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

До захисту допущено
В.о. завідувача кафедри

_____ **Микола ГРАЙВОРОНСЬКИЙ**
(підпис)

«_____»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною программою «Системи, технології та математичні методи кібербезпеки»
спеціальності: 125 «Кібербезпека»

на тему: Безпека систем управління інтелектуальним транспортом на базі платформи Linux

Виконав (-ла): здобувач вищої освіти **IV** курсу, групи ФБ-73
(шифр групи)

Петренко Кирил Миколайович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник к.т.н., доцент кафедри ІБ Стьопочкіна Ірина Валеріївна _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Київ - 2021 року

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

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри

_____ Микола ГРАЙВОРОНСЬКИЙ
(підпис)

«___»_____2021 р.

ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти

Петренку Кирилу Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Безпека систем управління інтелектуальним транспортом на базі платформи Linux»,
керівник роботи к.т.н., доцент кафедри ІБ Стьопочкіна Ірина Валеріївна,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « » _____ 2021 р. №

2. Термін подання здобувачем вищої освіти роботи 07 червня 2021 р.

3. Вихідні дані до роботи: технології IoT, операційна система Linux.

4. Зміст роботи: Огляд сучасних пристроїв IoT та їх архітектур, аналіз придатності Linux для пристроїв IoT, розробка політики безпеки для пристроїв інтелектуального транспорту, наочна реалізація механізмів безпеки за допомогою утиліт Linux, порівняння вплив даних механізмів на швидкодію.

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

6. Дата видачі завдання 5 жовтня 2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Формулювання теми дипломної роботи, визначення мети роботи та постановка відповідних до неї задач	27.08.2020 – 04.10.2020	
2	Огляд наукової літератури за темою та написання оглядового розділу роботи	05.10.2020 – 19.02.2021	
3	Вивчення методів та підходів, що використовуються для побудови пристроїв IoT	22.02.2021 – 13.03.2021	
4	Порівняння сучасних архітектур пристроїв IoT	15.03.2021 – 21.03.2021	
5	Огляд Linux платформи та оцінка її для використання у пристроях IoT	21.03.2021 – 01.04.2021	
6	Написання другого розділу дипломної роботи	01.04.2021 – 10.04.2021	
7	Проходження переддипломної практики (розроблення політки безпеки)	11.04.2021 – 15.05.2021	
8	Оцінка та аналіз отриманих результатів	16.05.2021 – 18.05.2021	
9	Написання третього розділу дипломної роботи	20.05.2021 – 27.05.2021	
10	Створення презентації для захисту дипломної роботи	29.05.2021 – 01.06.2021	
11	Передзахист дипломної роботи	03.06.2021	
12	Допрацювання дипломної роботи та презентації	03.06.2021 – 16.06.2021	
13	Захист дипломної роботи	18.06.2021	

Здобувач вищої освіти _____ Кирил ПЕТРЕНКО
(підпис)

Керівник роботи _____ Ірина СТЬОПОЧКІНА
(підпис)

РЕФЕРАТ

Дипломна робота має обсяг 49 сторінок, містить 18 рисунків, 5 таблиць, 7 посилань та 1 додаток.

Стрімке та невпинне зростання пристроїв IoT у купі з складністю запровадити механізми безпеки призводить до організації величезних ботнетів які можуть порушити доступність інтернет каналу країни, та інший атак. Надшвидке зростання кількості пристроїв у мережі та використання застарілих механізмів та архітектур є першопричинами таких уразливостей.

Об’єктом дослідження є пристрої IoT.

Предметом дослідження є механізми та застосунки ОС Linux та можливість використання їх у системах інтелектуального транспорту.

Метою роботи є дослідження можливостей Linux платформи для використання останньої як головної ОС у пристроях IoT та розробка політики безпеки для даних пристроїв у сфері управління інтелектуальним транспортом.

Дана робота містить опис переваг використання Linux у пристроях IoT, приклади використання базових застосунків для забезпечення конфіденційності, цілісності та доступності. У ході роботи була розроблена політика безпеки для пристроїв інтелектуального транспорту, а також порівняння платформи яка існує на даний час та запропонованої платформи яка базується на Linux.

Ключові слова: Розумний транспорт, ОС Linux, IoT, V2I комунікація.

ABSTRACT

Thesis has a volume of 49 pages, contains 18 figures, 5 tables, 7 references and 1 appendix. The rapid and relentless growth of IoT devices, along with the difficulty of implementing security mechanisms, leads to the organization of huge botnets that can disrupt the availability of the country's Internet channel, and other attacks. The rapid growth in the number of devices on the network and the use of outdated mechanisms and architectures are the root causes of such vulnerabilities.

The object of research is IoT devices.

The subject of research is the mechanisms and applications of Linux and the possibility of using them in intelligent transport systems.

The aim of the work is to study the capabilities of the Linux platform to use the latter as the main OS in IoT devices and to develop a security policy for these devices in the field of intelligent transport management.

This work describes the benefits of using Linux in IoT devices, examples of using basic applications to ensure privacy, integrity and accessibility. In the course of the work, a security policy was developed for intelligent transport devices, as well as a comparison of the current platform and the proposed platform based on Linux.

Keywords: Smart transport, Linux OS, IoT, V2I communication.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

RFID – Radio Frequency IDentification – радіочастотна ідентифікація

IoT – internet of things – інтернет речей

API – application programming interface – прикладний програмний інтерфейс

ЕЦП – електронний цифровий підпис

ФС – файлова система

ПЗП – постійний запам'ятовувальний пристрій

ВСТУП

У сучасному світі пристрої IoT займають дуже важливе місце у сучасних мегаполісах, а значить і в житті населення таких мегаполісів. За останнє десятиліття пристроїв IoT стало в декілька тисяч разів більше. В основному такі пристрої полегшують рутинні задачі та автоматизують більшість процесів виключаючи з останніх людський фактор.

Актуальність роботи. Пристрої IoT останнім часом дуже “щільно” увійшли у життя сучасних людей. Все більше девайсів які раньше були надто простими, такі як лампи розжарювання, на даний час можуть являтися високотехнологічним пристроєм. Попри всі переваги таких пристроїв вони вже не можуть просто реалізовувати над прості речі в керуванні собою. З ростом апаратних ресурсів, кількості пристроїв та їх значимості у повсякденних процесах пристрої IoT потребують нової платформи та політики безпеки.

Метою роботи є дослідження можливостей Linux платформи для використання останньої як головної ОС у пристроях IoT та розробка політики безпеки для даних пристроїв у сфері управління інтелектуальним транспортом.

Для досягнення мети роботи, поставлено такі **завдання**:

1. Дослідити літературу що описує сучасну архітектуру пристроїв IoT;
2. Дослідити можливості архітектури Linux та застосунків для їх використання в пристроях IoT;
3. Розробка політики безпеки для пристроїв IoT на базі Linux;
4. Наочна демонстрація вирішення проблем конфіденційності, цілісності та доступності для таких пристроїв;
5. Оцінити результати спроектованої системи.

Об’єктом дослідження є пристрої IoT.

Предметом дослідження є механізми та застосунки ОС Linux та можливість використання їх у системах інтелектуального транспорту.

1 ОГЛЯД INTERNET OF THINGS

Термін “Інтернет речей” вперше було використано Кевіном Ештоном у 1999 році [1], коли він працював над оптимізацією виробництва. Він запропонував більш швидкий обмін даними між пристроями за допомогою RFID міток. І у 2004 році в журналі Scientific American публікується стаття [2], у якій наглядно описується використання “розумних речей” у побуті (рисунок 1.1).

