Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню бакалавра

студента *Таратути Філіппа Євгеновича*

академічної групи *125-17-2*

спеціальності *125 Кібербезпека*

спеціалізації[1]

за освітньо-професійною програмою *Кібербезпека*

на тему *Розробка засобів захисту IoT-мережі на базі блокчейнів від DDoS-атак*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи				
розділів:				
спеціальний	ст. викл. Мєшков В.І.			
економічний	к.е.н., доц. Пілова Д.П.			
Рецензент				
Нормоконтролер				

Дніпро
2021

ЗАТВЕРДЖЕНО:
завідувач кафедри
безпеки інформації та телекомунікацій
_____ д.т.н., проф. Корнієнко В.І.

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу ступеня бакалавра

студенту *Таратуті Філіппу Євгеновичу* **Академічної групи** *125-17-2*
(прізвище ім'я по-батькові) (шифр)

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

на тему *Розробка засобів захисту IoT-мережі на базі блокчейнів від DDoS-атак*

затверджену наказом ректора НТУ «Дніпровська політехніка» від _____ № ____

Розділ	Зміст	Термін виконання
Розділ 1	Загальний аналіз IoT-мережі, DDoS-атаки та блокчейну	26.02.2021 29.03.2021
Розділ 2	Аналіз вразливостей IoT-мережі, аналіз реалізації DDoS-атак, модель порушника, запровадження блокчейн-технології до системи захисту мережі	1.04.2021 24.05.2021
Розділ 3	Розрахунки на реалізацію проекту	26.05.2021 14.06.2021

Завдання видано _____ Мєшков В.І.
(підпис керівника) (прізвище, ініціали)

Дата видачі: 08.01.2021р.

Дата подання до екзаменаційної комісії: 15.06.2021р.

Прийнято до виконання _____ Таратута Ф.Є.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 63 с., 9 рис., 1 табл., 5 додатків, 10 джерел.

Об’єкт дослідження: мережа Інтернет речей.

Мета роботи: аналіз вразливостей IoT-мережі, аналіз реалізації DDoS-атаки на мережу, розробка та запровадження системи захисту на основі блокчейну.

Методи розробки: аналіз, опис, проектування та розрахунки.

У першому розділі розглянуто архітектуру IoT-мереж, DDoS-атак, розглянуто основні принципи захисту мережі та проведено огляд блокчейн-технології.

У другому розділі виконано аналіз вразливостей типової IoT-мережі, розглянуті можливі варіанти реалізації DDoS-атаки, розроблено систему захисту на основі блокчейну.

У третьому розділі проведено розрахунок капітальних та експлуатаційних витрат на проектування та впровадження нового комплексу для захисту IoT-мережі від DDoS-атак на основі блокчейну.

Практичне значення та наукова новизна: запронована система захисту дасть можливість попереджувати DDoS-атаки на IoT-мережу.

ІНФОРМАЦІЙНА БЕЗПЕКА, ІОТ-МЕРЕЖА, ВРАЗЛИВІСТЬ, DDOS-АТАКА, БЛОКЧЕЙН

РЕФЕРАТ

Пояснительная записка: 63 с., 9 рис., 1 табл., 5 приложений, 10 источников.

Объект исследования: сеть Интернет вещей (IoT).

Цель: анализ уязвимостей IoT сети, анализ реализации DDoS-атаки на сеть, разработка и внедрение системы защиты на основе блокчейна.

Методы разработки: анализ, описание, проектирование и расчеты.

В первом разделе рассмотрены архитектура IoT-сетей, DDoS-атак, рассмотрены основные принципы защиты сети и проведен обзор блокчейн-технологии.

Во втором разделе выполнен анализ уязвимостей типичной IoT-сети, рассмотрены возможные варианты реализации DDoS-атаки, разработана система защиты на основе блокчейна.

В третьем разделе проведен расчет капитальных и эксплутационных затрат на проектирование и внедрение нового комплекса защиты для защиты IoT-сети от DDoS-атак.

Практическое значение и научная новизна: предложенная система защиты позволит предупреждать DDoS-атаки на IoT-сеть.

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ, IOT-СЕТЬ, УЯЗВИМОСТЬ, DDOS-АТАКА, БЛОКЧЕЙН

ABSTRACT

Explanatory note: 63 pages, 9 drawings, 1 table, 5 applications, 10 sources.

Object of study: Internet of Things network.

Purpose: analysis of IoT network vulnerabilities, analysis of the implementation of a DDoS attack on a network, development and implementation of a blockchain-based protection system.

Development methods: analysis, description, design and calculations.

The first section discusses the architecture of IoT networks, DDoS attacks, considers the basic principles of network protection and provides an overview of blockchain technology.

In the second section, an analysis of the vulnerabilities of a typical IoT network is carried out, possible options for implementing a DDoS attack are considered, and a blockchain-based protection system is developed.

In the third section, the calculation of capital and operating costs for the design and implementation of a new protection complex to protect IoT-network from DDoS-atacks was carried out.

Practical Value and Scientific novelty: proposed protection system will prevent DDoS-atacks on IoT-network.

INFORMATION SECURITY, IOT NETWORK, VULNERABILITY, DDOS ATTACK, BLOCKCHAIN

СПИСОК УМОВНИХ СКОРОЧЕНЬ

ABI – Application Binary Interface;

API – Applied Programming Interface;

DDoS – Distributed Denial of Service;

DRDoS – Distributed Reflection Denial of Service;

DNS – Domain Name System;

DSA – Digital Signature Algorithm;

EVM – Ethereum Virtual Machine;

GPS – Global Positioning System;

HTTP – HyperText Transfer Protocol;

ICMP – Internet Control Message Protocol;

IoT – Internet of Things;

IRC – Internet Relay Chat;

JSON RPC – JavaScript Object Notation Remote Procedure Call;

PoS – Proof of Stake;

PoW – Proof of Work;

RIPEMD – RACE Integrity Primitives Evaluation Message Digest;

RSA – Rivest Shamir Adleman;

SHA – Secure Hash Algorithm;

TCP/IP – Transmission Control Protocol/Internet Protocol;

UDP – User Datagram Protocol;

WSN – Wireless Sensor Network;

ПЗ – Програмне забезпечення.

ЗМІСТ

ВСТУП

Інтернет речей (IoT) – в цілому відноситься до інтеграції фізичних пристроїв, які можуть працювати, спрацьовувати та взаємодіяти автономно, оптимізуючи та надаючи нові послуги у широкому діапазоні областей. З огляду на останні розробки у виробництві пристроїв, IoT, як очікується, сприятиме поширенню ефективного управління ресурсами в таких додатках, як розумні електромережі, інтелектуальні простори, розумні міста, автоматизація галузі, охорона здоров'я тощо. На рисунку 1 проілюстровано, як велика кількість IoT-речей, таких як датчики та виконавчі механізми, взаємодіють із централізованою мережею через мережу зв'язку.

Станом вже на 2020 рік до мережі Інтернет було підключено більш ніж 50 мільйонів Інтернет-речей. Через великий об'єм мережі, а також важливість інформації, яку ці пристрої виробляють, безпека IoT-мережі посідає вагоме місце. Більшість таких пристроїв обмежені з точки зору ресурсів та обчислювальних можливостей, що погіршує їх безпеку. Наприклад, обчислювальна складність типових методів безпеки робить їх непригідними при розробці Інтернет-пристроїв. Таким чином, аби забезпечити безпечний зв'язок між великою кількістю пристроїв, необхідно забезпечити нові методи захисту та протоколи, які будуть відповідати специфікаціям прийнятих стандартів. Більшості IoT речам нині не вистачає пам'яті та обчислювальних ресурсів сучасних комп'ютерних пристроїв, роблячі їх вразливими шикорому спектру кібер-атак. Серед них великою загрозою є DDos атаки. Такі атаки реалізуються шляхом використання ботів, або впровадженням несанкціонованих пристроїв, а потім використання їх або інших скомпрометованих пристроїв для полегшення реалізації DDoS атаки шляхом генерування невпинного трафіку. Існуючим механізмам захисту бракує гнучкості та ресурсів. Рішенням цієї є заміна централізованої IoT інфраструктури на децентралізовану. IoT пристрої зможуть отримувати доступ до децентралізованої системи за допомогою смарт-контрактів.

1 СТАН ПИТАННЯ. ПОСТАНОВА ЗАДАЧІ

1.1 Стан питання

Одна з головних проблем безпеки, яку потрібно вирішити в IoT-мережі – це атака розподіленої відмови в обслуговуванні (DDoS-атаки). Атаки розподіленої відмови в обслуговуванні (DDoS-атаки) – це особливий тип атаки відмови у обслуговуванні, які переповнюють цільову або пов'язану інфраструктуру зловмисним трафіком. Така атака досягається шляхом використання ботів, шкідливих програм, скомпрометованих комп'ютерів та інших пристроїв під управлінням зловмисника. Це серйозно обмежує пропускну здатність та підключення до мережі, що призводить до порушення всіх служб у мережі.

У DDoS-атаках, спрямованих на Інтернет-речі, пристрої використовуються для перевантаження ресурсів мережі або цільових обчислювальних пристроїв. За останній час, DDoS-атаки, що використовують простий протокол виявлення служб, почали набирати популярність через наявні у протоколі вразливості. Багато пристроїв, таких як телекамери, бездротові роутери, або пристрої на основі IP, використовують цей протокол. Це ставить їх у великий ризик бути використаними у атаках.

Мета цієї роботи дослідити використання блокчейн-технологій для надання рішень, щодо безпеки Інтернет-речей з акцентом на DDoS-атаки.

1.2 Результати досліджень

За останні роки, кількість таких атак зросла. DDoS-атаки мають просту мету завадити або перервати послуги, доступні в Інтернеті, і їх мотивація може змінюватися від особистої образи до політичного шантажу. Реалізація DoS порівняно з іншими типами атак надзвичайно проста в умовах необхідних знань та програмних засобів. Таким чином, кількість DoS-атак постійно збільшується, що вимагає постійного дослідження проблемної області для того, щоб визначити оновлену систематику поточних атак, яка послужить основою для розробки методів захисту конкретного випадку. Розробка

концепції IoT та збільшення бездротових сенсорних мереж (WSN, а також застосування інших технологій бездротового зв'язку призводить до підвищеного ризику DDoS-атак на Інтернет-мережі. IoT- пристрої передають величезну кількість важливої інформації, а її виток може призвести до великих збитків.

На сьогодні запропоновано багато підходів до захисту від DDoS-атак. Однак лише деякі з них були розглянуті для широкого використання через їх ефективність та складність реалізації. Одне з таких рішень було запропоноване компаніями Akamai та CloudFlare. Їх рішенням засновані на хмарних технологіях, можуть поглинати DDoS-атаки, збільшуючи пропускну здатність мережі та перекладаючи зобов'язання виявлення атаки з пристроїв, експортуючи записи потоків з маршрутизаторів та комутаторів. Додатковий аналіз виконується в хмарі, а фільтрація пакетів використовується для збалансування, переправлення, або скидання трафіку всередині хмари. Однак для таких рішень потрібен сторонній постачальник послуг захисту від DDoS, що передбачає додаткові витрати та зниження продуктивності послуг.

1.3 IoT-мережа

IoT-мережа охоплює обробку даних та зв'язок між пристроями різних платформ та надає можливість автономної роботи без втручання людини. В останні десятиліття цей термін асоціюють із еволюцією Інтернету, і представляють не лише як сучасну технологію, але й соціальну парадигму. IoT вважається розширенням існуючої Інтернет-мережі і забезпечує підключення до Інтернету різних видів пристроїв. Підключення до Всесвітньої комп'ютерної мережі дозволяє керувати пристроями дистанційно та мати доступ до них, як до постачальників послуг, роблячи пристрої розумними об'єктами. Розвиток IoT-мереж бере свій початок з дев'яностих років минулого сторіччя, коли перші IoT-пристрої були підключені до TCP/IP мережі. Починаючи з цього часу, IoT-мережі продовжують невпинно

розвиватися, запроваджуючи нові технологію та вимагаючи нових рішень щодо їхньої безпеки.

