.5), пропускну спроможність (рис 4.6) та кількість запитів в секунду (рис 4.7) отримаємо деградацію продуктивності системи.

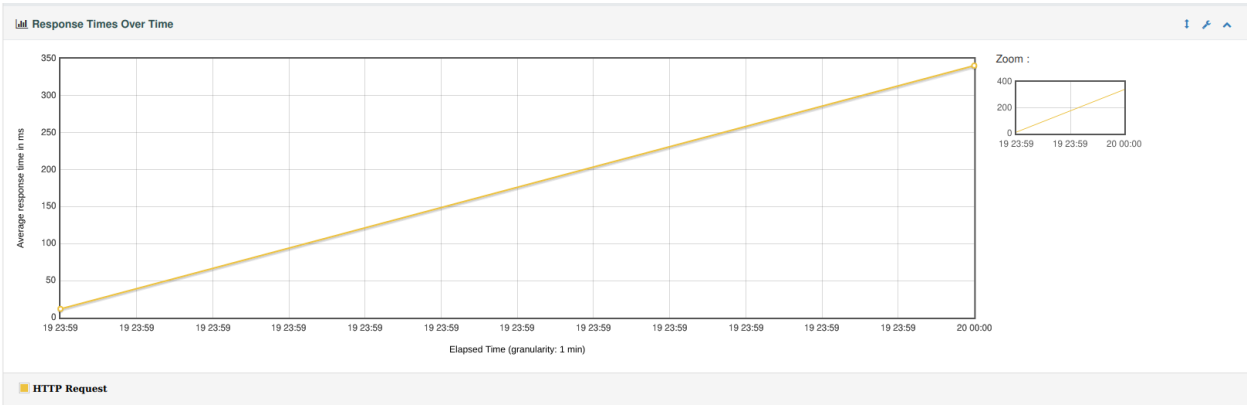

Рисунок 4.5 – Зміна інтервалу відповіді за часом

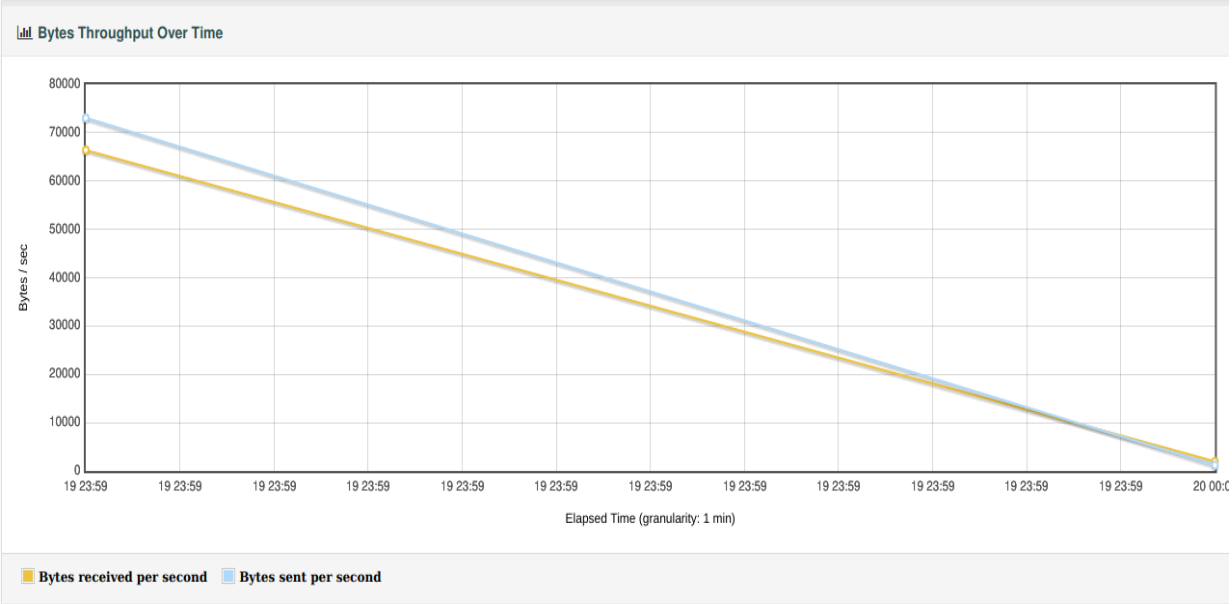

Рисунок 4.6 – Пропускна спроможність за часом

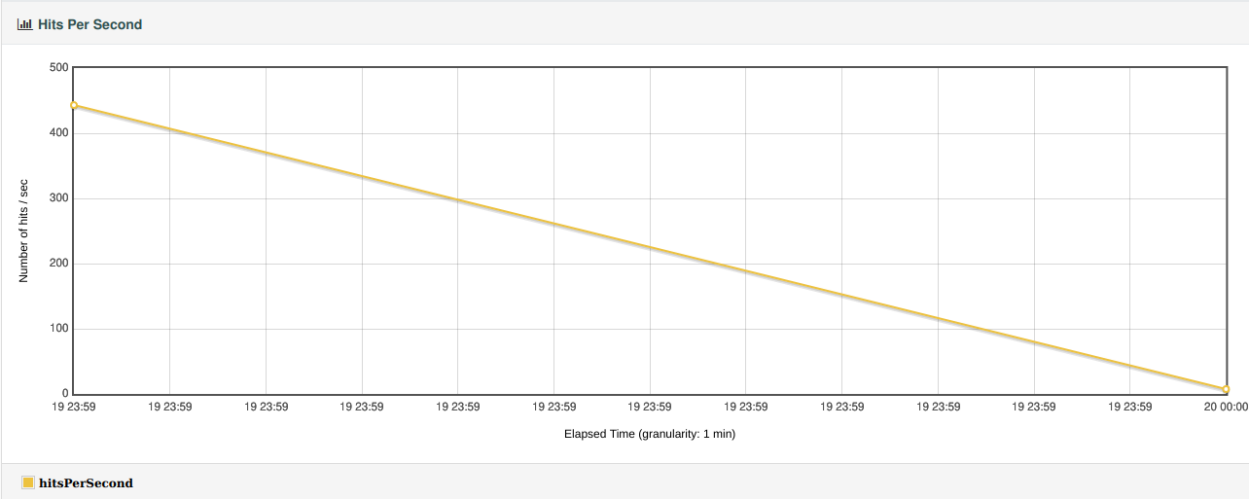

Рисунок 4.7 – Кількість запитів в секунду

Причиною зниження кількості відповідей у секунду можуть бути помилки через недостатню провідність UART або перегрів центрального процесора, оскільки він не маж елемента охолодження.

4.2 Тестування безпеки

Розберемо вектори атаки на систему. Через те, що кількісті компонентів системи малочисельна та лише один компонент має вихід в Інтернет – web-додаток – розглянемо атаки на цей компонент. Він приймає запит з параметрами кількості та типу згенерованих чисел, тому треба перевірити можливість ін’єкції коду. Обидва параметри обробляються як строки, після

чого перевіряються на коректність. За результатами аналізу вразливості параметрів можливості для ін’єкції знайдено не було. Конфіденційність кожної відповіді гарантується сертифікатами протоколу HTTPS, тому єдиний вектор для атак це порушення доступності.

З точки зору мережі є лише один вразливий компонент – SSH-сервер. Тестування безпеки SSH серверу важливе для виявлення можливих слабкостей або вразливостей, які можуть бути використані зловмисниками для злому системи або крадіжки даних. Це може включати перевірку конфігурації SSH, сили паролів, використання ключів аутентифікації та інші аспекти безпеки. Можливі дії для тестування безпеки SSH серверу:

- Перевірка конфігурації SSH - вона має містити обмеження на вхід, включаючи заборону входу для користувача root, обмеження на методи аутентифікації та інші опції, що покращують безпеку.
- Використання сканерів вразливостей - інструменти, які можуть сканувати SSH сервер на наявність відомих вразливостей, можуть допомогти вам ідентифікувати та виправити можливі проблеми безпеки.
- Використання ключів для аутентифікації - аутентифікація за допомогою ключів часто вважається безпечнішою, ніж аутентифікація за допомогою паролів, оскільки ключ складніше вгадати методом підбору.
- Аудит системних журналів: перевірка системних журналів на підозрілі активності, які можуть вказувати на проблеми безпеки або спроби вторгнення.