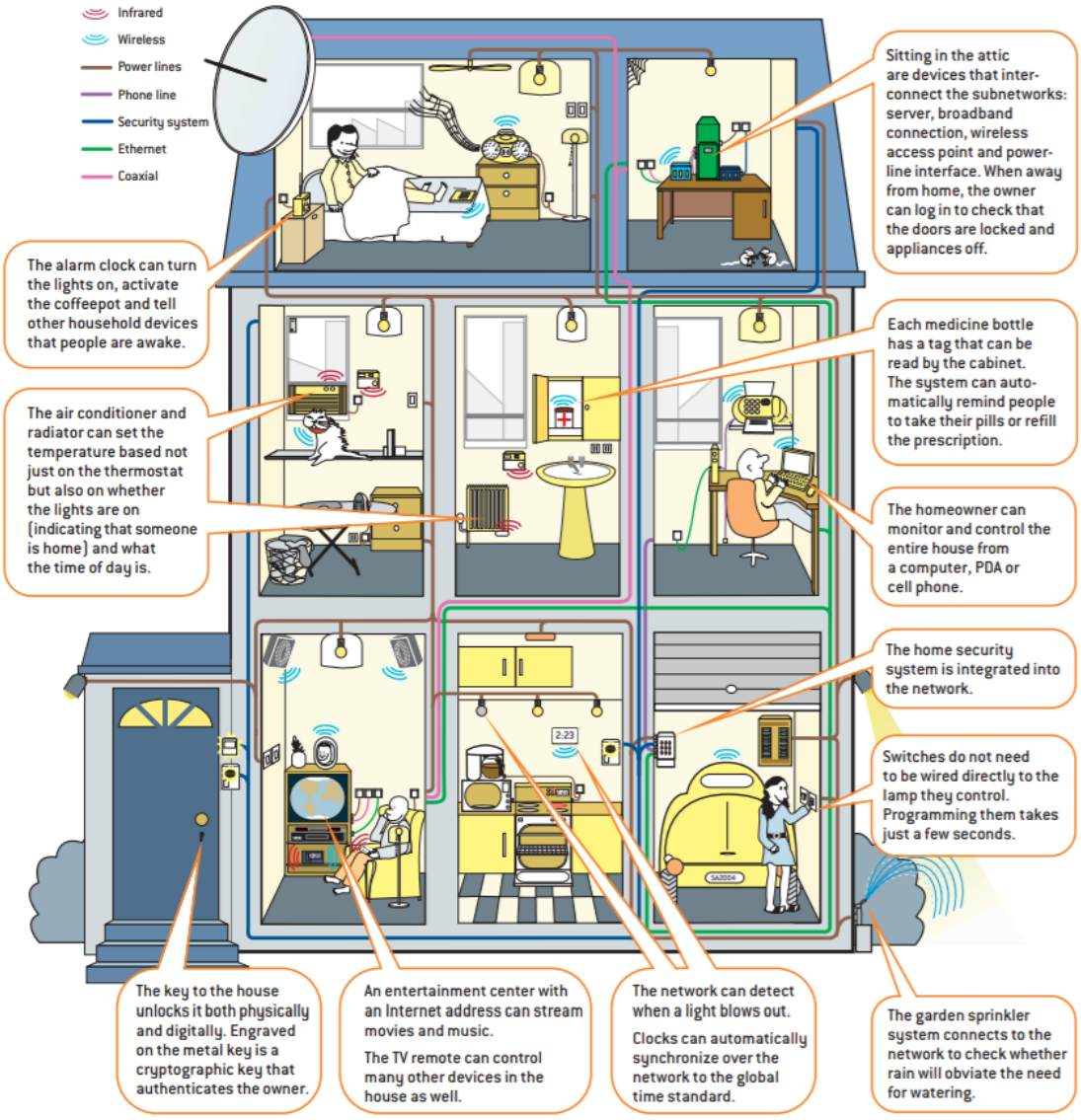

Рисунок 1.1 – Опис взаємодій розумного дому з[2]

Вже в 2004 році автори повністю описали функціонал розумного дому. Але широкого розповсюдження та впровадження такі речі набули через десять років.

1.1 Апаратна платформа

На той час розповсюдженість обчислювальної техніки була надто мала оскільки вона була надто дорогою. Це несло за собою відсутність будь якого середовища передачі даних так, як комп’ютер якщо і був, то був тільки один. У свою чергу пристрої IoT будувались на примітивних мікросхемах і як наслідок були дуже примітивні по функціоналу. Для зв’язку використовувались дротові лінії типу One Wire або телефоні лінії так, як на той час були широко розповсюджені. З прогресом виробництва та здешевлення напівпровідникових елементів на зміну телефонним лініям зв'язку прийшов Wi-Fi. У свою чергу безпровідний зв’язок для пристроїв IoT є ідеальним:

- Простота масштабування

  Для підключення нового пристрою, або зміни локації для вже підключеного не потрібно ні прокладати нові лінії зв’язку ні переналаштовувати пристрій. Також не є необхідністю прокладати окремі лінії як у випадку з Ethernet де кожен хост потребує окремої лінії, що у свою чергу потребує додаткових витрат.
- Спрощення кінцевого пристрою

  Задачі маршрутизації, передачі даних та автентифікації переносяться на існуючу мережу. Як наслідок кінцевий пристрій може містити тільки модуль підключення до цієї мережі і не витрачати процесорний час на управління мережею.
- Розмежування середовища передачі даних та пристроїв які ці дані генерують

  У свою чергу це спрощує розширення мережі незалежно від пристроїв.

З розповсюдженням смартфонів та персональних комп’ютерів Wi-Fi можна знайти в будь якому будинку. Це і призвело до розповсюдження пристроїв IoT які працюють у мережі Wi-Fi.

У таких пристроях одним з найрозповсюдженіших контролерів виступає сімейство мікроконтролерів ESP82xx та ESP32. В таблиці 1.1 наведено їх характеристики.

Таблиця 1.1 – Характеристики ESP8266 та ESP32

	ESP8266	ESP32
Micro Controller Unit	Xtensa Single-core 32-bit L106	Xtensa Dual-Core 32-bit LX6 with 600 DMIPS
Робоча частота	80 MHz	160 MHz
Оперативна пам’ять	160 kBytes	512 kBytes
Вбудована пам’ять	512 KiB - 4 MiB	2 MiB - 4 MiB

Такі пристрої вміють:

- Апаратно керувати корисним навантаженням
- Збирати інформацію з різноманітних датчиків
- Передавати дані у мережі
- Можливість виходу до мережі Інтернет.

1.2 Архітектура сучасних пристроїв IoT

Так як вбудованих ресурсів у контролера небагато і обробляти дані локально він не зможе, то розробники впроваджують клієнт-серверну архітектуру. Пристрій являє собою точку передачі, збору та перетворення інформації, вся керуюча та поведінкова “логіка” знаходиться у хмарі. Більшість виробників пристроїв такого класу притримуються саме такої архітектури. Це вдало знижує ціну на кінцеві пристрої так, як для задач такого класу не потрібно багато обчислювальних ресурсів, а базові характеристики подібних мікроконтролерів підходять під такі задачі. Будь яка керуюча інформація або оновлення пристрої отримують з хмари,

але обмеженість апаратних ресурсів та програмна складність впровадження шифрування та/або ЕЦП і їх подальший вплив на кінцеву ціну пристрою призводить до того що розробники не реалізують жодних заходів для для забезпечення цілісності, конфіденційності та доступності.

Окремо слід виділити архітектуру коли в мережі існує центральний шлюз до якого підключаються всі інші пристрої і вже через шлюз здійснює передачу даних до серверу. Але така архітектура притаманна пристроям які водночас:

- не мають самостійних ресурсів для виходу у мережу де знаходиться головний сервер
- мають мережевий протокол передачі даних з низьким енергоспоживанням та низькою дальністю безперебійної передачі.

Наочними прикладами можуть виступити мережі які базуються на протоколах ZigBee або Bluetooth Mesh. Такі протоколи за рахунок низького енергоспоживання дозволяють розробляти компактні пристрої які можуть працювати від неперезаряджуваних компактних елементах живлення протягом років. Вимоги від шлюзу - це апаратна реалізація цих протоколів та реалізація механізмів підключення до серверу. Попри розширені апаратні характеристики шлюз реалізує логічне розмежування пристроїв для подальної структурної передачі інформації серверу та не реалізує ніяких механізмів безпеки. Це наслідок складності розробки програмного забезпечення під мікроконтролери, а також розробкою окремих механізмів що тягне за собою сумісність з іншими пристроями. Як наслідок це буде значно впливати на на ціну кінцевого продукту, а саме за рахунок зниження ціни пристрої IoT здобули таку популярність.

1.3 Вразливості та атаки на пристрої IoT

Так як розробники зупиняються на архітектурі коли кожен пристрій є самостійним і має вихід до мережі Інтернет у першу чергу цікавий зловмисникам для організації ботнетів. Відсутність будь-якої перевірки оновлень на автентичність дозволяє повністю захоплювати контроль над пристроєм. Наочним

прикладом є ботнет Mirai. Mirai - ботнет та хробак який зламує методом перебору пар логін/пароль за замовчуванням та захоплює пристрій. У 2016 році такий ботнет успішно здійснив [3] DDoS атаку на інтернет канал Ліберії пропускною здатністю 5.1 ТБ/с.

Дослідження безпеки пристроїв IoT компанією Hewlett-Packard у 2014му році [4] показало:

- 70 відсотків протестованих пристроїв використовують незахищені протоколи передачі даних
- 60 відсотків з них не шифрують пакети оновлень при завантаженні
- 90 відсотків пристроїв збирають персональну інформацію.

І можна висунути теорію що при збільшенні апаратних ресурсів пристрою та здешевлення виробництва останнього відсоток пристроїв які використовують шифрування повинен збільшитися. Попри це за результатом звіту компанії Unit42 [5] за 2020 рік кількість IoT пристроїв, на кінець 2019 року становила 4.8 мільярдів пристроїв, що являє собою 30 відсотків від усіх пристроїв мережі Інтернет та 98 відсотків усього IoT трафіку є незашифрованим та передається у відкритому вигляді. Це підтверджує гіпотезу що апаратна обмеженість, складність розробки програмного забезпечення під існуючі платформи та забезпечення зворотної сумісності примушує розробників нехтувати механізмами безпеки.

1.4 OWASP Internet of Things Project

Open Web Application Security Project (OWASP) – це відкритий проект для забезпечення безпеки веб-застосунків. Команда OWASP випускає рекомендації спираючись на технічний рівень та людський фактор. Одним із напрямків організації є OWASP Top-10. Ідея цього документу є висвітлення основних проблем безпеки того чи іншого продукту. У 2018 році OWASP випустили Top-10 і для IoT області [6].

У документі можна виділити основні проблеми IoT:

1. Слабкі, вшиті або паролі які легко вгадуються

Пристрої IoT зі слабкими паролями за замовчуванням схильні до кібератак. Виробники пристроїв IoT повинні звертати увагу на налаштування пароля при першому запуску пристрою. Часто трапляється так, що або пристрій не дозволяє користувачам змінювати пароль за замовчуванням, або користувачі вважають за краще не змінювати його, навіть якщо вони можуть. Більш того, успішна спроба отримати несанкціонований доступ до одного пристрою робить інші в системі уразливими, оскільки пристрої IoT часто використовують одні й ті ж паролі за замовчуванням.