Нині не існує єдиного визначення цьому поняттю. Досліджувач IoT-мереж Луїджі Атцорі описав IoT, як різні речі або предмети, такі як мітки для ідентифікації радіочастот, датчики, пускачі та стільникові телефони. Ці пристрої взаємодіють між собою, співпрацюючи з сусідами для досягнення спільних цілей. Досліджувач поділяє це визначення на орієнтоване на Інтернет(проміжне програмне забезпечення), об'єктно-орієнтоване(датчики та виконавчі механізми) та орієнтоване на семантику(представлення та зберігання інформації). Існує також така думка, що IoT має головним чином зосереджуватися на «речах», і впровадження IoT-мереж має починатися із збільшення інтелекту пристроїв. Деякі визначення у літературі випливають саме з такого бачення. Європейський кластер досліджень Інтернет речей представляє IoT, як «глобальну мережеву структуру, динамічну, із можливостями до само налаштування, засновану на стандартизованих та сумісних протоколах зв'язку, де фізичні «речі» та віртуальні машини мають ідентифікатори, фізичні та віртуальні особистості та використовують інтелектуальні інтерфейси, ідеально інтегровані у мережу. Типова схема IoT-мережі наведена на рисунку 1.1.

Сектор стандартизації телекомунікацій запронував модель, яка складається з чотирьох рівнів:

- рівень застосунків. Відповідає за надання послуг користувачам;
- рівень підтримки застосунків. Відповідає за підтримку застосунків, яка відповідає вимогам конкретного пристрою та загальним вимогам, які застосовуються до більшості пристроїв, такі як обробка та збереження інформації, та запровадження безпеки інформації;
- мережевий рівень. Відповідає за функції управління мережевим підключенням, такі як управління мобільність, автентифікація, авторизація та облік;

- Рівень пристроїв. Цей рівень представлений пристроями та шлюзами для зв'язку пристроїв із сервером, та розглядає свої елементи як процесори, носії пам'яті, прошивку, датчики та виконавчі пристрої та їх особливості. Особливостями пристроїв є здатність взаємодіяти з мережею безпосередньо. Вони можуть збирати та надсилати інформацію безпосередньо, не використовуючи шлюзи для зв'язку. Особливості шлюзу включають підтримку декількох інтерфейсів, що дозволяють спілкуватися із пристроями, незважаючи на різні типи дротового та бездротового підключення.

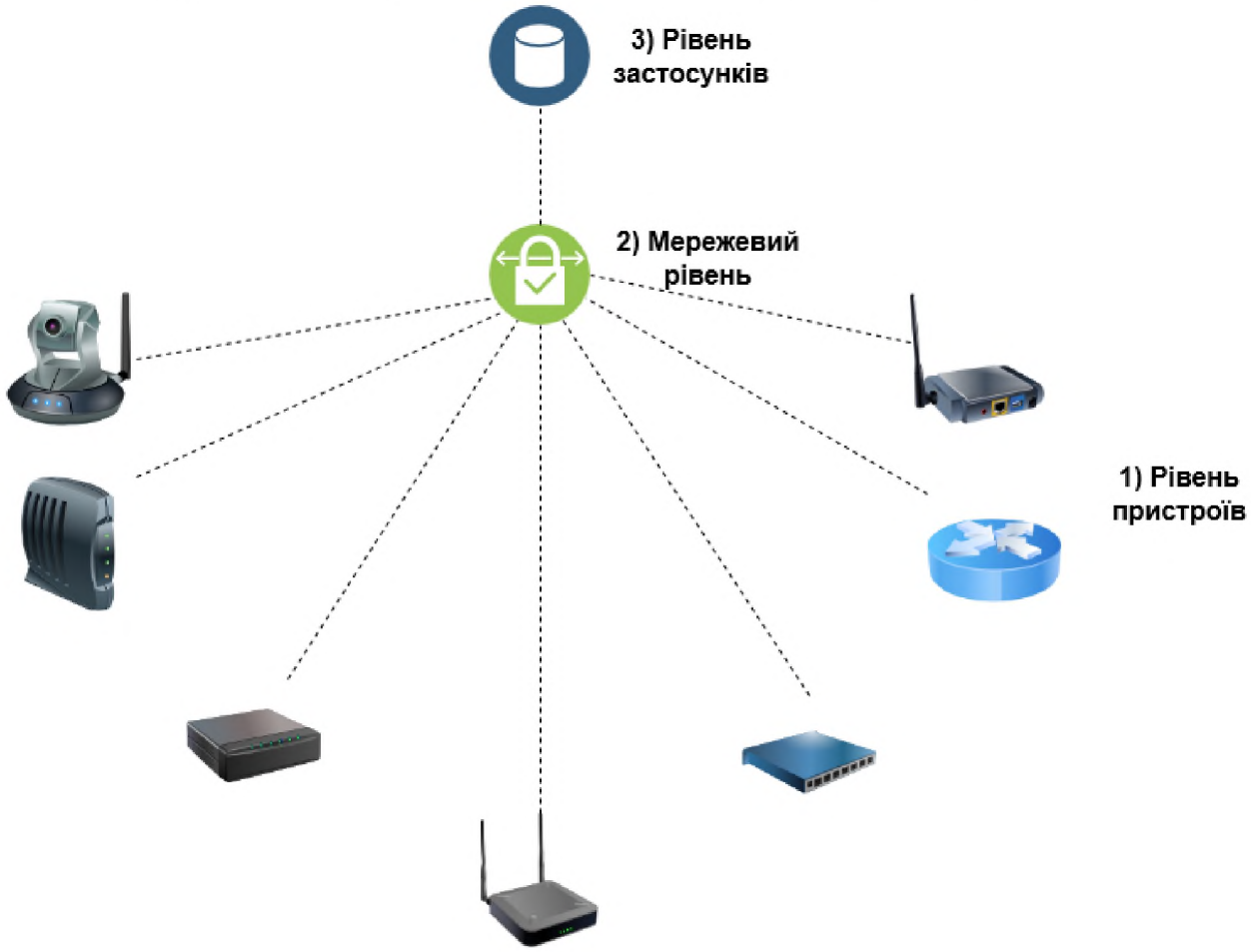

Рисунок 1.1 – Типова схема IoT-мережі

Ця модель включає в себе функції управління та функції безпеки, пов'язані з усіма рівнями. Вони також класифікуються на загальні та специфічні можливості. Загальні функції управління IoT включають управління пристроями, такими як віддалена активація та деактивація, діагностика, оновлення програмного забезпечення, управління статусом пристрою, топологією мережі та управлінням трафіком та перевантаженням.

Загальні функції безпеки включають авторизацію, автентифікацію, конфіденційність та цілісність даних.

Існує безліч різноманітних програм для IoT. Ці програми пронизують повсякденне життя людей, перетворюючи світ на розумний світ, що дозволяє обчисленням стати «невидимими» для користувача. Існує безліч сфер, у яких застосовується IoT, основні з яких:

- Смарт-продукція, така як смартфони, розумні будинки, розумні автомобілі, смарт-телевізори та переносні пристрої.

- Смарт-здоров'я: профілактика та охорона здоров'я, наприклад, моніторинг та контроль частоти серцевих скорочень під час фізичних вправ, моніторинг стану пацієнтів у лікарнях та їхніх будинках. Запобігання проблемам із здоров'ям стає більш ефективним завдяки збору інформації з організму людини, а поставлення діагнозу стає більш точним завдяки веденню профілю пацієнта із довгостроковими записами.

- Інтелектуальний транспорт: повідомлення про дорожній рух, управління маршрутом, дистанційний контроль транспортним засобом, координація автомобільних доріг.

- Інтелектуальне постачання енергії: моніторинг енергетичних установ, розумних підстанцій, постачання електроенергії та дистанційне вимірювання житлових лічильників електроенергії.

- Смарт-промисловість: економія енергії, контроль забруднення, безпека виробництва, моніторинг виробництва продукції, відстеження товарів у ланцюгу поставок, моніторинг екологічних умов та процесів контролю виробництва.

- Розумне місто: моніторинг вібрацій та стану будинків, мостів, історичних пам'яток. Розумне електричне освітлення, системи моніторингу управління вогнем та сигналізацією. Розумні дороги з попередженнями щодо кліматичних умов та аварій, моніторинг паркувальних місць, оптимізація збору сміття.

1.4 DDoS-атаки

Розподілена атака відмови в обслуговувані(DDoS) – це зловмисні дії, спрямовані на порушення нормального трафіку IoT-мережі, заважаючи IoT-пристроям передавати інформацію до серверу шляхом переповнення серверу великої кількістю запитів. При такій атаці сервер не може вчасно приймати інформацію від IoT-пристроїв на відповідати на неї, через що уся мережа опинитися у недієздатному стані. Найчастіше, DDoS-атаки досягають такого ефекту використовуючи власні пристрої, або компрометуючи IoT-пристрої мережі, розповсюджуючи серед них зловмисне програмне забезпечення. Заражені пристрої називають ботнетом. Як тільки необхідна кількість пристроїв заражена зловмисним ПЗ, зловмисник може розпочати DDoS-атаку. Кожен заражений пристрій відправляє на сервер запити, які потенційно можуть призвести до перевантаження системи та відмови в обслуговувані звичайного трафіку. Розпізнати зловмисний трафік досить часто буває важко, оскільки заражені пристрої є частиною IoT-мережі.

DDoS атаки спрямовані на те, щоб спричинити відмову у доступі автентифікованим користувачам, роблячи недоступними мережеві ресурси. Процес атаки в значній мірі націлений на розподілений доступ до пристроїв, використовуючи відомі вразливості. Атаки спрямовані на різні рівні інфраструктури мережі, наприклад, рівень додатків або рівень зв'язку. На основі архітектури мережі, DDoS-атаки класифікуються наступним чином:

- атаки на рівні додатків. Ця атака спрямована на вичерпання цільових ресурсів мережі, призводячи до відмови в обслуговуванні. Для реалізації атаки зловмисник використовує вразливості застосунків або системи, що спричиняє нестабільність мережі. Ці атаки часто плутають з помилками при розробці системи через низький рівень трафіку, необхідний для їхньої успішної реалізації. Прикладами таких атак є HTTP-flood, slowloris та атака нульового дня. HTTP-flood – це така атака, при якій від великої кількості пристроїв надходять HTTP-запити, вимагаючи таким чином постійного доступу та вичерпуючи ресурси цільових пристроїв. Типова реалізація HTTP-flood представлена на рисунку 1.2. Атаки Slowloris надсилає неповні запити до

системи із визначеним інтервалом часу з метою тримати канали запитів задіяними протягом тривалого часу, роблячи неможливим надходження запитів від законних користувачів.