Конфігурація на даний момент використовує аутентифікацію по паролю. Спробуємо вгадати його за допомогою сканеру nmap. Для цього створимо файл з логіном користувача та використаємо стандартний словник з паролями(рис 4.8):

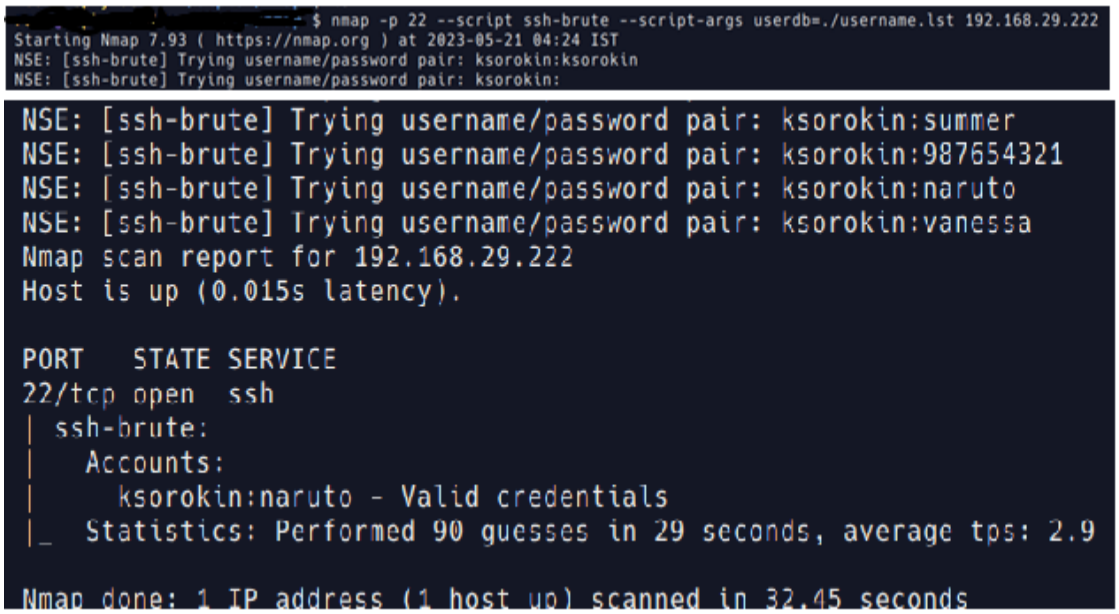

Рисунок 4.8 – Результат атаки подбором

Для покращення безпеки потрібно використати ключі для аутентифікації. Для використання ключів для аутентифікації в SSH, потрібно згенерувати пару ключів (приватний та відкритий) на клієнтській машині, а потім скопіювати відкритий ключ на сервер.

ВИСНОВКИ

В результаті, розроблена система здатна генерувати потік випадкових байті, писати його у серійний порт, збирати цей потік трансформуючи його у числа на основі параметрів переданих при запиту, з регульованою довжиною кількості чисел і регульованим типом даних.

Після порівняльного аналізу сучасних технологій було прийнято рішення використовувати Python з фреймворком FastAPI для розробки веб-серверу та бібліотеки pyserial для взаємодій с серійним портом.

Система має великий потенціал і може бути вдосконалена в майбутньому, наприклад, шляхом додавання елементів веб-сайту, додавання Websockets, корекції помилок, які призводять до загублення запитів. Також систему необхідно вдосконали та роздивитися захист від DDoS.