2. Незахищені мережеві сервіси

Мережеві служби, запущені на пристрої, можуть становити загрозу безпеці та цілісності системи. Коли вони доступні в Інтернеті, вони відкривають шлях для несанкціонованого віддаленого доступу і витоку даних. Зловмисники можуть успішно поставити під загрозу безпеку кінцевої точки IoT, скориставшись недоліками, присутніми в моделі мережевої комунікації.

3. Незахищені інтерфейси екосистеми пристроїв

Існує кілька інтерфейсів, таких як веб-інтерфейс, серверний API, хмара та мобільний інтерфейс, які забезпечують чітку взаємодію користувача з пристроєм. Однак відсутність належної аутентифікації, слабке шифрування і недостатня фільтрація даних можуть негативно позначитися на безпеці пристроїв IoT.

4. Відсутність безпечних механізмів оновлення пристроїв

Відсутність перевірки прошивки, незашифрованому передача даних, відсутність ЕЦП, відсутність механізмів захисту від відкату до попередніх версій, відсутність повідомлень про оновлення безпеки є причинами порушення безпеки IoT-пристроїв.

5. Використання незахищених або застарілих компонентів

Це має на увазі використання обладнання або програмного забезпечення сторонніх виробників та/або такого що являється не актуальним, яке пов'язане з ризиками і загрожує безпеці всієї системи. На промисловий IoT особливо впливають системи, які складно оновлювати і підтримувати. Такі уразливості

можуть бути використані для запуску атаки і порушення нормальної роботи системи.

6. Недостатній захист конфіденційності

Для коректної роботи пристроїв IoT може знадобитися зберігати конфіденційну інформацію користувачів. Однак ці пристрої часто не можуть забезпечити безпечне сховище даних, що призводить до витоку критично важливих даних при атаках. Крім пристроїв, атакам піддаються бази даних виробників.

7. Незахищена передача і зберігання даних

Відсутність шифрування при обробці конфіденційних даних під час передачі, обробки або в стані спокою - це можливість для хакерів вкрасти і розкрити дані. Шифрування необхідно скрізь, де відбувається передача даних.

8. Відсутність можливості керування пристроєм

Це відноситься до нездатності ефективно захистити всі пристрої в мережі. Це піддає систему численним загрозам. Незалежно від кількості задіяних пристроїв або їх розміру, кожне з них має бути захищене від витоку даних.

9. Небезпечні за замовчуванням налаштування пристроїв

Існуючі вразливості у налаштуваннях за замовчуванням наражають систему на безліч проблем з безпекою. Це можуть бути фіксовані та/або вшиті паролі, нездатність стежити за оновленнями безпеки і наявність застарілих компонентів.

10. Відсутність фізичного захисту

Відсутність фізичного захисту може легко допомогти хакерам отримати віддалений контроль над системою. Якщо не видалити фізичні порти або механізми налагодження система може піддатися атакам через відсутність фізичного захисту.

У цій роботі будуть розглядатися проблеми які притаманні платформам на яких будуються популярні пристрої IoT. В основному це механізми які не залежать від людського фактору, наприклад як перший пункт в Top-10.

Висновки до розділу 1

Розглянуто розвиток IoT пристроїв та причини які призвели до появи архітектури клієнт-сервер та проблематику такого рішення. Розглянуті популярні вектори атак на пристрої IoT та причини їх виникнення.

2 ОГЛЯД ТА АНАЛІЗ LINUX ДЛЯ ВИКОРИСТАННЯ В IOT ПРИСТРОЯХ

2.1 Переваги використання Linux в IoT

Використання Linux змінює підхід до вирішення задачі, тому що пристрій працює під контролем операційної системи, яка містить в собі програмну платформу та логічне розділення апаратних модулів та програм. Linux довела свою популярність у багатьох областях вбудованих обчислень завдяки високому рівню настройки і гнучкості, а також різноманітної апаратної підтримки. Ядро Linux, хоч і є монолітним, але має модулі ядра, що означає, що розробник або інженер може обирати драйвери і програмне забезпечення високого рівня, необхідне для конкретної системи. Підтримка великої кількості різних мікропроцесорних архітектур також є важливою перевагою Linux, оскільки у вбудованих системах може використовуватися мікропроцесор, який сильно відрізняється від тих, які є в ПК. Як проект програмного забезпечення з відкритим вихідним кодом, Linux також може використовуватися без обмежень і ліцензійних відрахувань, які можуть бути присутніми в комерційних пропозиціях. Крім модульної архітектури ще одним плюсом є величезна база готових програм, пакетів, драйверів та фреймворків, більшість з яких не мають програмних помилок та перевірені часом. Це значно спрощує пошук фахівців які вміють працювати з такими компонентами що в свою чергу пришвидшує розробку таких пристроїв та зменшує їх ціну.

2.2 Програмні можливості Linux в задачах взаємодій у системах IoT

Відмінність між пристроями розробленими на мікроконтролерах та пристроями з Linux є те, що у першому випадку робота пристрою є повністю детермінована, а Linux в свою чергу надає свою політику безпеки яка перевірялась та вдосконалювалася роками.

2.2.1 Журналювання

Одним з найбільш важливих аспектів в управлінні будь-якою системою є контроль за системними подіями. Linux пропонує незвичайний метод журналювання, а також дозволяє конфігурувати складові частини журналів. Журнали генеруються у текстовому форматі тому не потрібно ніяких додаткових інструментів для їх аналізу. Журнали генеруються програмою syslog в основний файл /var/log/messages. Налаштування журналу дуже гнучке, налаштувати можливо на будь-яку дію в системі. Також є окремі журнали такі, як /var/log/lastlog, /var/log/wtmp, /var/log/btmp, які містять інформацію про останній вхід користувача в систему, про всіх вдалих входах користувачів в систему і про всіх невдалих входах користувачів в систему відповідно. Журналювання є дуже важливим компонентом будь-якої інформаційної системи, оскільки надає повну історію усіх подій, яка у свою чергу допомагає розслідувати кіберінциденти, а також аналізувати причину відмови пристрою або програми. Таке журналювання цілком відповідає потребам систем на базі пристроїв IoT.

2.2.2 Цілісність компонентів ОС та модулів

Цілісність інформації є невід'ємною складовою у проектуванні захищеного пристрою. Інформація повинна бути захищена від навмисного, несанкціонованого або випадкового зміни в порівнянні з початковим станом, а також від будь-яких спотворень в процесі зберігання, передачі або обробки. Linux не має вбудованих механізмів забезпечення цілісності інформації, проте є досить великий спектр програм для забезпечення цілісності: strace, Samhain, DaemonFS, aide, OSSEC та інші.

Слід виділити Tripwire та OSSEC так, як вони являються хостовою системою виявлення вторгнень. Метою хостової IDS є стеження за всіма подіями, що відбуваються в комп'ютерній системі і перевірка їх на відповідність моделі безпеки. Хостова IDS веде спостереження за поточним станом системи, за інформацією, що

зберігається, як в оперативній пам'яті, так і в файловій системі, за даними системних логів і перевіряє, наскільки цей стан відповідає «нормальному». Тобто така система унеможливлює безконтрольну зміну файлів, що для пристроїв IoT є дуже важливим.

Ще одним важливим компонентом є AppArmor. AppArmor – утиліта яка реалізує попереджувальний захис за допомогою політик безпеки, або вони також називаються профілі безпеки які регулюють доступ інших програм до системних ресурсів . У AppArmor містить набір вбудованих профілів, а також інструменти статичного аналізу та інструменти, засновані на навчанні, що дозволяють прискорити і спростити побудову нових профілів. Це рішення реалізує значну частину функціоналу SELinux, але набагато простіше в налаштуванні і експлуатації. Так само, як і SELinux AppArmor є реалізацією системи Mandatory Access Control (MAC), заснованої на архітектурі Linux Security Modules (LSM). Модель безпеки Apparmor полягає в прив'язці атрибутів контролю доступу не до користувачів, а до програм. AppArmor забезпечує ізоляцію за допомогою профілів, що завантажуються в ядро, як правило, при завантаженні. Оскільки AppArmor контролює саме програми та системні ресурси, а не користувачів, це ідеально підходить для пристроїв IoT через те, що контроль над користувачами в таких пристроях потрібен не так часто як контроль над програмами.

2.2.3 Шифрування ОС

На відміну від деяких інших операційних систем, в Linux є багато засобів для криптографічного захисту інформації - від шифрування поштових листувань до шифрування файлів і блокових пристроїв. Нас цікавить саме шифрування на рівні файлової системи, файлів і блокових пристроїв. Для початку варто розібратися, в чому різниця.

Шифрування на рівні файлових систем передбачає наявність “прошарку” між основною файловою системою якщо, звичайно, файлова система сама по собі не підтримує шифрування і користувачем. Перевага даного типу шифрування - те, що

ключі для всіх користувачів різні. Недолік же - якщо включити шифрування імен файлів, довжина допустимого імені зменшиться, крім того, користувач може зберегти файл в інше місце на диску, що автоматично нівелює користь до нуля. І ще одне але - навіть якщо включено шифрування імен, тимчасові мітки залишаються колишніми.

Шифрування блокових пристроїв відбувається на нижчому рівні, під файловою системою. При цьому сама файлова система не знає, що вона знаходиться на зашифрованому томі. Переваги у даного способу протилежні недолікам попереднього. Недолік же полягає в тому, що доведеться кожного разу при завантаженні/монтуванні вводити пароль.

Найбільш популярною програмою шифрування блокових пристроїв є cryptsetup яка використовує модуль ядра dm-crypt як обчислювальну машину для дискового шифрування і працює за специфікацією Linux Unified Key Setup.