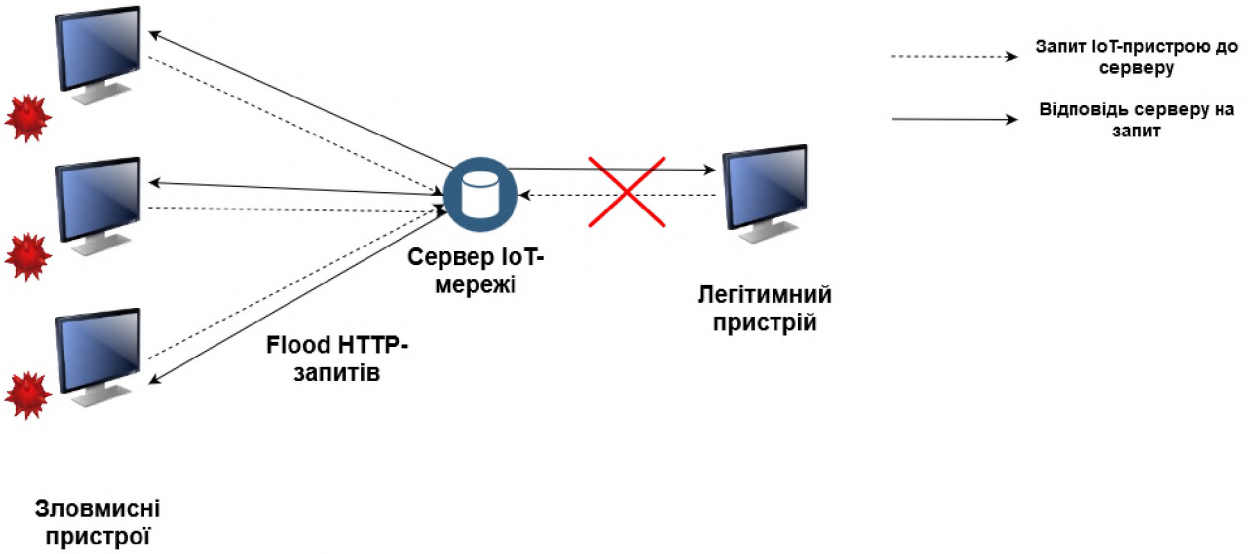

Рисунок 1.2 – Типова реалізація HTTP-flood

- атака вичерпання ресурсів: Під час таких DDoS-атак, використовуються вразливості мережевого рівня та рівня зв'язку. Ці атаки також називають атаками виснаження стану, оскільки вони виснажують обчислювальні ресурси системи, такі як обчислювальна потужність, оперативна пам'ять та пам'ять жорстких дисків. Оскільки ця атака використовує вразливості протоколів, крім того, що вона є досить об'ємною, вона формує суміш між конкретними повідомленнями та їх об'ємом, який надсилається жертві. TCP SYN floods відправляють безліч синхронізованих повідомлень жертві, але не представляють ніякого підтвердження доступу для встановлення зв'язку з підробленими IP-адресами. Таким чином, цільові ресурси системи вичерпуються, оскільки вони відповідають на кожне рукостискання, але ніколи не отримують підтвердження від зловмисника. Іншими прикладами таких атак є Ping of Death, що представляють собою пакети пінгу, що перевищують 65535 байт, роблячи пристрої недоступними, та атака смурфів, що дестабілізує послуги мережі шляхом надсилання великого обсягу ICMP-пакетів. Як видно з рисунка 1.3, зловмисник створює

мережевий пакет, доданий до помилкової IP-адреси, передаючи ICMP-повідомлення. Вузли мережі зобов'язані відповісти на запит. Відповіді слідують нескінченим циклом, надсилаючись назад до IP-адрес мережі;

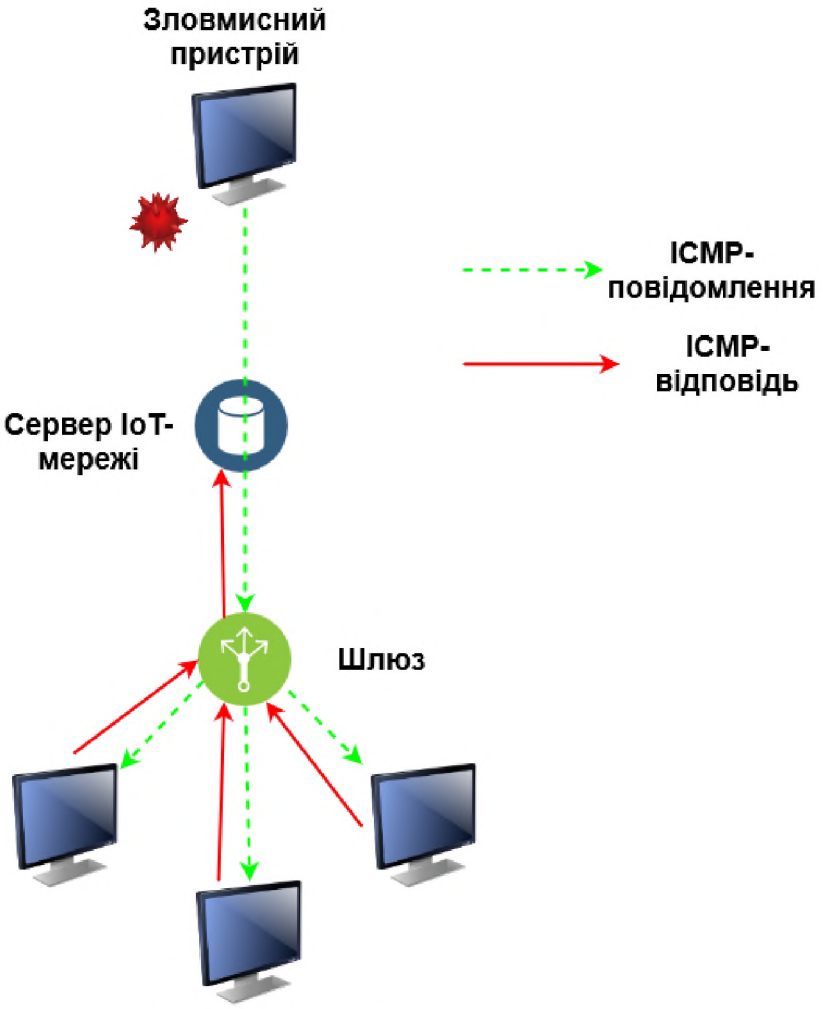

Рисунок 1.3 – Реалізація атаки вичерпання ресурсів

- атаки об'єму. Масивні обсяги даних надсилаються жертві за допомогою ботнетів або інших методів поширення даних, вичерпуючи пропускну смугу між сервером та пристроями. Зазвичай, для використання будь-якого надмірно збільшеного пакету використовується протокол UDP. Атаки посилення DNS виконують сервісні запити на заміну поля адреси зловмисника на адресу жертви, викликаючи посилення відповіді серверами та вичерпуючи пропускну смугу жертви.

Розпізнати DDoS-атаку зазвичай досить складно. Найбільш очевидною ознакою атаки є зниження ефективності серверу та сповільнення послуг, що

надаються ним. Проте існують й інші причини, що спричиняють такі наслідки, наприклад, легальне підвищення трафіку. Саме тому, для встановлення того, чи дійсно причиною сповільнення трафіку є DDoS-атака, необхідно перевірити додаткові ознаки. Серед них:

- Підозріла кількість запитів, які відправляються від одного пристрою.
- Непояснювальний сплеск запитів до однієї кінцевої точки.
- Дивні ознаки запитів, такі як сплески запитів в один і той же час або через один і той самий проміжок часу.

За останній час, із збільшенням використання IoT-мереж зростає й кількість та різноманіття DDoS-атак. Ці атаки можуть відбуватися на різних рівнях IoT-мережі. На фізичному рівні можуть бути реалізовані різні види атак, такі як захоплення пристроїв або посилання великої кількості даних з серверів зловмисника. Найпопулярнішою з усіх є атака, при якій зловмисник створює перешкоди для законного трафіку даних та виснажує ресурси серверу. Розглянута надалі система захисту на основі блокчейну є досить дієвою проти таких атак.

1.5 Технологія блокчейн

Про технологію блокчейн поза контексту криптовалют почали активно розмовляти ще у 2014 році. В цей час люди почали розуміти, що децентралізований підхід, який використовується у Bitcoin, може бути використаний у інших областях, і що технологія блокчейн є важливою складовою. Наприклад, використання цієї технології у системах обліку може захистити данні від несанкціонованих модифікацій. Забезпечення у системах таких властивостей, як незмінюваність, цілісність, trustlessness, надійне узгодження отримало широкий відклик від суспільства.

Блокчейн-мережа складається з двох компонентів: криптографічно захищеного списку записів, та загальнодоступної мережі обчислювальних пристроїв. Ці записи є загальнодоступними, незмінними та послідовно генерованими, відомими як блоки. Це неперервна послідовність ланцюга

блоків, побудованого за певними правилами, яких містить у собі записи. Схема такого зберігання даних зображена на рисунку 1.4.

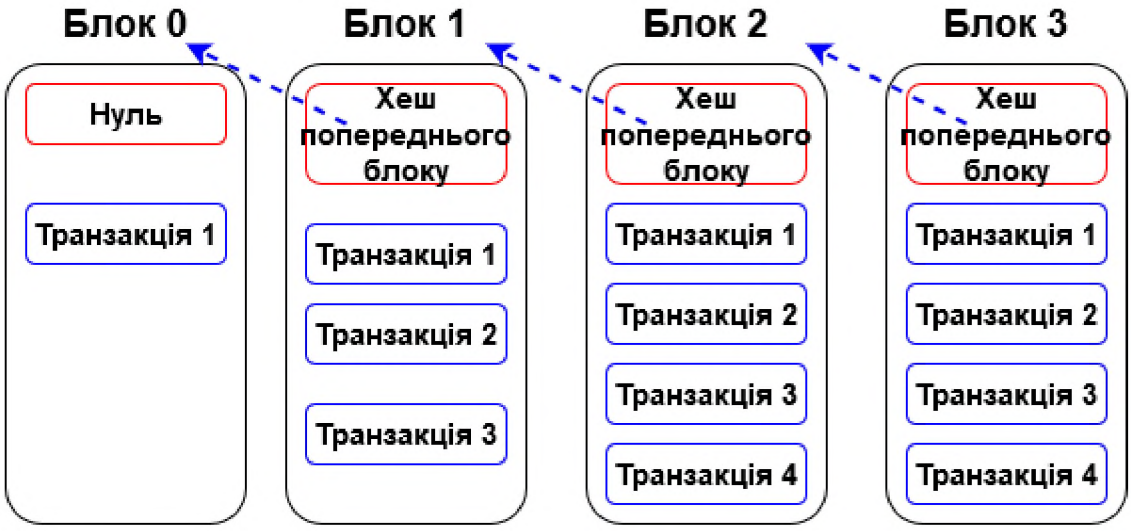

Рисунок 1.4 – Спосіб зберігання даних у блокчейні

Зв'язок між блоками забезпечується не лише за допомогою нумерації, але й тим, що кожен блок містить власний хеш, а також хеш попереднього блоку. Завдяки такій системі, записи у блоках є незмінними, оскільки будь-які зміни призведуть до змін у хеші блоку, а також у всіх наступних блоках.

Кожний з обчислювальних пристроїв у блокчейн-мережі представляє собою вузол, який зберігає копію ланцюга блоків, та здатен приймати або відхиляти нові блоки, спостерігаючи за даними у них. Кожен запис має часову мітку та додається після верифікації ланцюгом. Якщо ж зловмисник спробує відправити транзакцію під виглядом IoT- пристрою та отримати доступ до сервера IoT-мережі, то доступ буде відхилено, оскільки у нього відсутній приватний ключ пристрою, який би міг підтвердити право доступу. Список адрес пристроїв, які мають доступ до відправлення даних на сервер, зберігається у блоках блокчейн-мережі на кожному з вузлів, що забезпечує незмінність цих записів. Якщо один з вузлів спробує підмінити данні, інші зможуть це побачити та відхилити блоки, запропоновані зловмисним вузлом.

У контексті блокчейну, вузол – це електронні пристрої, які підключені до мережі та мають свою IP-адресу. Вузли є кінцевими точками мережі. Для того, щоб IoT-пристрої могли проходити верифікацію у блокчейн-мережі, він

має відправити транзакцію до одного з вузлів. Тому вузли також є точкою перерозподілу комунікацій. Пристроям не потрібно бути вузлами блокчейн-мережі для того, що взаємодіяти з нею. Їм достатньо взаємодіяти з одним із вузлів. Вузол може виконувати одну або декілька з наступних дій: приймати або відторгати транзакцію, перевіряти чинність транзакцій, зберігати криптографічнозахищені блоки, виступати у ролі точки зв'язку. Приклад взаємодії IoT-пристрою з вузлами блокчейн-мережі зображено на рисунку 1.5