Крім того, реалізація має схеми згідно з якими вона була розроблена та які наглядно демонструють алгоритм реалізація та її розгортання на фізичних приладах.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. UML Specification URL: https://www.omg.org/spec/UML/2.5.1/PDF (дата звернення 05.12.2017)
2. SSH Transport Layer URL: https://www.rfc-editor.org/rfc/rfc4253.txt (дата звернення 26.01.2006)
3. SSH Client Authentication Layer URL: https://www.rfc-editor.org/rfc/rfc4252.txt (дата звернення 26.01.2006)
4. SSH Connection Layer URL: https://www.rfc-editor.org/rfc/rfc4254.txt (дата звернення 26.01.2006)
5. SSH Protocol Assigned numbers URL: https://www.rfc-editor.org/rfc/rfc4250.txt (дата звернення 26.01.2006)
6. HTTP/1.1 Message Syntax and Routing URL: https://www.rfc-editor.org/rfc/rfc7230.txt (дата звернення 17.06.2014)
7. HTTP/1.1 Semantics and Content URL: https://www.rfc-editor.org/rfc/rfc7231.txt (дата звернення 17.06.2014)
8. HTTP/1.1 Conditional Requests URL: https://www.rfc-editor.org/rfc/rfc7232.txt (дата звернення 17.06.2014)
9. HTTP/1.1 Range Requests URL: https://www.rfc-editor.org/rfc/rfc7233.txt (дата звернення 17.06.2014)
10. HTTP/1.1 Authentication URL: https://www.rfc-editor.org/rfc/rfc7235.txt (дата звернення 17.06.2014)
11. HTTP Over TLS URL: https://www.rfc-editor.org/rfc/rfc2818.txt (дата звернення 09.05.2000)
12. Holly McKinley. SSL and TLS: A Beginners Guide. // SANS Institute Information Security Reading Room. – 2019.

13. Global Sign: What is an SSL Certificate? URL: https://www.globalsign.com/en/ssl-information-center/what-is-an-ssl-certificate/ (дата звернення 21.08.2020)
14. Ван Тілборг Х. К. А. Основи криптології: Fundamentals of Cryptology. — М. : Мир, 2006. — 472 с.
15. IBM Knowledge Center: Cryptographic security protocols: SSL and TLS. URL: https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10630_.htm (дата звернення 06.06.2022)

ДОДАТОК А

Код веб-додатку

```
__author__ = “Kostiantyn Sorokin”

# Importing context manager to provide lifecycle method
from contextlib import asynccontextmanager
# Importing our framework, FastAPI
from fastapi import FastAPI
# Importing pyserial
import serial
# Importing class that reads from port and calculates bit number
from qrng_serial_port import QRNGSerialPort

# initializing iostream for serial port
ser = serial.Serial(
    port='/dev/ttyUSB0',
    baudrate=9600,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=1
)

#  initializing class for serial port operations
qrng_ser_port = QRNGSerialPort(ser)

# initializing FastAPI app according to asgi standard
```

```
app = FastAPI()

# on shutdown we want to close the port
@app.on_event("shutdown")
def shutdown_event():
    qrng_ser_port.close_port()

# Initializing the function below as endpoint
@app.get("/api/generate-numbers")
async def generate_numbers(dtype: str = "int32", length: int = 1):
    res = qrng_ser_port.read_random_bytes_to_serial_port(length, dtype)
    return {"numbers": res.tolist()}
```

Код классу для роботи з портом

```
__author__ = “Kostiantyn Sorokin”

import random  # Importing random package to generate random bytes
import numpy as np  # Importing numpy to work with data type
import serial  # Importing serial for type safety and autocompletion



class QRNGSerialPort:
    def __init__(self, serial_port: serial.Serial):
        self.serial_port = serial_port # get the serial_port from outside testing

     # write 8 random bytes to port
    def write_random_bytes_to_serial_port(self):
            self.serial_port.write(random.randbytes(8))
```

```
    # this method reads bytes from stream and serrializes them into array
    def read_random_bytes_to_serial_port(self, length, dtype):
        byte_length = length * np.dtype(dtype).itemsize
        random_bytes = self.serial_port.read(byte_length)
        return np.frombuffer(random_bytes, dtype=dtype)

    #closes port
    def close_port(self):
        self.serial_port.close()
```

Код генератору випадкових чисел

```
__author__ = “Kostiantyn Sorokin”
# Importing pyserial
import serial
# Importing atexit to provide lifecycle method
import atexit
# Importing class that reads from port and calculates bit number
from qrng_serial_port import QRNGSerialPort

if __name__ == "__main__":
    # initializing iostream for serial port
    ser = serial.Serial(
        port='/dev/ttyUSB0',
        baudrate=9600,
        parity=serial.PARITY_NONE,
        stopbits=serial.STOPBITS_ONE,
        bytesize=serial.EIGHTBITS,
        timeout=1
    )
```

```
#  initializing class for serial port operations
qrng_ser_port = QRNGSerialPort(ser)
# on shutdown we want to close the port
atexit.register(qrng_ser_port.close_port)
# write random bytes in endless for loop
while True:
    qrng_ser_port.write_random_bytes_to_serial_port()
```