Відповідно для шифрування на рівні ФС використовують eCryptfs. eCryptfs - це криптографічний файлова система Linux. Вона зберігає криптографічні метадані для кожного файлу в окремому файлі, таким чином, що файли можна копіювати між комп'ютерами. Файл буде успішно розшифрований, якщо у вас є ключ. Таке рішення широко використовується для реалізації зашифрованою домашньої директорії, наприклад, в Ubuntu.

2.2.4 Шифрування трафіку

Однією із найбільших проблем в IoT є шифрування трафіку так, як на це у апаратної платформи не вистачає ресурсів. У Linux є декілька підходів до реалізації шифрування трафіку.

Якщо використовуються веб-додатки: які базуються, наприклад, на REST і весь трафік передається HTTP запитами то шифрування реалізують за допомогою HTTPS. Але часто в пристроях IoT використовуються одночасно декілька різних протоколів таких, як MQTT, HTTP, DDS, і не всі вони у собі містять механізми

шифрування. Тому такий трафік спрямовують через VPN і переносять задачі шифрування та аутентифікації на VPN протокол.

Окремо можна виділити SSH-тунелювання, як метод, який ідеологічно схожий на VPN, але архітектурно відрізняється. SSH-тунель – це мережевий тунель, який створений SSH-з'єднанням та використовується для шифрування даних всередині даного тунелю. Використовується для того, щоб убезпечити передачу даних в Інтернеті. Великою перевагоє SSH-тунелю є те що він шифрує трафік будь-яких протоколів у незалежності від останніх.

2.2.5 Підпис оновлень

Дуже важливим фактором є перевірка оновлень на автентичність. Для реалізації перевірки оновлень на автентичність виробник ПО накладає ЕЦП на пакет оновлення. У свою чергу в Linux є спеціальна програма під назвою GnuPG, для перевірки ЕЦП на автентичність.

У зв'язку вище перерахованими особливостями великі компанії почали випускати спеціальні дистрибутиви та впроваджувати такі розробки у свої продукти. Наочним прикладом є дистрибутив “Ubuntu for the Internet of Things” [7], де розробникам ПО вдалося успішно вирішити проблеми безпеки, простоти розробки та простоти впровадження. Основними ідеями дистрибутива є:

- Використання відкритого програмного забезпечення;
- Підтримка дистрибутиву патчами безпеки;
- Виділення для кожного додатку власного середовища;
- Шифрування ;
- Підтримка різних архітектур.

Завдяки такому підходу компанії Canonical вдалося створити новий підхід до проектування пристроїв IoT, спростивши складність проектування та вивести безпеку таких пристроїв на новий рівень.

2.3 Огляд основних підсистем управління транспортом та запропоновані схеми їх захисту

Причинами поширеності загроз для сучасних транспортних засобів є наступні фактори:

1. Розробляється ПЗ, відкрите до зовнішніх взаємодій (в тч з соцмережами);
2. Універсальні інтерфейси, які можуть бути використані в тому числі і зловмисником;
3. Наявність розважальних вразливих функцій та комунікаційних пристроїв (адаптери MP3 та iPod та порти USB). Мультимедійний інтерфейс (поки!) відокремлений від керуючого модуля;
4. Використання типових рішень з ПЗ та моделей (MBD) для автоматичної генерації коду без врахування безпеки.

Розглянемо основні атаки, які притаманні сучасним системам пристроїв інтернету речей, які забезпечують функціональність смарт-транспорту (зокрема, автомобілів).

- атаки на CAN шину, внаслідок широкомовного розповсюдження і незахищеності протоколів, які тут використовуються;
- атаки під час діагностичних робіт (сканування при прямому доступі до CAN, зараження вірусами через порт OBD II, перепрограмування функцій внаслідок несанкціонованого доступу до шини, використання уразливих PassThru пристроїв (типу TIS, VCM, HDS та ін.))
- атаки через розважальну інфраструктуру автомобіля.
- атаки через BlueTooth систему. Дослідники свідчать про наявність вразливостей у вихідних текстах Linux-подібної системи ОС ECU, зловмисники використовують сканери Bluetooth та ін.
- атаки через стільниковий зв’язок.
- атаки на TPMS.
- атаки на PKES.

- атаки на Wi-Fi функції, які підтримуються автомобілем, із використанням стандартних уразливостей протоколів безпровідного зв’язку.

У зв’язку із великою кількістю наявних атак та уразливостей виникає думка про те, що із розвитком розумного транспорту потрібно переходити на інші принципи управління автомобілем, і, особливо, безпілотним транспортом, розвиток якого є перспективою найближчого майбутнього.

Зокрема, в організації взаємодій смарт-транспорту беруть участь такі системи:

1) Розширена система управління та безпеки транспортних засобів (AVCSS);

AVCSS засновані на датчиках, встановлених у транспортних засобах. Вони представляють водіям візуальні сповіщення та інформацію про небезпечні ситуації. Автомобілі, обладнані AVCSS, можна вважати автономним транспортним засобом рівня 1. Стандарт SAE J3016_201401 визначає 5 рівнів автономних транспортних засобів, де рівень 0 - це транспортний засіб без автоматизації, а рівень 5 повністю автоматизований, здатний керувати автомобілем за будь-яких дорожніх та екологічних умов без водія людини.

2) Розширена система громадського транспорту (APTS).

APTS надає громадянам доступ до інформації про автобуси, включаючи наявність місця, місце розташування та передбачуваний час прибуття. Дозволяє користуватися мобільними квитками з використанням Near Field Communications (NFC) для оплати різних видів транспорту. Система також може приймати динамічні рішення - наприклад, затримувати автобуси, які курсують достроково.

3) Управління комерційним транспортом (CVO). Система забезпечує автоматичний моніторинг транспортних засобів, включаючи автобуси, швидку допомогу, вантажні автомобілі та ін.

4) системи управління парком. Дозволяє керувати комерційним транспортом, із використанням GPS-відстежуючих пристроїв. Використовуючи зібрані дані, організація може стежити за своїм автопарком (чи судами) для зниження витрат.

Також:

- відстеження витрат палива; відповідність водія маршрутам та протоколам;

- Аналіз експлуатаційних витрат;
- Забезпечення ефективної логістики;
- Адміністрація вантажу;
- Адміністративні процеси комерційних транспортних засобів;
- Зважування в русі (WIM);
- Безпека на дорозі;
- Моніторинг безпеки на борту;
- Планування небезпечних матеріалів та реагування на інциденти.

Таким чином, загрози спрямовані у цих системах, стосуються конфіденційності, доступності та цілісності. В свою чергу, рішення кіберзахисту стосуються таких основних можливостей та обмежень:

1. Можливість перезавантажити чи блокувати пристрій вручну (бортовий комп’ютер, локальну систему керування маршрутом) у випадку некоректної поведінки;
2. Обмеження на поведінку у крайніх режимах та виключних ситуаціях на низькому рівні;
3. Захист застосунків та протоколів, які виконують управління та взаємодію із бортовими системами транспорта.
4. Захист баз даних транспортної системи від НСД до даних.

Найбільшою проблемою у системах такого класу є забезпечення конфіденційності інформації, яка збирається та передається по незахищених каналах. Забезпечити конфіденційність запропоновано за допомогою використання OpenVPN який містить у собі можливість шифрування даних за допомогою криптостійких алгоритмів. Це водночас вирішує проблему приховання протоколів та захист від MITM, бо дані передаються повністю зашифрованими. Проблему цілісності вирішує OpenSSL у складі OpenVPN, оскільки останній перевіряє цілісність пакетів. Також у деяких випадках потрібно забезпечити конфіденційність для тих даних, які зберігаються локально на пристрої та даних які підготовлюються до відправки. Це можна реалізувати за допомогою або окремого

шифрованого розділу на ПЗП, вбудованого в пристрій або шифруванням всього ПЗП. На рисунку 2.1 зображена архітектура такого рішення.

Рисунок 2.1 – Архітектура забезпечення конфіденційності

Не менш важливою проблемою є перевірка на цілісність та автентичність пакетів оновлень для уникнення випадків шкідливого модифікування програм. Таку перевірку запропоновано реалізувати за допомогою програми GnuPG яка у свою чергу реалізує перевірку автентичності за допомогою ЕЦП. Цілісність перевіряється за допомогою контрольної суми. Виробник ПЗ накладає ЕЦП на пакет оновлень, а система спочатку перевіряє цілісність пакету і потім його автентичність. Ниже приведена архітектурна схема такої перевірки.

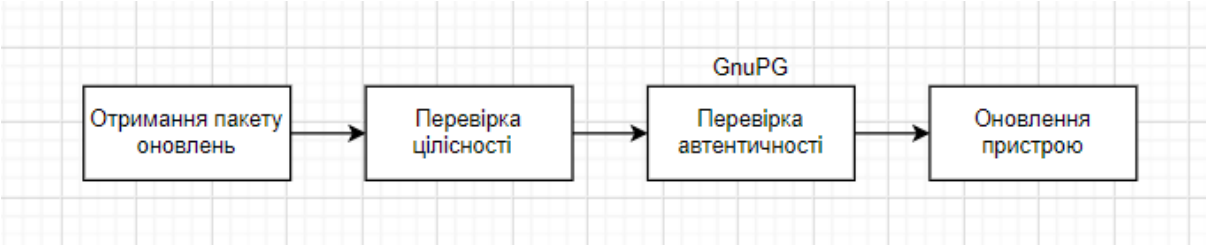

Рисунок 2.2 – Архітектура забезпечення автентичності

Висновки до розділу 2

Запропоновано обґрунтоване рішення із запровадження захисту на базі Linux до складу систем, які реалізують взаємодії між пристроями IoT та відповідними центрами оновлень та/або керування, зокрема систем управління інтелектуальним транспортом. Наведено переваги використання Linux-орієнтованих пристроїв з

точки зору забезпечення конфіденційності, цілісності. При цьому доступність також буде зберігатись на потрібному рівні, оскільки розвиток апаратних засобів дозволяє запровадити відповідні бортові системи управління транспортом на основі ОС Linux без втрати показників швидкодії. Також віддалені сервери, які здійснюють взаємодію із розумним транспортним пристроєм, повинні базуватись на тих самих принципах, під управлінням ОС Linux.