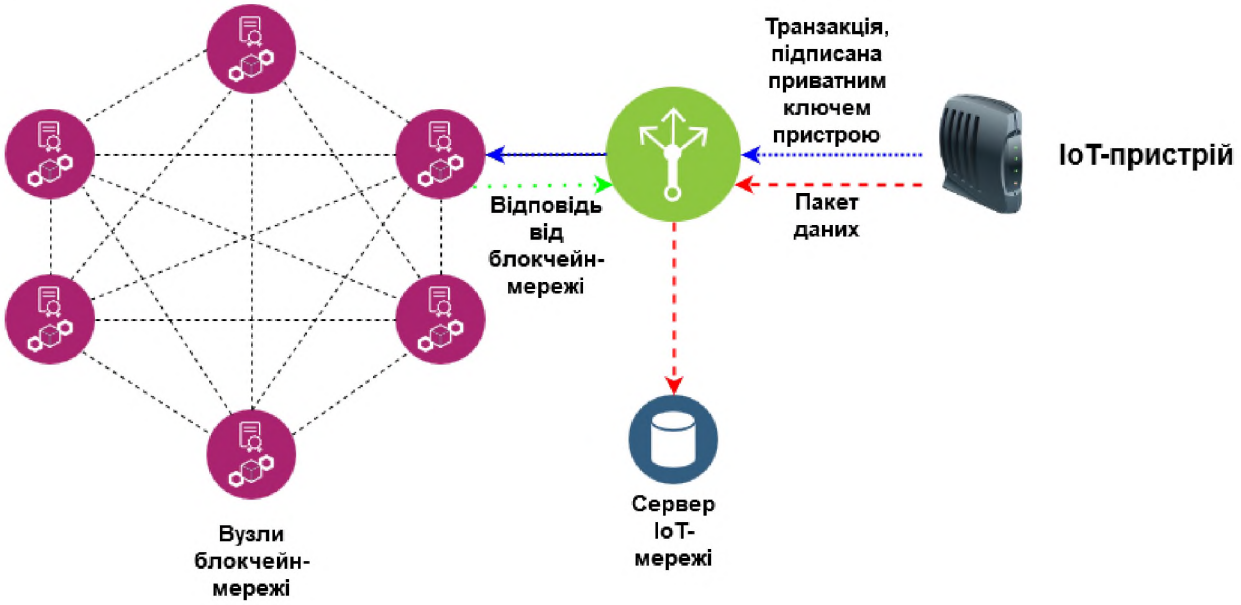

Рисунок 1.5 – Приклад взаємодії пристроїв з вузлами блокчейн-мережі

Блокчейн заснований на наступних поняттях: база даних, блок, хеш, майнер, транзакція та консенсус:

- база даних: цей аспект охоплює фундаментальні можливості блокчейнів або накопичення інформації у нетрадиційному вигляді(у формі рядків та стовпців). У блокчейні всі записи транзакцій користувачів зберігаються з високою пропускною здатністю, децентралізованим контролем та незмінними записами;

- блоки: блоки зберігають дані, пов'язані з різними транзакціями серез учасників. Вони об'єднані у ланцюг, зберігаючи хеш-значення попередніх блоків, утворюючи цикл щільно взаємопов'язаних даних. Блок розділений на дві частини: заголовок та тіло блоку. Заголовок зберігає інформацію про блок, в той час як тіло блоку зберігає інформацію транзакцій;

- хеш: це складні математичні задачі, які відповідають за ідентифікацію та перевірку блоку. Вузол повинні вирішити цю задачу, для того щоб додати блок до ланцюга. Хеш є унікальним для кожного запису, гарантуючи таким чином верифікацію блоку;

- вузол: вузол у блокчейн мережі, який вирішує обчислювальну задачу знаходження хеша нового блоку. Такий процес називається майнінгом блоку. Згенерований блок поширюється по всій мережі, а майнер винагороджується за свою роботу;

- транзакція: це найменша кількість інформації, яка зберігається у блоці як тільки блок верифікується більшість учасників мережі. Записи є доступними та незмінними;

- консенсус: консенсус щодо того, як генерувати блоки – це ключова характеристика блокчейну, досягнута за допомогою різноманітних механізмів. Найвідоміші види консенусу – це PoW та PoS. Перший з них базується на роботі, яку проводить майнер для генерації блоку, тоді як другий розподіляє цю роботу на основі віртуальних валютних токенів учасників.

Для того, щоб IoT-пристрій зміг стати учасником блокчейн-мережі, та проходити перевірку та легітимність, для нього створюється обліковий запис. Він складається з двох ключів: публічного та приватного. Публічний ключ – це 42 унікальні шістнадцяткові символи. Саме публічний ключ вважається адресою пристрою, яку можна порівняти з IP-адресою. Для того, щоб отримати дозвіл на відправлення даних до серверу, пристрою необхідно спочатку відправити транзакцію, підписану приватним ключем, до блокчейн-мережі. Підписана транзакція розшифровується публічним ключем пристрою для підтвердження того, що саме він відправив її. Також перед отриманням дозволу до відправлення даних до серверу призводиться ще декілька перевірок, які будуть розглянуті пізніше.

Обліковий запис на основі блокчейн-технології забезпечує такі властивості, як цілісність історії зміни бази даних, мінімізація затримок

синхронізації та резервного копіювання, можливість здійснення аудиту в режимі реального часу.

1.6 Консенсус та Proof-of-work

У блокчейн не існує центральної влади. Блоки створюються вузлами незалежно. У блокчейн-мережі будь-хто може почати генерувати блоки. Повні вузли зберігають увесь ланцюжок блоків, і перевіряють кожен новий блок. Коли певна кількість вузлів мають однакові блоки в основному ланцюжку, вони вважаються такими, що досягли консенсусу.

Консенсус – це стан згоди учасників системи. Алгоритм досягнення консенсусу – це певний механізм або протокол, за яким вузли досягають згоди у створені блоків. Таким чином транзакції у блоках неможливо підмінити.

Механізм консенсусу складається з двох етапів: перевірки блоку та вибору найширшого ланцюга. Ці два етапи виконуються кожним вузлом незалежно. Блоки транслюються мережею, і кожен вузол, що отримує новий блок, повторно передає його своїм сусідам. Але перед цим, вузол виконує перевірку блоку, щоб забезпечити поширення лише валідних блоків. Більшість нині існуючих блокчейн-мереж використовують консенсус proof-of-work. Хоча існує багато інших видів консенсусів, у цій роботі буде описано лише консенсус proof-of-work.

Головна ідея консенсусу proof-of-work – уникнути кібератаки на вузли. Використовується механізм, у якому вузол має довести, що він витратив де-який час, аби вирішити математичну задачу знаходження хеша блоку, результат якої задовольняє де-яку вимогу, згідно якої інші користувачі мережі перевіряють валідність блоку. Завдання пошуку такого результату базується на двох принципах. По-перше, PoW має бути важким, проте не неможливим. По-друге, перевірка результату має бути набагато швидшою та легшою для перевірки.

Коли пристрій відправить транзакцію до блокчейну, вона вкладеться в блок з максимальною ємність 1 мегабайт, а потім дублюється через кілька

вузлів. Вузли перевіряють законність транзакцій у кожному блоці. Для проведення цієї перевірки вузлам необхідно розв'язати «математичну головоломку», відому як проблема proof-of-work. Після перевірки блоку іншими вузлами, він додається до блокчейну.

PoW генерує блоки наступним чином: відправник додає до повідомлення довільне число і застосовує до повідомлення хеш-функцію. Мета вузла – знайти такий хеш цього повідомлення, щоб він відповідав поточній складності блоку. Довільне число змінюється до тих пір, поки не буде знайдено необхідний хеш блоку. Фактично, пошук необхідного хеша для блоку зводиться до пошуку довільного числа.

1.7 Основні принципи безпеки IoT-мережі

Безпека та конфіденційність є основними принципами будь-якої інформаційної системи. Через IoT-мережі щодня проходить величезна кількість інформації. Зрозуміло, що важливим чинником є не лише безпека та цілісність цих даних, а й стійкість мережі до атак та забезпечення таким чином пропускної спроможності мережі. Зазвичай, досягти захищеності можна шляхом забезпечення комбінації автентифікації, авторизації та ідентифікації. Ці поняття визначаються наступними концепціями:

- Цілісність. Гарантія того, що інформація не була змінена за винятком тих, хто має відповідні права вносити ці зміни. У контексті блокчейну цілісність забезпечується гарантією незмінності транзакцій. Для перевірки цілісності транзакцій використовують стійкі криптографічні механізми.

- Доступність. Гарантія того, що користувачі системи зможуть користуватися нею за необхідністю. Іншими словами, послуга завжди активна на запит законного користувача, а це потребує складних комунікаційних інфраструктур та баз даних. Блокчейн досягає цієї мети, дозволяючи користувачам встановлювати зв'язок з кількома іншими користувачами та підтримувати блоки у децентралізованому вигляді за допомогою зберігання копій мережі на різних пристроях.

- Конфіденційність. Гарантія того, що сторонні особи не отримають інформацію. Тобто лише ті, хто мають відповідні права, зможуть отримати доступ до інформації, незалежно від того, обробляється вона чи передається. Задля забезпечення цього принципу у блокчейні використовується механізм для псевдоанонімізації із використанням хеш-функцій для ідентифікації користувачів.

- Автентифікація, авторизація та аудит. Забезпечення перевірки осіб, які виконують певні функції у системі, якими правами володіє особа, а також зберігання інформації, яку використовувала особа. Структура блокчейну забезпечує ці три функції, оскільки лише користувачі, у яких наявні приватні ключі, можуть виконувати транзакції, і всі транзакції перевіряються та є відкритими.

- Безвідмовність. Гарантія того, що один користувач не зможе заперечити або заборонити дії іншого користувача в системі. Ця властивість також забезпечує доказ того, що певни користувач здійснив певну дію, таку як переказ коштів або відправлення повідомлення. Оскільки кожна транзакція підписується приватним ключем, то користувач не зможе заперечити свої дії у системі.

Приватність – це право користувача на безпечну передачу своєї інформації. У блокчейн-мережі користувачі використовують адресу для передачі транзакцій. У одного користувача може бути необмежена кількість адрес. Транзакція може розглядатися як ланцюжок підписів, що підтверджують володіння та передачу цінностей, таким чином забезпечуючи аудит. У випадку з IoT-мережею, цінними записами є список пристроїв, що можуть надсилати дані на сервер. Одне з найбільших занепокоєнь полягає у тому, що транзакції можуть розкривати інформацію про пристрій, якої потенційно міг би скористатися зловмисник.

Концепція конфіденційності у блокчейні полягає у забезпеченні анонімності та розв'язувані транзакцій. Анонімність транзакції потребує

неможливості зв'язати конкретну транзакцію з IoT-пристроєм. Зв'язати конкретну блокчейн-адресу з IoT-пристроєм є майже неможливим, оскільки ці адреси створюються на сервері IoT-мережі. Співвідношення облікового запису із конкретним IoT-пристроєм також зберігається лише на сервері.

1.8 Механізми забезпечення безпеки

Володіння цінностями та транзакції в мережі здійснюються із використанням концепції приватних ключів та цифрових підписів. Ключі генеруються на основі концепції криптографії на відкритому ключі. Кожна транзакція вимагає підпису, який вважається дійсний і підтверджує володіння цінностями.

1.8.1 Хеш-функція

Хеш-функції – це математичні функції, що генерують суму, відбиток даних. При застосуванні до певного набору даних, функція генерує вихідний результат, який є унікальним. Хоча й існує імовірність існуваннях двох наборів даних з однаковим хешем, ця імовірність дуже низка. Одне з найпоширеніших застосувань хешу – це перевірка цілісності даних. Обсяг результату хеш-функції залежить від використаного алгоритму, але найголовніше те, що довжина хеш-функції завжди однакова незалежно від розміру вхідних даних. У блокчейні широкої популярності набув алгоритм SHA-3, вихідна довжина хешу якого дорівнює 256 байтам. Алгоритм хешування завжди повинен мати наступні властивості[5]:

- Хеш-функція – одностороння. Знайти вихідні дані з хешу має бути обчислювально дуже складно.

- Стиснення. Бажано, щоб розмір хеша представляв невелику частку вхідних даних.

- Легкість розрахунку. Алгоритм хешування не повинен бути дорогим для обчислення.

- Дифузія або так званий «Лавинний ефект». Аби перешкодити зворотному проектуванню алгоритму, при зміні одного біта вхідних даних, результат хеш-функції має бути змінений щонайменше на 50%.

- Колізія. Обчислювально важко знайти два набори вхідних даних, які генерують однаковий хеш.

1.8.2 Шифрування

Шифрування – це сукупність методів, які перетворюють інформацію на щось, що сторонній агент не зможе зрозуміти. Системи шифрування працюють наступним чином: отримуючи повідомлення та ключ, система генерує нове зашифроване повідомлення, яке передається по незахищеним канал, не ризикуючи бути зрозумілим особам, які не мають ключа дешифрування. Система вважається повною лише у тому випадку, коли зашифроване повідомлення можна розшифрувати.

У блокчейні використовується пара ключів: публічний та приватний. Перший – для шифрування, а другий – для дешифрування і навпаки. Це досягається завдяки використанню математичних функцій, які мають властивість бути незворотними. Для генерації пари ключів у блокчейні використовується метод еліптичних кривих. Цей метод забезпечує надійність та велику кількість унікальних ключів. Ефективність алгоритму шифрування можна виміряти за наступними показниками:

- Обчислювальне навантаження. Вимірює ефективність, із якою алгоритм може запроваджувати зміну ключів;

- розмір ключів. Чим він менший, тим зручніше користувачам використовувати ключі. Наприклад, у алгоритмі RSA розмір ключів у бітах складає відповідно1088 та 2048 та публічного та приватного ключа, у алгоритмі DSA – 1026 та 160 бітів у методі еліптичних кривих – 161 та 160 бітів. У цьому аспекті метод еліптичних кривих має значну перевагу [6];

- розмір діапазону. Він відповідає кількості бітів, необхідних для передачі повідомлення після кодування та підписання;

Якщо порівнювати алгоритми шифрування RSA та еліптичних кривих, то другий однозначно перемагає за усіма показниками. Саме тому цей метод використовується у таких блокчейн-мережах як Bitcoin та Ethereum.

1.8.3 Цифровий підпис, адреса та гаманець

Цифровий підпис можна визначити як шифрування хешу документа, використовуючи приватний ключ для підпису та публічний ключ для підтвердження того, хто підписав документ. У Ethereum для підпису транзакцій та повідомлень використовується алгоритм цифрового підпису еліптичної кривої. Ця версія заснована саме на еліптичних кривих. Складність логарифму не дозволяє третім особам підписувати документи без приватного ключа користувача. Ця властивість також гарантує, що користувач не зможе відмовитися від свого підпису. Зазвичай, підписується хеш повідомлення. Завдяки цьому, розмір підпису завжди складає фіксовану кількість бітів. Підпис повинен гарантувати цілісність, невідмовність від підпису та достовірність. У Ethereum приватний ключ отримується шляхом генерації випадкового числа довжиною 256 біт, а публічний ключ шляхом множення приватного ключа на одну точку в кривій, відому як «точка генератора». Користувачу достатньо зберегти лише свій приватний ключ, оскільки він зможе отримати публічний ключ у будь-який час.

З цього моменту, у вузла вже наявна пара ключів, необхідна для генерування адреси. Не слід плутати цю адресу із IP-адресою. Адреса у блокчейн – це набір цифр, отриманий за допомогою приватного ключа. Адреса використовується для того, щоб повідомити системі, хто є відправником транзакції, оскільки кожна транзакція підписується приватним ключем відправника. Вузол повинен виконати операцію подвійного хешування, спочатку використовуючи SHA-256, а потім RIPEMD-160 [7].

Користувачі мережі Ethereum мають ключі для підтвердження свої транзакцій. Як правило, ці ключі зберігають у спеціальних електронних

гаманцях. У гаманці використовується функція для генерування ключів та їх зберігання. Існує два види гаманців: детерміновані та випадкові. Гаманець використовує один первинний ключ, який називають сід-фразою, для створення інших ключів за допомогою хеш-функції. Зберігається у такому випадку лише сід-фраза, оскільки усі інші ключі можуть бути пораховані за її допомогою. Випадковий гаманець використовує алгоритм для генерації випадкових чисел з довжиною 256 біт. Такий гаманець повинен зберігати всі створені ключі [8].

1.9 Мережа Ethereum та смарт-контракти

Ethereum – одна з найбільших програмних платформ, яка дозволяє будувати децентралізовані додатки. На відміну від мережі Bitcoin, у якій можливі лише прості фінансові транзакції, Ethereum надає набагато ширший потенціал. Розробники Ethereum називають свій продукт децентралізованою комп'ютерною мережею, побудованою на основі блокчейн-технологій.

Як і більшість інших децентралізованих мереж, Ethereum використовує криптографічно захищені транзакції для підтримки безпеки мережі та верифікації блоків. Мережа складається з більш ніж 10000 вузлів, частина з яких використовується для перегляду записів, а інша – для генерації блоків. Унікальною особливістю Ethereum є те, що користувачі можуть розробляти програми, які працюють на блокчейні так само, як програмне забезпечення працює на комп'ютерах. Для роботи цих програм, існує спеціальне програмне забезпечення EVM. Ця віртуальна машина розміщена на кожному з вузлів мережі, та використовує пам'ять та ресурси вузла. Будь-які зміни у віртуальній машині можна зробити лише через транзакцію, а стан цих змін зберігається у блоках. Завдяки цьому, провести які-небудь несанкціоновані зміни неможливо.

Невід'ємною складовою Ethereum є смарт-контракти. Смарт-контракт – це програмний код, який працює на блокчейні. Він складається з набору функцій та інформації, яка зберігається у кожному вузлі системи. Хто завгодно

може написати смарт-контракт, та розмістити його у мережі Ethereum. Розміщення контракту є такою ж транзакцією, тому вона відправляється до одного з вузла, який потім розмістить програмний код у мережі. Ключовою особливістю смарт-контрактів є те, що як тільки код розміщений у мережі, його неможливо змінити.

Кожен контракт має своє сховище для зберігання інформації, яке є невразливим до несанкціонованої модифікації. Єдиний спосіб змінити або прочитати інформацію – це викликати функцію смарт-контракту. Кожен виклик, який змінює стан змінних у сховище – є транзакцію, яка перевіряється вузлами мережі. Для перевірки IoT-пристроїв, сервер IoT-мережі заздалегідь робить записи до сховища блокчейну. У цих записах вказується, які пристрої можуть відправляти дані. Запис цих пристроїв можна здійснити лише через транзакцію, підписану лише приватним ключем серверу, саме тому підробити ці записи зловмисником неможливо. Коли IoT-пристрій захоче відправити дані до серверу, йому необхідно буде спочатку пройти перевірку, викликавши метод смарт-контракту. Цей виклик відправляється до вузла блокчейн-мережі за допомогою транзакції. Вузол в свою чергу викликає метод смарт-контракту на своїй копії віртуальної машини, додає результат до блоку, який перевіряється іншими вузлами та відправляє відповідь назад до IoT-мережі. Схема перевірки пристрою за допомогою блокчейн-мережі та смарт-контракту зображена на рисунку 1.6.

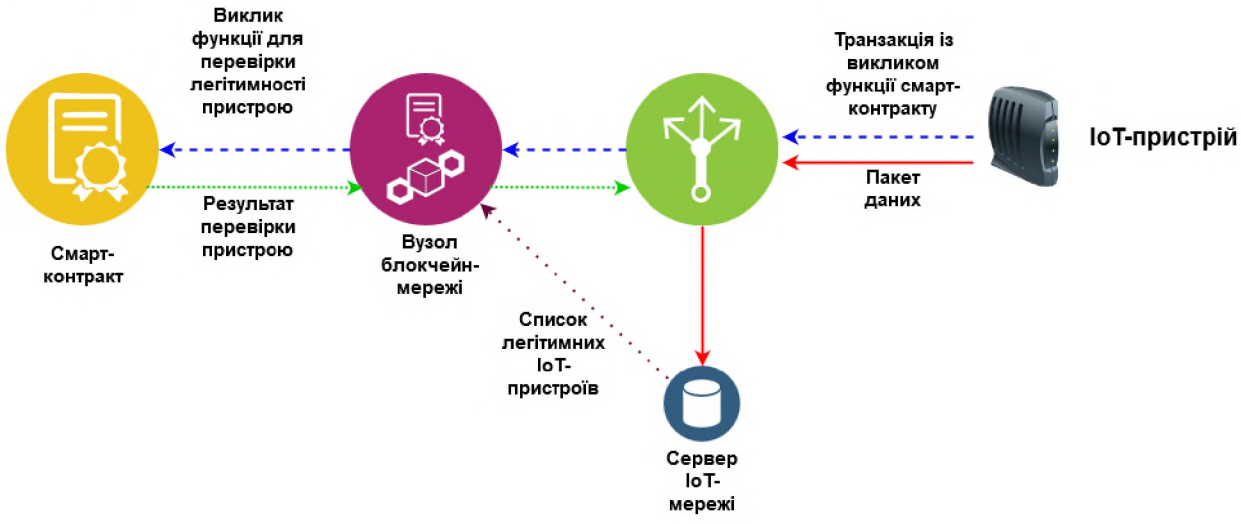

Рисунок 1.6 – Перевірка пристрою блокчейн-вузлом за допомогою смарт-контракту

Для кожної транзакції та операції у мережі Ethereum існує таке поняття, як gas limit. Газ – це аналогія до слова «ресурс» у мережі. Газ – це вартість операцій, яку стягує вузол за обробку транзакції. Оскільки вузол використовує власні ресурси для виклику функцій, затрати цих ресурсів необхідно покрити. Для цього, до транзакції додається певна кількість коштів, яка заздалегідь знаходиться на адресі пристрою. В автомобільній аналогії це те, скільки бензозаправна станція буде стягувати з водія, щоб заповнити його автомобіль бензином - зазвичай частина вартості за галон або літр. При відправці транзакції користувач може вказати максимальну кількість газу, яку він готовий оплатити та ціну за кожну одиницю газу. При виконанні функції смарт-контракту, кожна операція, така як математичні операцію, або запис до сховища коштують певну кількість газу. У випадку, якщо вказаної користувачем кількості газу недостатньо, то транзакція скасовується, а кошти, витрачені на газ не повертаються відправнику. Ціна на газ зростає і падає в залежності від того, наскільки завантажена мережа Ethereum, тобто скільки транзакцій необхідно перевірити.

Кожен смарт-контракт має свою унікальну публічну адресу, і так саме може зберігати на ньому кошти та використовуватися як звичайна адреса у блокчейн-мережі. Смарт-контракти є публічними, це означає що кожен може звернутися до його функцій, якщо не обмежити доступ до них. Саме тому, при проектуванні смарт-контракту слід бути уважним та приділяти велику увагу безпеці контракту.

1.10 Життєвий цикл смарт-контракту

Цикл 1. Проектування

Розробка смарт-контрактів – це відповідальний процес, оскільки як тільки він потрапляє у мережу, його код не може бути змінено. Смарт-контракти пишуться на спеціальній мові програмування – Solidity, який потім

компілюється у байткод та ABI. Байткод – це машинний код, який необхідний для мережі Ethereum для того, щоб викликати функції контракту. ABI(Прикладний інтерфейс байткоду) – це інтерфейс, необхідний для децентралізованих додатків для того, щоб звертатися до смарт-контракту.
При розробці смарт-контракту треба пам'ятати про його безпеку, оскільки він є публічним і будь-хто може звернутися до нього. Існує багато базових підходів, які необхідно притримуватися. Безпека для смарт-контракту для IoT-мережі є ще важливішої, оскільки від цього залежить стійкість мережі.

Цикл 2. Розміщення контракту у мережі

Для того, що розмістити смарт-контракт у мережі необхідно мати байткод контракту, та адресу з ефіром, оскільки розміщення смарт-контракту також є транзакцією. Вузол приймає таку транзакцію та розміщує код у мережі, який потім розійдеться по іншим вузлам мережі. При розміщення також викликається конструктор смарт-контракту: спеціальна функція, яка викликається лише один раз та необхідна для ініціалізації початкового стану смарт-контракту.

Цикл-3. Робота із розміщеним смарт-контрактом

Як тільки контракт розміщено у мережі, його неможливо змінити. Змінювати стан сховища контракту можливу лише через його публічний інтерфейс. Виклик кожної функції, яка змінює сховище контракту, коштує газ. Тому при проектуванні контракту слід пам'ятати про це та оптимізувати функції контракту.

1.11 Висновок

Незважаючи на те, інфраструктура IoT продовжує невпинно розвиватися, поточні рішення щодо її безпеки не відповідають потребам мереж. DDoS-атаки залишаються великою загрозою, яка може заважати стабільній роботі IoT-мереж. Аби перекласти відповідальність самих

пристроїв щодо аутентифікації запитів, необхідно створювати додаткову мережу, або звертатися за допомогою до третіх компаній.

Блокчейн-технологія та публічна мережа Ethereum може допомогти у короткі терміни вирішити це питання. Система із смарт-контрактів зможе перейняти на себе відповідальність автентифікації запитів до IoT-мереж. Завдяки природі блокчейну, зловмисник не матиме можливості вплинути на дані, що зберігаються у мережі, а тому всі спроби реалізувати DDoS-атаку не матимуть ніякого впливу на IoT-мережу.