3 ПОЛІТИКА БЕЗПЕКИ ІОТ ПРИСТРОЇВ ПІД КЕРУВАННЯМ LINUX ТА НАОЧНА ДЕМОНСТРАЦІЯ

3.1 Поведінкові правила для забезпечення цілісності, конфіденційності та доступності

3.1.1 Забезпечення конфіденційності

Так як у даній сфері конфіденційність є однією з найважливіших компонентів, тому для її реалізації можна ввести декілька базових правил.

- Будь-яка комунікація з керуючим сервером або будь-яким ресурсом у мережі Інтернет або локальній мережі, повинна бути зашифрована. Це може бути шифрування вбудоване в протокол та/або передача даних по завідомо безпечних каналах. Прикладами таких каналів є VPN з’єднання або SSH тунель до власних серверів.
- Вбудований запам'ятовуючий пристрій повинен повністю та надійно шифруватися. Якщо на повне шифрування усього диску не вистачає ресурсів то шифруватися повинні: будь-яка інформація яка є конфіденційною та/або містить персональну інформацію клієнта, файли конфігурації, журналів та оновлень.
- Користувацьким програмам за замовчуванням заборонено доступ до всього окрім ресурсів над якими вони оперують. Наприклад програма має права на запис тільки до своєї директорії та/або до пристрою, датчику.

Завдяки таким правилам та коректному налаштуванні відповідних програм на пристрої досягається конфіденційність(рисунок 3.1).

Рисунок 3.1 – Забезпечення конфіденційності

3.1.2 Забезпечення цілісності

Цілісність за значимістю є не менш важливою ніж конфіденційність. Цілісність має забезпечуватись як на рівні протоколів, так і на рівні ФС і файлів які отримуються та передаються. У ОС Linux цілісність даних на рівні ФС забезпечується за допомогою вбудованих в неї механізмів та/або іншими програмними засобами наприклад Tripwire. Цілісність на рівні протоколів забезпечується такими протоколами як OpenVPN, SSH та інші, які містять у собі механізми та забезпечують цілісність даних які вони передають. Цілісність файлів які передаються забезпечується порівнянням хеш-суми файлу до передачі та після.

Отже для можна ввести базові правила для забезпечення цілісності:

- Передача даних повинна здійснюватись за допомогою протоколів які у собі містять механізми для забезпечення цілісності даних які передаються.
- Файлова система повинна регулярно перевірятися на наявність помилок.
- Перевірка цілісності файлів які передаються за допомогою хеш-сум

3.1.3 Забезпечення доступності

У даній області забезпечення доступності з боку пристрою під управлінням ОС Linux є моніторинг каналу передачі даних та аналіз його завантаженості. Так як об’єми даних між IoT пристроями невеликі то останні повинні контролювати різке, безпричинне збільшення трафіку та повідомлення про даний інцидент резервні сервери.

3.1.4 Забезпечення автентичності даних

Будь-які дані або пакети оновлення які впливають на детерміновану роботу пристрою повинні бути підписаними за допомогою ЕЦП для перевірки автентичності.

3.1.5 Журналювання

Для забезпечення спостережності та спрощення пошуку помилок журналювання можливо реалізувати за допомогою вбудованої syslog яку можна налаштувати на журналювання будь-яких дій у системі, а також самих журналів ОС Linux. Для даної системи у першу чергу потрібно журналювати:

- інформацію про авторизацію користувачів, включаючи вдалі і невдалі спроби входу в систему, а також задіяні механізми аутентифікації. Записується системою у окремий файл /var/log/auth.log.
- помилки та системні критичні повідомлення, у ядрі лінукс мітки KERN_ERR та вище.
- звіти вбудованого в систему аудиту

3.2 Розробка політик безпеки для взаємодій пристроїв та підсистем транспортної системи

Таблиця 3.1 – Внутрішня політика безпеки пристроїв

Тип пристрою	Послуги безпеки			
	Конфіденційність	Цілісність	Доступність	Спостережність
Світлофор	-	+ Самотестування основних функцій при перезапуску. контроль перепрошивання	+ Можливість контролю з боку керуючої системи та подібних пристроїв для синхронізації	+ Надсилання сигналів про збій до керуючої системи, які повинні логуватись у керуючій системі та/або хмарі

Продовження таблиці 3.1

Датчик дорожнього руху	-	+ Захист від сторонніх втручань (на апаратному рівні)	+ Можливість збору інформації з боку керуючої системи (доступність оцінюється методами надсилання пакетів про нормальне функціонування)	+ Надсилання сигналів про нормальне функціонування, які повинні логуватись у керуючій системі та/або хмарі
Камера відеонагляду	+ (надсилання зібраних даних - номери порушників тощо в шифрованому виді)	+ Самотестування із встановленим періодом та при перезавантаженні, контроль при перепрошиванні	+ Можливість контролю з боку керуючої системи та подібних пристроїв для синхронізації	+ Ведення внутрішнього журналу, резервна копія журналу у хмарі

Кінець таблиці 3.1

Транспортний засіб громадського транспорту (з водієм)	Конфіденційність ключів для спілкування, які зберігаються на пристрої	+ Самотестування із встановленим періодом та при перезавантаженні, контроль цілісності при перепрошиванні	+ Можливість контролю з боку керуючої системи та подібних пристроїв для синхронізації	+ Ведення внутрішнього журналу, резервна копія журналу у хмарі
Транспортний засіб без водія (безпілотний, напр. трамвай)	Конфіденційність ключів для спілкування, які зберігаються на пристрої	+ Самотестування із встановленим періодом та при перезавантаженні, захист від сторонніх втручань на апаратному рівні, контроль цілісності при перепрошиванні	+ Можливість контролю з боку керуючої системи та подібних пристроїв для синхронізації	+ Ведення внутрішнього журналу, резервна копія журналу у хмарі

Таблиця 3.2 – Зовнішня політика безпеки пристроїв

Тип пристрою	З ким спілкується шляхом безпровідного зв’язку, яка властивість інформації має забезпечуватись при обміні (Конфіденційність К, Цілісність Ц+Автентифікація А)						
	Хмара	Центр управління (сервери)	Світлофор	Безпілотний пристрій	Транспортний пристрій	Камера	Датчик дорожнього руху
Світлофор	Логи (Ц+А)	Керуючі сигнали (Ц+А)	Синхронізуюча інформація (Ц)	Довідкова інформація (Ц+А)	Довідкова інформація (Ц)	-	-
Датчик дорожнього руху	Інформація спостережності (Ц+А)	Довідкова /керуюча інформація (Ц+А)	Синхронізуюча інформація (Ц)	Довідкова інформація (Ц+А)	Довідкова інформація (Ц)	-	Синхронізуюча (Ц+А)

Кінець таблиці 3.2

Камера відеонагляду	Інформація про порушників та дорожню ситуацію, копія логів (К, Ц+А)	Керуючі сигнали/ інформація про порушників (К, Ц+А)	-	-	-	Синхронізуюча інформація (Ц)	-
Транспортний засіб громадського транспорту (з водієм)	Логи (Ц+А)	Керуючі сигнали/ логи (Ц+А)	-	-	Синхронізуюча інформація (Ц+А)	-	-
Транспортний засіб без водія (безпілотний, напр. трамвай)	Логи (Ц+А)	Керуючі сигнали/ логи, інформація про місцеположення (Ц+А+К)	-	Синхронізуюча інф., інформація про місцеположення (Ц+А+К)	-	-	Синхронізуюча інформація, інформація про місцеположення

Було проаналізовано існуючі можливості ОС Linux і визначено, що політики безпеки, представлені у табл. 3.1 , 3.2 можуть бути забезпечені засобами ОС та засобами відповідних протоколів комунікації (OpenVPN).

Також проаналізована апаратна платформа і здійснене порівняння пристроїв під керуванням Linux та апаратних контролерів у таблицці 3.3.

Таблиця 3.3 Порівняльний аналіз можливостей ОС Linux та апаратних контролерів, що реалізують одні й ті самі функції

Показник	ОС Linux, Експертний бал по шкалі від 1 до 5	Апаратний контролер, Експертний бал по шкалі від 1 до 5	Вага і-го показника wi (сума всіх ваг=1)
Забезпечення журналювання, p1	5	3	0,1
Забезпечення самотестування та контролю цілісності,p2	5	2	0,2
Забезпечення/підтримка шифрування, конфіденційності, p3	5	3	0,2
Забезпечення/підтримка швидкодії, p4	3	5	0,2

Кінець таблиці 3.3

Забезпечення/підтримка контролю автентичності та цілісності, в тому числі при перепрошиванні (Засоби ЕЦП), p5	5	1	0,3
Сума балів	23	14	1
Зважена сума	4.6	2,6	

3.3 Практичне впровадження

Наочною реалізацію вище перерахованих правил було здійснено на одноплатному комп’ютері OrangePi під управлінням ОС ARMbian.

3.3.1 Конфіденційність

У даному експерименті було вирішено шифрувати канал передачі даних за допомогою протоколу OpenVPN. Налаштована VPN point-to-Site з режимом аутентифікації по сертифікату. Такий спосіб гарантує надійний спосіб шифрування, аутентифікації та цілісності переданих даних. У експерименті підключення до мережі Інтернет було здійснено за допомогою 3G-модему, стільникового зв'язку та утиліти wvdial.

Конфігурація серверу OpenVPN(Рисунок 3.2):

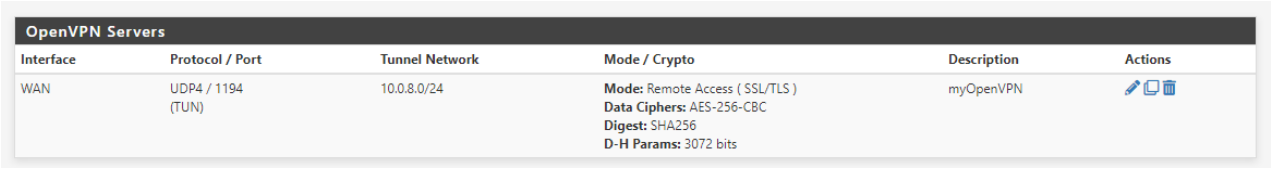

Рисунок 3.2 – Налаштування серверу

Конфігурація клієнта:

Wvdial (рисунок 3.3):

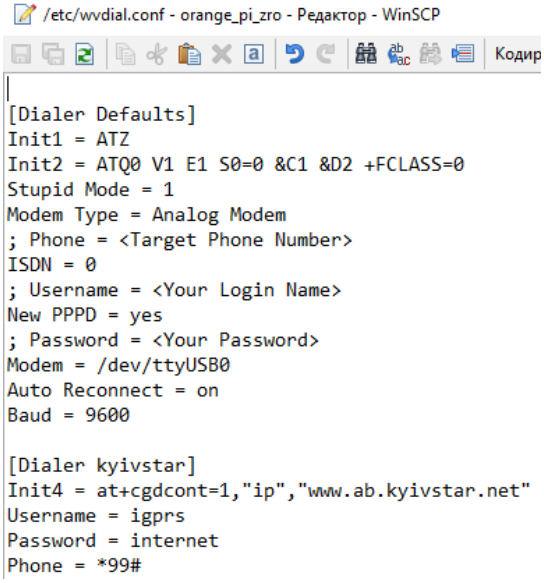

Рисунок 3.3 – Налаштування wvdial

Підключення до серверу за допомогою OpenVPN(рисунок 3.4):

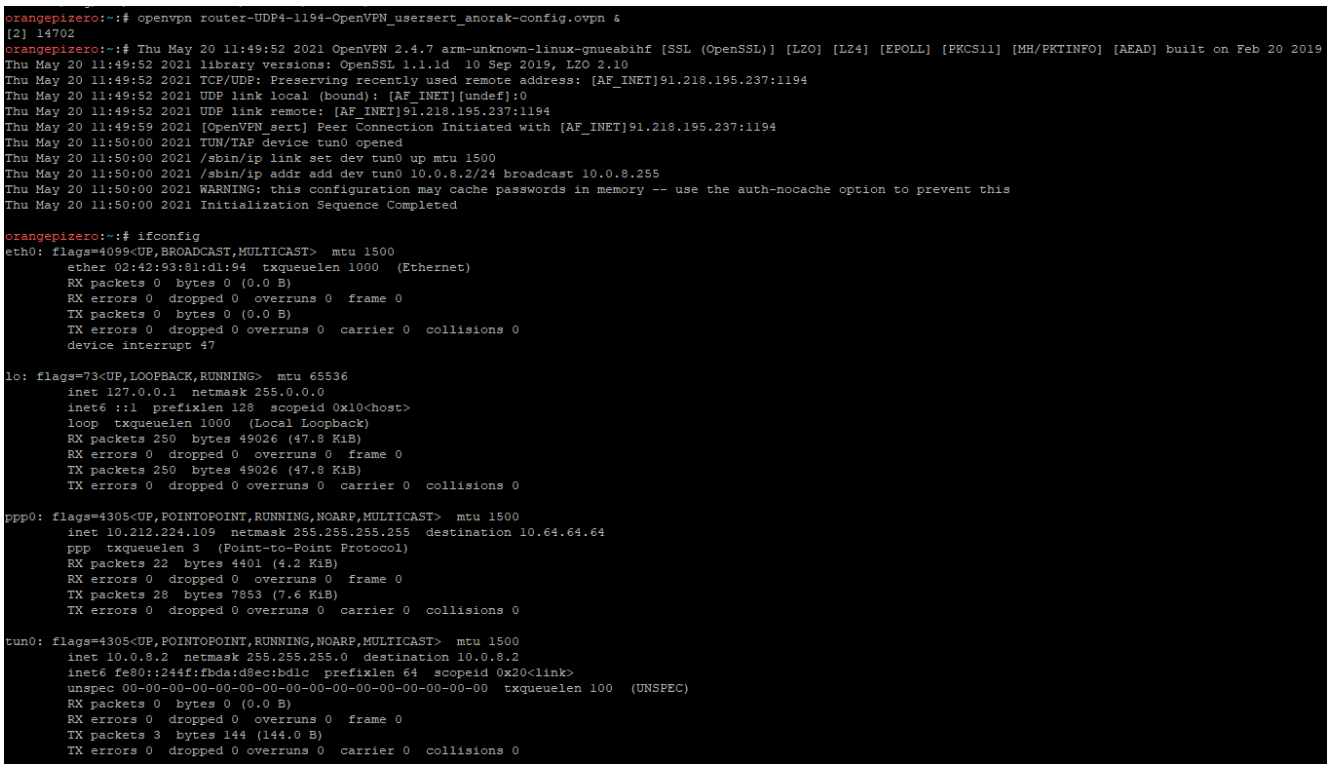

Рисунок 3.4 – Процес підключення OpenVPN

На сервері відразу логується дана подія(рисунок 3.5):

Last 1068 OpenVPN Log Entries. (Maximum 2000)

Time	Process	PID	Message
May 20 11:57:16	openvpn	97342	OpenVPN_usersert_anorak/46.211.93.65:14710 MULTI_sva: pool returned IPv4=10.0.8.2, IPv6=(Not enabled)
May 20 11:57:16	openvpn	97342	46.211.93.65:14710 [OpenVPN_usersert_anorak] Peer Connection Initiated with [AF_INET]46.211.93.65:14710
May 20 11:57:16	openvpn	97342	46.211.93.65:14710 peer info: IV_TCPNL=1
May 20 11:57:16	openvpn	97342	46.211.93.65:14710 peer info: IV_COMP_STUBv2=1
May 20 11:57:16	openvpn	97342	46.211.93.65:14710 peer info: IV_COMP_STUB=1
May 20 11:57:16	openvpn	97342	46.211.93.65:14710 peer info: IV_LZO=1
May 20 11:57:16	openvpn	97342	46.211.93.65:14710 peer info: IV_LZ4v2=1
May 20 11:57:16	openvpn	97342	46.211.93.65:14710 peer info: IV_LZ4=1
May 20 11:57:16	openvpn	97342	46.211.93.65:14710 peer info: IV_PROTO=2
May 20 11:57:16	openvpn	97342	46.211.93.65:14710 peer info: IV_PLAT=linux
May 20 11:57:16	openvpn	97342	46.211.93.65:14710 peer info: IV_VER=2.4.7

Рисунок 3.5 – Запис логів

Шифрування файлів GPG(рисунок 3.6):

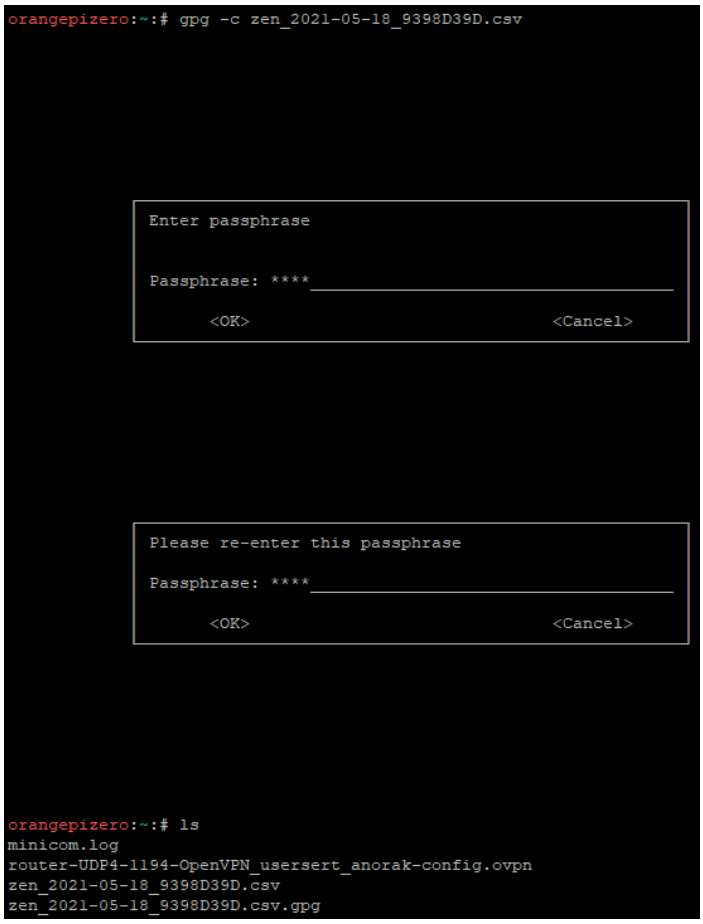

Рисунок 3.6 – Робота програми GPG

Після роботи програми отримуємо файл з розширенням .gpg зашифрований за допомогою симетричного алгоритму AES-256.