2 СПЕЦІАЛЬНА ЧАСТИНА

2.1 Складові моделі мережі

Модель IoT-мережі складається з наступних елементів:

- IoT-пристрої. Пристрої представляють з себе IoT-вузли, які відповідають за зондування, обробку та передачу даних через шлюз. Звичайний шлюз може використовуватися для підключення декількох пристроїв. Кожен пристрії має обмеження по газу, значення якого дорівнює, або є меншим за те, з яким може впоратися пропускна спроможність IoT та Ethereum мережа.

- Шлюз. Представляє собою шлюз загального призначення, до якого підключено декілька IoT-пристроїв. Шлюз забезпечує підключення до мережі усіх сусідніх пристроїв і може надавати додаткові можливості, такі як агрегація даних, або додаткові функції з безпеки. Різні шлюзи можуть використовуватися для різних типів пристроїв, або один шлюз може використовуватися для пристроїв різного призначення.

- Смарт-контракт. Представляє собою регулюючий орган, який відповідає за авторизацію пристроїв, а так слідкує за тим, щоб пристрій не був скомпрометованим. Як тільки пристрій почне посилати понаднормові за обсягом повідомлення, на сервер від смарт-контракту буде відправлятися повідомлення.

- Вузол блокчейн-мережі. У мережі Ethereum є безліч добровольців, які веріфікують транзакції та отримують за це винагороду. Майнери також відповідають за записи нових даних до сховища смарт-контракту, гарантуючи, що стан сховища не буде змінений несанкціоновано.

2.2 Схема IoT-мережі

Запронована модель захисту на основі блокчейн-технологій може бути використана у будь-якій IoT-мережі, саме тому у цьому розділі буде наведена типова модель мережі.

Типова IoT-мережа включає в себе пристрої для збору інформації, передавальні пристрої, та інформаційний сервер. Одна з найновітніших архітектур IoT-мережі, поділяє її на 3 загальні рівні [2, 4]: 1) Рівень сбору інформації 2) рівень обміну та передачі даних на сервер 3) рівень обробки та зберігання інформації. Розуміючи усі рівні цієї архітектури та використовуючи її як шаблон, можна розробити узагальнену схему захисту мережі.

2.2.1 Рівень збору інформації

Рівень збору інформації, або рівень пристроїв складається з фізичних об'єктів та сенсорних пристроїв, таких як камери спостереження, GPS-навігатори, усі види датчиків та інші. Основною функцією цього рівня є сприйняття та ідентифікація об'єктів та збір інформації. Цей рівень зазвичай займається ідентифікацією та збором специфічної інформації сенсорними пристроями. Залежно від типу датчиків, інформація може бути про місце розташування, температуру, вологість, вібрації або хімічні зміни у повітрі. Зібрана інформація потім передається рівень передачі інформації.

Збір даних проводиться великою кількістю різних пристроїв та для цього використовуються різні технології. Як основа будь-якої IoT-системи, підключені пристрої відповідають за надання найважливішого елементу IoT – даних. Для збору фізичних даних пристроям у першу чергу необхідні сенсори. Вони можуть бути впроваджені у пристрої, або бути реалізовані, як самостійні об'єкти для вимірювання та збору телеметричних даних. Наприклад, у сільськогосподарських системах використовуються датчики для вимірювання таких параметрів, як температура або вологість повітря, у системах «Розумне місто» датчики використовуються для вимірювання вібрацій або швидкості автомобілів на дорогах.

Ще одним незамінним елементом цього рівня є виконавчі механізми. Тісно працюючи із датчиками, ці механізми можуть реагувати на отримані дані якимись діями. Наприклад, на основі отриманих сигналів, системи розумного поливу у режимі реального часу аналізують ситуацію та визначають які водяні клапани слід відкрити для поливу. Клапани залишаються відкритими до тих пір, поки датчики не повідомлять про відновлення значень за замовчуванням. Очевидно, що все це відбувається без втручання людини.

Важливим є також те, що підключенні пристрої мають можливість не тільки двосторонньо взаємодіяти з відповідними шлюзами або системами збору даних, але також мати можливість розпізнавати та розмовляти між собою для збору та обміну інформацією та співпрацювати у режимі реального часу. Особливо у випадку обмежених ресурсів та пристроїв, що працюють від акумуляторів, досягнення взаємодії між ними є непростою задачею, оскільки такий зв'язок потребує великої обчислювальної потужності та споживає енергію та пропускну здатність. Саме використання блокчейн-технології може допомогти зекономити ресурси та забезпечити безпечний зв'язок між пристроями.

2.2.2 рівень обміну та передачі інформації

Цей рівень відповідає за транспортування даних від рівня збору інформації до прикладного рівня через різні канали зв'язку, наприклад бездротові або кабельні канали. Основні технології передачі включають у себе 2G, 3G, 4G, 5G, Bluetooth, залежно від підключених пристроїв. Також використовується багато різних протоколів, які визначають як інформація має бути підготовлена для передачі, як вона має передаватися та як її слід отримувати. Оскільки інформація зазвичай передається на великі відстані через усі можливі види передач, необхідні такі протоколи, які можуть забезпечити безперебійну передачу та отримання інформації, а також інструкції на випадок помилок у роботі. Функцією цього рівня є передача даних до серверів, де ця інформація оброблюється та використовується.

Хоча даний рівень і функціонує у безпосередній близькості від датчиків та виконавчих механізмів, слід описувати його як окремий рівень, оскільки він є «мостом» між пристроями та сервером та має вирішальне значення у архітектурі IoT-мережі. Через цей рівень проходить величезна кількість даних, яку можуть створювати пристрої, можливість збору, відбору та транспортування даних мають бути у центрі уваги. Як посередник між підключеними пристроями та серверами, шлюзи та системи зв'яку забезпечують необхідну точку зв'язку, яка пов'язує решту рівнів. Шлюзи полегшують зв'язок між датчиками та рештою системи, перетворюючи дані датчиків у формати, які легко передаються та використовуються іншими компонентами системи. Більше того, вони можуть контролювати, фільтрувати та відбирати дані, щоб мінімізувати обсяг даних, які необхідно надіслати на сервер, що позитивно впливає на витрати на передачу у мережі та на час відгуку. Таким чином, шлюзи разом із використанням блокчейну, забезпечують місце для локальної попередньої обробки та верифікації даних датчиків, які стискаються у пакети, готові до подальшої обробки. Ще одним аспектом, який підтримують шлюзи, є безпека. Оскільки шлюзи відповідають за управління потоку інформації у двох напрямках, за допомогою належних інструментів шифрування та захисту вони можуть запобігати витоку інформації, а також зменшити ризик зловмисних зовнішніх атак на пристрої.

2.2.3 Рівень обробки інформації

Збір, зберігання, аналіз та обробка даних, отриманих з рівня передачі інформації, здійснюється на рівні серверу, який включає в себе хмарні технології, загальні обчислення, обробку даних та мега бази даних. За допомогою цих технологій аналізуються та обробляються великі обсяги даних, які потім стають доступними для використання. Враховуючи кількість роботи, яку виконую цей рівень, є доцільним перекласти обов'язки з надання захисту іншим механізмам.

Сервер є мозком IoT-мережі. Центр обробки даних або хмарна система, призначена для зберігання, обробки та аналізу великих обсягів даних для глибшого розуміння, використовуючи потужні механізми аналізу даних та машинного навчання. Будучи важливим інструментом у мережі, сервер є об'єктом нападу зловмисників. Навіть враховуючи обчислювальні можливості серверів, часто вони не здатні самостійно впоратися із DDoS-атакою. Саме тому доцільним є використання додаткових механізмів, таких як блокчейн-технології.

Модель IoT-мережі складається з:

- декількох IoT-пристроїв;
- серверу для обміну даними;
- шлюзів для підключення пристроїв до серверу.

2.3 Модель DDoS-атак на IoT-мережу

Архітектура DDoS-атаки залежить від того, як саме зловмисник взаємодіє з пристроями. Зазвичай, архітектуру мережі можна поділити на чотири види, які використовують для здійснення DDoS-атаки: модель агента-обробника, модель рефлектора, модель на основі IRC, веб-модель:

- Модель агента-обробника. До цієї моделі входять клієнти (IoT-пристрої), обробники(сервер) та агенти. Зловмисник використовує клієнтів для спілкування з обробниками за допомогою програмних пакетів, розташованих у Інтернеті, які заважають мережі та передають інформацію від клієнтів агентам. Агент – це блок коду, який працює на скомпрометованій системі та проводить атаку на сервер. Агентом називається як скомпрометований пристрій, так і зловмисний код. Відповідно до налаштувань мережі, набір агентів може взаємодіяти як з одним, так і з декількома обробниками.
- Модель рефлектора. Ця модель схожа на попередню, але у ній також використовуються нескомпрометовані пристрої, які називаються рефлекторами. Через рефлекторів обробники можуть відправляти пакети

даних на сервер. Дуже часто обробники підробляють IP-адресу серверу, таким чином змушуючи рефлекторів надсилати на сервер відповіді. Це призводить до великої кількості мережевого трафіку, який адресований цільовому хосту. Рефлектором може бути будь-який пристрій або хост в Інтернеті, здатний відповідати на IP-запити, тому що зловмисники не потрібно компрометувати його. DDoS-атаки, які використовують таку модель атаку називаються DRDoS.

- Модель на основі IRC. Ця модель схожа на модель агента-обробника, з тією відмінністю, що клієнт підключається до агентів, які використовують канал зв'язку на основі IRC замість обробників. IRC – це текстовий клієнт-сервер протокол, який використовується для реалізації багатокористувацької та багатоканальної системи чату.

- Веб-модель. Ця модель схожа на попередню, але тут зв'язок базується на HTTP/HTTPS. Більше того, більшість клієнтів повністю налаштовані та контролюються за допомогою складних PHP-скриптів та зашифрованих комунікацій, тоді як агенти використовуються для звітності.

2.3.1 Вразливості, які використовуються DDoS-атаками

DDoS-атаки можуть використовувати різні вразливості, щоб загрожувати своїм жертвам. Враховуючи різні підходи, які може використовувати нападник, атаки можна поділити на дві категорії: зниження пропускної спроможності та атака на ресурси [9]:

- Зниження пропускної спроможності. Під час атак такого типу на сервер відправляється велика кількість пакетів даних, які є нібито законними, щоб зменшити пропускну здатність каналів зв'язку, а також обчислювальну спроможність серверу, заважаючи обробляти законні пакети даних. Такі атаки можна поділити на Flood-атаки та Amplification-атаки. У Flood-атаках зловмисник безпосередньо надсилає величезний обсяг трафіку на сервер, перенавантажуючи його ресурси та перешкоджаючи доступ законним

користувачам. Під час Amplification-атаки, агенти використовують рефлекторів у якості посередників. Flood-атаки є найбільш популярними, оскільки їх реалізація є досить простою, але в той же час ефективною. З іншого боку, Amplification-атаки користуються великою популярністю, оскільки невеликі DNS запити генерують набагато більші пакети відповідей, тому зловмисник здатен видавати себе за пристрої, підробляючи IP-адресу і замість пристрою відправляти велику кількість запитів. Через це, сервер буде перенавантажений великою кількістю цих запитів.

- Атака на ресурси. Ці атаки спрямовані на те, щоб перешкодити серверу відповідати на законні запити за допомогою вичерпання ресурсів і можуть бути охарактеризовані у експлуатації протоколів та атаках деформованих пакетів [1, 3]. Під час атак експлуатації протоколів, зловмисник намагається виявити помилки у реалізації протоколу або якоїсь функції та використати їх для вичерпання ресурсів серверу, в той час як атаки деформованих пакетів формують неправильні IP-пакети та відправляють їх від агентів до цілі, вказуючи одну й ту саму IP-адресу як джерело запиту та отримувача.