Відповідно розшифрування файлу (рисунок 3.7):

```
orangepizero:~:# gpg zen_2021-05-18_9398D39D.csv.gpg
gpg: WARNING: no command supplied.  Trying to guess what you mean ...
gpg: AES256 encrypted data
gpg: encrypted with 1 passphrase
File 'zen_2021-05-18_9398D39D.csv' exists. Overwrite? (y/N) y
orangepizero:~:#
```

Рисунок 3.7 – Розшифрування файлу gpg

3.3.2 Автентичність

Програма GPG підтримує накладання ЕЦП на файли. Для цього потрібно створити ключ, за замовчуванням створюються ключ RSA-3072 (рисунок 3.8).

```
orangepizero:~:# gpg --list-keys
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2023-05-20
/root/.gnupg/pubring.kbx
------------------------
pub   rsa3072 2021-05-20 [SC] [expires: 2023-05-20]
      BCFC399ED58DA26790D74D90500AE03A7A426D0C
uid           [ultimate] iotkey
sub   rsa3072 2021-05-20 [E] [expires: 2023-05-20]
```

Рисунок 3.8 – Створення ключа шифрування

Далі за допомогою даного ключа підписуемо файл (рисунок 3.9):

```
orangepizero:~:# gpg -b cat.txt
orangepizero:~:# ls
cat.txt      router-UDP4-1194-OpenVPN_usersert_anorak-config.ovpn
cat.txt.sig  zen_2021-05-18_9398D39D.csv
minicom.log  zen_2021-05-18_9398D39D.csv.gpg
orangepizero:~:#
```

Рисунок 3.9 – Накладання ЕЦП

Тепер спробуємо змінити інформацію в файлі та спробуємо перевірити файл на автентичність(рисунок 3.10):

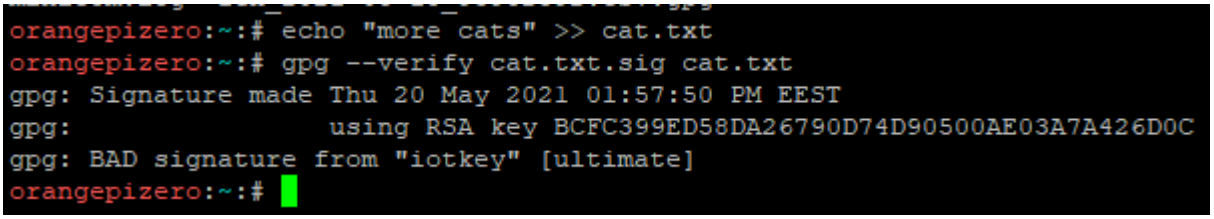

Рисунок 3.10 – Невдала перевірка файлу після зміни вмісту

Програма коректно перевірила ЕЦП та як і очікувалось, вивела що даний ЕЦП не належить цьому файлу.

3.3.3 Цілісність

Для перевірки ФС на помилки у Linux є стандартна утиліта fsck. У даному випадку доречно її використовувати при кожному запуску системи.

Налаштовується це наступним чином(рисунок 3.11):

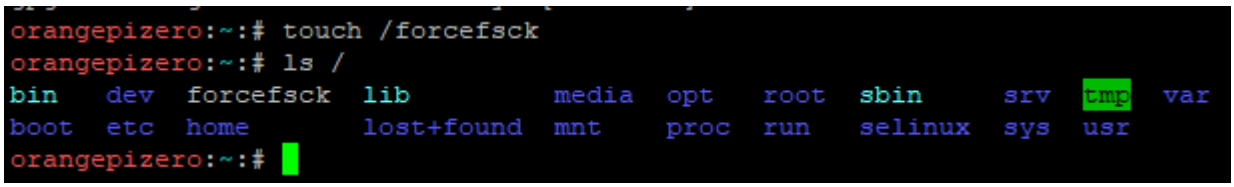

Рисунок 3.11 – Створення файлу forcefsck

Тепер після завантаження ядра ОС виконується перевірка ФС(рисунок 3.12):

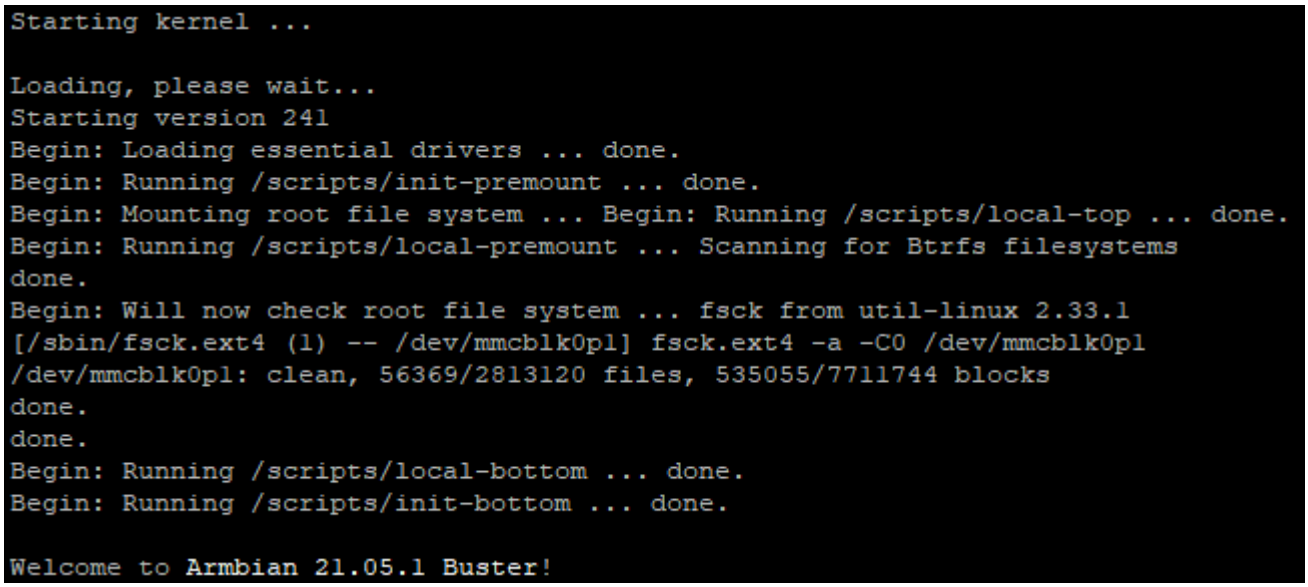

Рисунок 3.12 – Перевірка цілісності при запуску ОС

3.3.4 Журналювання

Налаштування серверу збирання логів syslog-ng(рисунок 3.13):

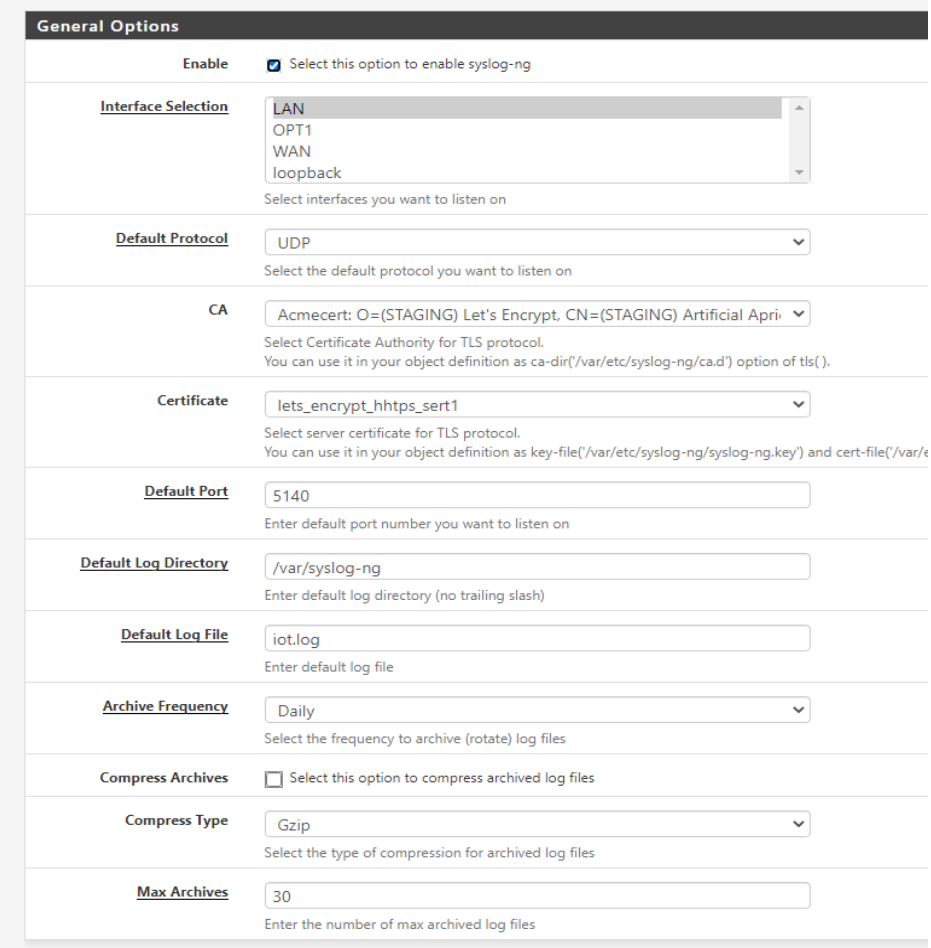

Рисунок 3.13 – Конфігурація syslog-ng

Налаштування клієнта на отримання логів на сервер(Рис. 3.14-3.15):

```
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
user.*                          -/var/log/user.log

#
# Logging for the mail system.  Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info                       -/var/log/mail.info
mail.warn                       -/var/log/mail.warn
mail.err                        /var/log/mail.err

#
# Some "catch-all" log files.
#
*.=debug;\
        auth,authpriv.none;\
        news.none;mail.none     -/var/log/debug
*.=info;*.=notice;*.=warn;\
        auth,authpriv.none;\
        cron,daemon.none;\
        mail,news.none          -/var/log/messages

#
# Emergencies are sent to everybody logged in.
#
*.emerg                         :omusrmsg:*

*.info;mail.none;authpriv.none;cron.none @192.168.1.1:5140
```

Рисунок 3.14 – Конфігурація клієнта

Showing 47 of 47 messages

May 20 19:20:53 192.168.2.108 rsyslogd: [origin software="rsyslogd" swVersion="8.1901.0" x-pid="2718" x-info="https://www.rsyslog.com"] start
May 20 19:20:53 192.168.2.108 rsyslogd: [origin software="rsyslogd" swVersion="8.1901.0" x-pid="608" x-info="https://www.rsyslog.com"] exiting on signal 15.
May 20 19:20:53 192.168.2.108 rsyslogd: imuxsock: Acquired UNIX socket '/run/systemd/journal/syslog' (fd 3) from systemd. [v8.1901.0]
May 20 19:20:53 192.168.2.108 systemd[1]: Started System Logging Service.
May 20 19:20:53 192.168.2.108 systemd[1]: Starting System Logging Service...
May 20 19:20:53 192.168.2.108 systemd[1]: Stopped System Logging Service.
May 20 19:20:53 192.168.2.108 systemd[1]: Stopping System Logging Service...
May 20 19:20:53 192.168.2.108 systemd[1]: rsyslog.service: Succeeded.
May 20 19:22:17 192.168.2.108 sshd[2731]: Accepted password for root from 192.168.1.252 port 11141 ssh2
May 20 19:22:17 192.168.2.108 systemd-logind[555]: New session 58 of user root.
May 20 19:22:17 192.168.2.108 systemd[1]: Started Session 58 of user root.
May 20 19:41:12 192.168.2.108 rngd[592]: stats: Entropy starvations: 0
May 20 19:41:12 192.168.2.108 rngd[592]: stats: FIPS 140-2 failures: 0
May 20 19:41:12 192.168.2.108 rngd[592]: stats: FIPS 140-2 successes: 10
May 20 19:41:12 192.168.2.108 rngd[592]: stats: FIPS 140-2(2001-10-10) Continuous run: 0

Рисунок 3.15 – Журнал syslog-ng

3.4 Перевірка швидкості взаємодій

Після налаштувань даної системи доречно перевірити наскільки введення додаткових перевірок вплине на швидкість обміну даними. Так як на швидкість взаємодій впливає тільки шифрування та час на шифрування і розшифрування, перевірка ЕЦП. У даному експерименті відлік часу здійснювалось за допомогою вбудованої утиліти time. У експерименті розміри файлу були взяті наближені до реальних умов, а саме: розмір файлу на передачу – 3 мегабайти, файлу на шифрування та ЕЦП – 50 мегабайт. В випадку без шифрування вважаємо що операція по розшифруванню або шифруванню займає 0 секунд часу виконання. Отримані результати висвітленні у таблиці 3.4.

Таблиця 3.4 – Час виконання команд

Тип задачі	Час витрачений на команду без налаштувань безпеки	Час витрачений на команду з налаштованими механізмами безпеки
Передача даних по незахищених каналах зв’яку	1 хвилина 23.03 секунд	1 хвилина 25.81 секунд
Розшифрування файлу на ПЗП	0 секунд	4.412 секунди
Перевірка ЕЦП	0 секунд	3.646 секунди
Накладання ЕЦП	0 секунд	4.745 секунди

Як результат можна зробити висновок що передача даних з налаштованим шифруванням потребує на 2-3 секунди більше що в умовах даного експерименту можна вважати погрішністю так, як використовувався модем з підтримкою лише третього покоління мобільного зв’язку. У випадках використання ЕЦП та шифрування файлів потребується 3.5-5 секунд для файлу в 50 мегабайт що для

сфери інтелектуального транспорту не являється критичним, беручи до уваги що використовувався один із примітивних процесорів на старій архітектурі та ключі шифрування для ЕЦП були 3072 біти, що для таких пристроїв є надлишковим.

Висновки розділу 3

В результаті проведеного аналізу було показано ефективність Linux-орієнтованих рішень для платформи пристроїв, які беруть участь у складі транспортної системи.

Розроблено політики безпеки для пристроїв різних видів.

Дані пристрої, якщо їхня функціональність заснована на лінукс-платформі, а не на контролерах чи мікроконтролерах, забезпечують всі вимоги, які висуваються до них політикою безпеки всередині пристрою та політикою безпеки по взаємодії з оточуючими пристроями. Функціональність пристрою при цьому не погіршиться і навпаки, стане ще більш гнучкою.

Було проведено експеримент, для якого розгорнуто всі необхідні засоби, запропоновано послідовність заходів, які необхідно виконати, щоби налаштувати на платформі Linux всі необхідні налаштування безпеки, які можуть знадобитись для кожного із розглянутих типів пристроїв.

Запропоновано спосіб безпечного обміну, який забезпечує необхідну в транспортній системі автентифікацію, конфіденційність, не порушуючи швидкодії.

ВИСНОВКИ

Дана робота присвячення дослідженню Linux платформи для використання її як головної ОС IoT пристроїв. Запропоновано обґрунтоване рішення із запровадження захисту на базі Linux до складу систем, які реалізують взаємодії між пристроями IoT та відповідними центрами оновлень та/або керування, зокрема систем управління інтелектуальним транспортом. Наведено переваги використання Linux-орієнтованих пристроїв з точки зору забезпечення конфіденційності, цілісності.

В результаті проведеного аналізу було показано ефективність Linux-орієнтованих рішень для платформи пристроїв, які беруть участь у складі транспортної системи.

Розроблено політики безпеки для пристроїв різних видів.

Дані пристрої, якщо їхня функціональність заснована на лінукс-платформі, а не на контролерах чи мікроконтролерах, забезпечують всі вимоги, які висуваються до них політикою безпеки всередині пристрою та політикою безпеки по взаємодії з оточуючими пристроями.

Було проведено експеримент, для якого розгорнуто всі необхідні засоби, запропоновано послідовність заходів, які необхідно виконати, щоби налаштувати на платформі Linux всі необхідні налаштування безпеки, які можуть знадобитись для кожного із розглянутих типів пристроїв.

Запропоновано спосіб безпечного обміну, який забезпечує необхідну в транспортній системі автентифікацію, конфіденційність, не порушуючи швидкодії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Черняк Л. Платформа Интернета вещей [Електронний ресурс] / Леонид Черняк. – 2012. – Режим доступу до ресурсу: https://www.osp.ru/os/2012/07/13017643.
2. Gershenfeld N. The Internet of Things [Електронний ресурс] / N. Gershenfeld, R. Krikorian, D. Cohen // Scientific American. – 2004. – Режим доступу до ресурсу: http://cba.mit.edu/docs/papers/04.10.i0.pdf.
3. Whittaker Z. Mirai botnet attackers are trying to knock an entire country offline [Електронний ресурс] / Zack Whittaker // Zero Day. – 2016. – Режим доступу до ресурсу: https://www.zdnet.com/article/mirai-botnet-attack-briefly-knocked-an-entire-country-offline/.
4. HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://www.hp.com/us-en/hp-news/press-release.html?id=1744676.
5. 2020 Unit 42 IoT Threat Report [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://iotbusinessnews.com/download/white-papers/UNIT42-IoT-Threat-Report.pdf.
6. OWASP Internet of Things (IoT) Project [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project.
7. Ubuntu for the Internet of Things [Електронний ресурс] – Режим доступу до ресурсу: https://ubuntu.com/internet-of-things.

ДОДАТОК А КОНФІГУРАЦІЯ OPENVPN СЕРВЕРУ

Файл config.ovpn:

```
dev ovpns1
verb 1
dev-type tun
dev-node /dev/tun1
writepid /var/run/openvpn_server1.pid
#user nobody
#group nobody
script-security 3
daemon
keepalive 10 60
ping-timer-rem
persist-tun
persist-key
proto udp4
auth SHA256
up /usr/local/sbin/ovpn-linkup
down /usr/local/sbin/ovpn-linkdown
learn-address "/usr/local/sbin/openvpn.learn-address.sh anorak.pp.ua"
local 91.218.195.237
tls-server
server 10.0.8.0 255.255.255.0
client-config-dir /var/etc/openvpn/server1/csc
tls-verify "/usr/local/sbin/ovpn_auth_verify tls 'OpenVPN_sert' 2"
lport 1194
management /var/etc/openvpn/server1/sock unix
max-clients 3
push "dhcp-option DNS 192.168.1.1"
```

push "dhcp-option DNS 1.1.1.1"
push "dhcp-option DNS 8.8.8.8"
push "redirect-gateway def1"
client-to-client
capath /var/etc/openvpn/server1/ca
cert /var/etc/openvpn/server1/cert
key /var/etc/openvpn/server1/key
dh /etc/dh-parameters.3072
tls-crypt /var/etc/openvpn/server1/tls-crypt
ncp-disable
cipher AES-256-CBC
allow-compression no
persist-remote-ip
float
topology subnet