2.3.2 Вразливі рівні мережі та рівні автоматизації атак

DDoS-атаки розрізняються за рівнем TCP/IP протоколу, який використовується під час атаки: Мережевий рівень та рівень додатків. У DDoS-атаках мережевого рівня для здійснення атаки використовуються протоколи мережевого та транспортного рівнів. На рівні додатків атаки здійснюються для того, щоб вичерпати ресурси жертви(процесор, пам’ять, базу даних тощо). Прикладом атаки на рівні додатків є DNS Water Torture, яка націлена на авторитетні DNS-сервери, які не напряму пошкоджуються через надсилання величезного обсягу запитів до Open Resolvers. Ці запити перенаправляються до кешу DNS-серверів, а потім до авторитетних DNS-серверів. Незважаючи на те, що ціллю таких атак є останні сервери, інші також страждають через те, що їм доводиться витрачати свої ресурси на обробку цих запитів.

Спираючись на рівень автоматизації DDoS-атак, їх можна поділити на ручні, напівавтоматичні та автоматичні:

- Ручні. Під час таких DDoS-атак зловмисник сам сканує пристрої у пошуках вразливостей. Після виявлення вразливості, зловмисник самостійно проникає у пристрій, впроваджує зловмисний код для атаки, а потім вручну командує атакою. На сьогоднішній день, таких атак майже не існує, оскільки усі етапи атак вдалося автоматизувати.

- Напівавтоматичні. Під час таких атак, етапи пошуку пристроїв, їх експлуатація та впровадження коду є автоматизованими. Єдиний неавтоматизований етап – це початок атаки. Зловмисник сам вказує пристроям тип атаки, час початку, тривалість атаки та ціль. Враховуючи комунікаційні механізми, які використовуються між зловмисником та пристроєм, DDoS-атака може бути виконана за допомогою прямої комунікації(Модель агента-обробника) та непрямої комунікації(Модель на основі IRC).

- Автоматична. Під час таких атак, усі етапи є автоматизованими, тому потреба у комунікації між зловмисником та пристроями відсутня. Час початку, тип, тривалість та ціль нападу запрограмовані в коді атаки. У цій категорії атак від зловмисника потрібне мінімальне втручання, оскільки він бере участь лише у видачі команди, яка починає атаку. У напівавтоматичних та автоматичних атак компрометування пристроїв за допомогою стратегій автоматичного сканування та методів розповсюдження. Слід також відмітити, що у деякий атаках використовується змішаний підхід: сканування пристроїв та атака можуть бути автоматизовані, в той час як експлуатація та компрометування можуть проводитися вручну.

2.3.2 Стратегія пошуку пристрою зловмисником

На етапі пошуку пристроїв, зловмисник виявляє як можна більше вразливих пристроїв мережі за допомогою сканування мережі. DDoS атаки за природою сканування можна поділити на п’ять класів [1, 3]: Випадкове сканування, сканування списку записів, сканування покажчиків, зміщене сканування, сканування локальної підмережі:

- Випадкове сканування. Під цією стратегією розуміється, що кожен скомпрометований пристрій використовує різний сід для перевірки випадкових адрес у IP-просторі адрес. Ще один підхід чистого сканування, коли випадковим чином шукають такі IoT-пристрої, які оснащені типовими обліковими даними для входу.
- Сканування списку записів. За допомогою цієї стратегії апарат сканування, який використовує зловмисник, має зовнішній список можливих для перевірки пристроїв. Як тільки зловмисник виявляє та заражає новий пристрій, він пересилає частину початкового списку, щоб мати високу швидкість розповсюдження та відсутність колізії під час сканування.
- Сканування покажчиків. Деякі відомості, які зберігаються на скомпрометованих пристроях, можуть використовуватися зловмисником для пошуку нових пристроїв. Так само, як черв електронної пошти можуть використовувати інформацію з адресних книг заражених машин, або черви веб-сервісів, які вражають кожного клієнта, який має доступ до веб-сторінки, так саме черв у IoT-системі може поширюватися за допомогою відомостей, які зберігаються на пристроях.
- Зміщене сканування. Під час цієї стратегії спочатку використовується стратегія сканування покажчиків, аби додати невелику кількість пристроїв до бот-мережі. Після цього, усі заражені пристроїв мають спільну псевдовипадкову перестановку у просторі IP-адрес, і кожна адреса відображається в індексі у цій перестановці. Пристрій, заражений під час стратегії сканування покажчиків, починає сканування через перестановку, використовуючи індекс, як початкову точку. Кожного разу, коли він виявляє пристрій, який вже був заражений, він обирає нову початкову точку.

- Сканування локальної підмережі. Ця стратегія може використовуватися у будь-якій із вище названих стратегій, включаючи сканування пристроїв, які знаходяться у одній підмережі разом із скомпрометованим пристроєм. Цей прийом дозволяє одній копії програми сканування поставити під загрозу багато вразливих пристроїв.

2.3.4 Методи розповсюдження атаки та вплив на мережу

Після вербування пристроїв та їхньої експлуатації, пристрій-агент заражається кодом атаки і на основі обраного на цьому етапі механізму DDoS-атаки можна класифікувати наступним чином [1, 3]: поширення за допомогою центрального джерела, зворотньо-ланцюгове поширення та автономне поширення.

- Поширення за допомогою центрального джерела. Під час такої DDoS-атаки програмний код для атаки зберігається на центральному сервері зловмисника і завантажується за допомогою механізмів передачі файлів кожного разу, коли скомпрометовано новий пристрій.
- Зворотній ланцюг дозволяє додати до скомпрометованого пристрої код атаки для його подальшого розповсюдження. Скомпрометований пристрій стає ще одним джерелом для розповсюдження атаки. Цей механізм є більш довговічним, ніж попередній, оскільки є більш децентралізованим, а отже не має єдиної точки відмови.
- Автономне поширення. Під час такого підходу, додаткові файли не до завантажуються на скомпрометовані пристрої. Замість цього, інструкції атаки вводяться у пристрій безпосередньо під час фази експлуатації, таким чином зменшуючи можливість виявлення атаки.

Залежно від впливу DDoS-атак на мережі, їх можна віднести до однієї із двох категорій, а саме [3, 5]: руйнівна та деградуюча:

- Руйнівна DDoS-атака. Під час такого типу атаки, метою зловмисника є повність відмовити законним користувачам у використані мережі. Нині більшість атак відноситься до руйнівних атак. Основуючись на

можливості динамічного відновлення після DDoS-атак, їх можна додатково поділити на динамічно та нединамічно відновлювальні. Після динамічно відновлювальних атак, сервер здатен автоматично відновити свої сервіси, в той час як нединамічно відновлювальні означають, що сервер не зможе продовжити свою роботу без втручання людини. Це означає, що сервери необхідне ручне перезавантаження, а деяких випадках і переналаштування.

- Деградуюча DDoS-атака. Цей тип атаки спрямований на споживання Деякої частини ресурсів серверу, не спричиняючи повної відмови сервісів, аби деякий час залишатися непоміченою. Тим не менш, шкода від таких атак може бути величезною. Така атака може призвести до відмови в обслуговувані деяким користувачам у періоди особливо високого завантаження, а продуктивність сервісів може зменшитися і навіть стати меншою за середньо допустиму.

2.4 Вразливості IoT-мережі

На сьогоднішній день IoT-технології невпинно розвиваються, проте вони й досі є вразливими, а підходи із захисту IoT-мереж не є досконалими. Вразливості IoT-мережі можна поділити за наступним списком:

- Недостатні можливості пристроїв IoT та навколишнього середовища: недостатньо прикладені зусилля щодо захисту одного чи декількох рівнів IoT у типічній мережі є одніє з основних вразливостей. Такі міри захисту є вирішальними для всіх аспектів системи IoT, таких як мережеві служби, Web, API або мобільні інтерфейси, механізми оновлення. Крім того, використання застарілих або небезпечних компонентів додають ризик, який можна пом’якшити, обравши більш відповідні компоненти IoT.

- Недостатній фізичний захист: сильний акцент на електронній безпеці відсутній при розгляді систем IoT. Хоча традиційні послуги Інтернету як правило розташовані у захищених центрах обробки даних з озброєною охороною, відокремленим фізичним доступом, біометричними даними та низкою інших заходів фізичної безпеки, значна частина компонентів IoT

знаходиться у фізичному доступі не лише споживачів, але й зловмисників. Необхідні спільні зусилля, аби зменшити ймовірність того, що IoT-мережа може бути скомпрометована через фізичну доступність.

- Слабка автентифікація та механізми конфіденційності: проектування механізмів автентифікації потребує уважності, а також урахування потреб конфіденційності користувачів системи. Оскільки відомо, що рівень збору інформації обов'язково повинен бути розміщений у межах публічного доступу, заходи автентифікації повинні враховувати спроби маніпулювати компонентами, намагаючись отримати доступ до системи або вплинути на неї. Крім того, повинні бути вжиті заходи щодо розподілу та захисту даних користувачів і, можливо, навіть перешкоджати використанню системи приватними особами, де лише підтвердження використання системи може представляти інші ризики для користувачів.
- Слабкий моніторинг та управління системою: Незалежно від початкових зусиль з розробки та впровадження безпечної системи Інтернету речей, необхідно докласти спільних зусиль для постійного моніторингу, оцінки та модернізації системи. Система IoT буде невпинно змінюватися з часом із появою нових функцій та знаходженням вразливостей у використовуваних протоколах, на які спирається система, що також призводить до їх модифікації та модернізації системи. Неадекватні дії щодо управління мережею майже напевне призводять до негативних наслідків, таких як погіршення стану або навіть повний збій у системі, викриття конфіденційної інформації або навіть втрати контролю над системою.

Iot-мережі представляють нові виклики в і без того складному середовищі безпеки в Інтернеті. Збільшення числа підключених до мережі пристроїв уже представляє значну небезпеку для мережі. Крім того, розвиток мережі регулярно забезпечує зв'язок з новими класами пристроїв, кожен з яким може представляти набір унікальних проблем та вразливостей, з якими доведеться

боротися. В той час, як деякі з них можуть бути очевидними, інші можуть бути складними та потребувати детальнішого аналізу.

Вразливості мережі можуть відрізнятись залежно від рівня, у якому ця вразливість реалізована. Зрозуміло, що фізичні пристрої вразливі до багатьох можливих способів атак через їхню фізичну доступність для зловмисників. Зловмисник може отримати фізичний доступ до пристрою, підробити та, можливо , скомпрометувати пристрій для отримання доступу до всієї системи. Рівень передачі даних також є об'єктом для атак, оскільки засобом зв'язку є Інтернет, який, як відомо, вразливий до багатьох видів атак на протоколи, починаючи від добре відомих атак, і завершуючи невідомими атаками нульового дня. Менш поширеними є методології атак, які прагнуть порушити збір та цілісність даних, проте такі рівні все ще страждають від помилок у програмному забезпеченні, які можуть ховатися у програмному коді та пристроях. Погано написане програмне забезпечення все ще може стати об'єктом SQL-ін'єкцій та інші типи атак, які раніше використовувалися для проникнення у середовище даних. Програмне забезпечення все ще страждає від таких проблем, тому їх слід ретельно перевіряти, намагаючись пом'якшити атаки, які використовують переваги погано розробленого програмного забезпечення та обладнання. Найкращі проектні рішення ті, які не покладаються на гостру взаємодію із людиною з точки зору безпеки.

Хоча й існує велика кількість різноманітних атак, усі вони, як правило, підпадають під одну чи декілька категорій. Черви, як правило, класифікуються їх здатністю поширюватися по мережі, використовуючи один або декілька сценаріїв реплікації. Атаки відмови в обслуговуванні, як в одиничній, так і у розподіленій формі, як правило характеризуються захопленням ресурсів, необхідних для ефективної роботи системи. SQL-ін'єкції здійснюються за допомогою вставлення шкідливого коду у системні входи, намагаючись змінити поведінку програми. Обман – це процес видавання себе за іншу особу з метою викрадення або отримання незаконного доступу до системи. Підслуховування є методом перехоплення та використання системного

зв’язку. Перешкоди – це метод втручання у бездротовий канал зв’язку за допомогою трансляції шуму в смугу частот, що використовується системою. Шкідливе програмне забезпечення є програмним кодом, який бере контроль над одним чи декількома компонентами системи, щоб взяти під контроль усю систему. Метод грубої сили характеризується спробою отримати доступ шляхом процесу масового повторення в спробі отримати несанкціонований доступ. Зворотна інженерія – це процес вивчення системи для того, щоб визначити, як найкраще її використовувати, як правило, за допомогою аналізу різних компонентів системи. У таблиці 2.1 наведено усі загрози, а також рівні мережі, у яких вони можуть бути реалізовані.

Таблиця 2.1 – Загрози IoT-мережі

Атака	Опис	Рівень мережі
Черв	Атака, здатна порушити різні компоненти мережі за допомогою обходу системи	Рівень передачі інформація

Продовження таблиці 2.1

DDoS	Порушення атаки за допомогою виснаження ресурсів системи	Рівень збору інформації, рівень передачі інформації
SQL-ін’єкція	Погіршення цілісності системи при неправильній вставці даних	Серверний рівень
Обман	Проникнення в систему за допомогою маскування або підроблених даних	Рівень збору інформації, рівень передачі інформації, серверний рівень
Підслуховування	Несанкціоноване перехоплення системних даних	Рівень збору інформації, рівень передачі інформації, серверний рівень
Перешкоди	Запобігання належній роботі системи	Рівень збору інформації, рівень передачі інформації

Продовження таблиці 2.1

Зловмисне ПЗ	Введення зловмисного програмного коду з метою з метою порушення належної роботи системи	Рівень збору інформації, рівень передачі інформації, серверний рівень
Метод грубої сили	Повторні спроби вторгтися у систему шляхом повторних спроб	Рівень збору інформації, рівень передачі інформації, серверний рівень
Зворотня інженерія	Аналіз системи з метою знайдення вразливих місць	Рівень збору інформації, серверний рівень

2.5 IoT-Blockchain модель

Як вже було зазначено, мережа Ethereum – одна з найбільших програмних платформ, яка дозволяє будувати смарт-конракти та децентралізовані додатки на основі блокчейну. Для IoT-мережі буде розроблений смарт-контракт, який зможе верифікувати пристрої, а у випадку підозрілих дій повідомляти про це сервер IoT-мережі.

Для роботи даної системи захисту, для пристрою буде створена публічна адреса у блокчейні. Кожна адреса пристрою буде записана у смарт-контракті. Верифікація пристроїв у блокчейні відбуватиметься завдяки крипттографічним властивостям приватного ключа, яким пристрій підписуватиме транзакції. Контракт запрограмований для розмежування між надійними та ненадійними пристроями. Перш за все, пристрої повинні будуть зареєстровані у контракті разом із наданим лімітом по газу. В процесі реєстрації для пристрою буде створено новий аккаунт, який буде записаний у смарт-контракт разом із обмеженням по газу, яке буде пов’язане із

специфікацією пристрою, тобто значення буде залежати від пропускної здатності шлюзу та вимогами до ресурсів пристрою. Взаємодія між пристроями та сервером відбуватиметься лише через смарт-контракт. Якщо ж пристрій було скомпрометовано, то будь-які спроби відправляти великі обсяги даних будуть заблоковані смарт-контрактом, а до серверу IoT-мережі буде відправлене повідомлення, у якому буде зазначено про можливу DDoS-атаку та про скомпрометовані пристрої.

Рисунок 2.1 – Блокування скомпрометованого пристрою смарт-контрактом

2.6. Огляд смарт-контракту

Смарт-контракт у системі відповідає за безпечний зв'язок між пристроями та розподіленими серверами. Він розроблений із використанням мови програмування Solidity. Solidity – це високорівнева мова програмування, орієнтована на написання смарт-контрактів для Ethereum. Він має дві функціональні фази: ініціалізація та розгортання:

- Ініціалізація: на цьому етапі сервер IoT-мережі посилає транзакцію до блокчейн-вузла та розгортає смарт-контракт у мережі, який повертає адресу смарт-контракту, завдяки якій можна буде звертатися до його функцій. Ця адреса буде розіслана серед усіх пристроїв для того, щоб вони могли взаємодіяти з контрактом. Ліміт газу для кожного пристрою також виставляється сервером і при необхідності може бути змінений.

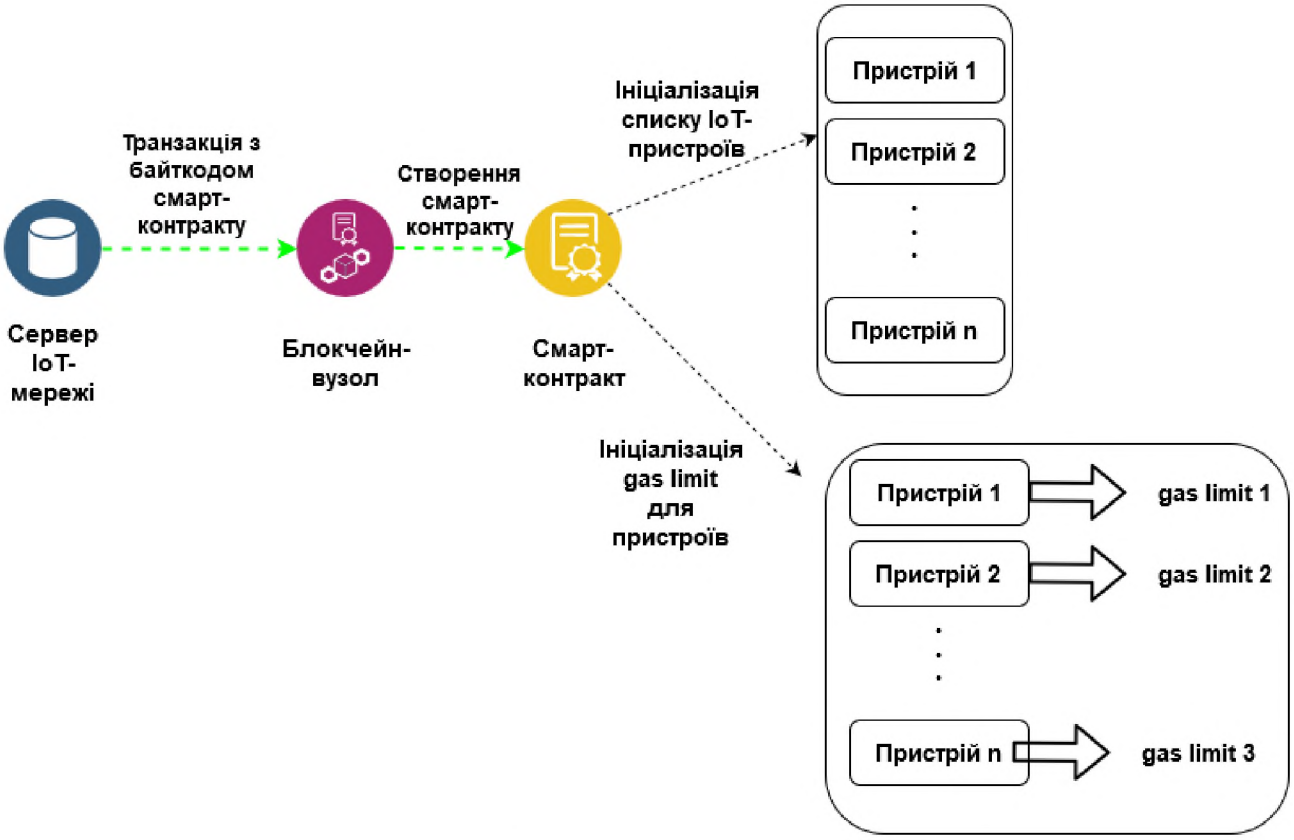

Рисунок 2.2 – Ініціалізація смарт-контракту

- Використання контракту: Усі IoT-пристрої у системі матимуть можливість відправляти транзакції до смарт-контракту. Це необхідно для того, щоб верифікувати пристрій. Якщо адреса пристрою знаходиться у списку легітимних пристроїв у смарт-контракті, та кількість викликів не перевищує gas limit, то пристрій вважається верифікованим. Лише після такої перевірки він зможе відправити дані до серверу. Якщо ж пристрій відсутній у списку, або кількість транзакцій перевищує gas limit, то пристрій вважається скомпрометованим. У такому випадку, відправлення даних на сервер забороняється, а смарт-контракт, у свою чергу, повідомляє сервер IoT-мережі

про можливу DDoS-атаку. Алгоритм, використаний у функції смарт-контракту, зображено на рисунку 2.3. У алгоритмі видно, що якщо існує повідомлення від пристрою, то пристрій перевіряється на валідність, і лише після цього повідомлення відправляється на сервер. Використовуючи список авторизованих пристроїв, який ведеться у смарт-контракті, контракт зможе ідентифікувати кожен пристрій.

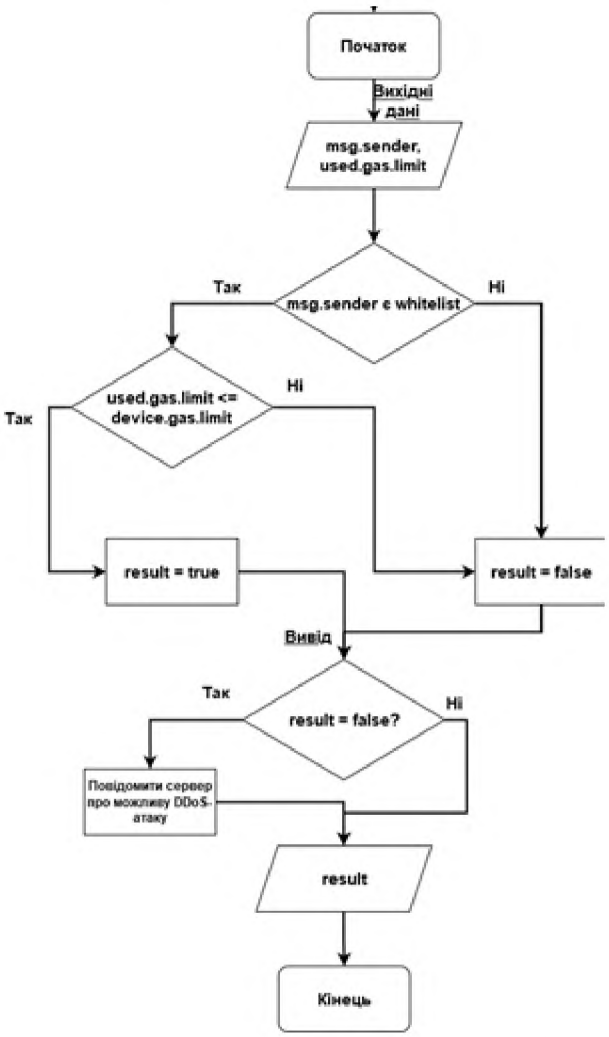

Рисунок 2.3 – Блок-схема методу валідації пристрою

2.7 Захист від DDoS-атак

Основна проблема DDoS-атаки, описана у цій роботі, являє собою те, що один чи декілька IoT-пристроїв можуть одночасно відправляти надзвичайно велику кількість повідомлень для того, щоб перевантажити сервер та виснажити його ресурси. У схемі атаки будь-які з пристроїв можуть почати постійно завантажувати дані на сервер та ініціювати DDoS-атаку. Такі атаки

можна попередити за допомогою такого атрибуту, як обмеження за газом у мережі Ethereum, оскільки це обмеження гарантує, що кількість використаних ресурсів не буде перевищувати ліміту. У запропонованому смарт-контракті ліміт газу встановлюється для будь-якої операції та пристрою, які оброблюються через нього. Такий механізм діє як запобігання перевантаженню системи.

Припустимо, що існує n IoT-пристроїв, кожен з яких має g ліміт газу. Максимальна пропускна здатність на сервері – B, тоді:

$$\sum_{i=1}^{n} g_i \leq B. \qquad (2.1)$$

Із рівняння 2.1 видно, що навіть у разі одночасного надсилання даних з усіх пристроїв, перевантаження серверу буде неможливе через встановлене обмеження і пропускна здатність серверу не буде вичерпана. Більше того, будь-яка DDoS-атака , яка має намір вичерпати ресурси серверу, спочатку почне вичерпувати власні ресурси до того моменту, коли ліміт газу пристрою буде вичерпано. Це обмеження припинить роботу зловмисного пристрою у рамках надсилання даних до серверу одразу, як тільки ліміт газу буде досягнено, і запобіжить перенавантаженню серверу. Ліміт газу встановлюється індивідуально для кожного пристрою, зважаючи на його потрібності та пропускну здатність серверу. Ліміт встановлюється у момент реєстрації пристрою у смарт-контракту і у разі необхідності, може бути змінений сервером.

Можливість створювати та розгортати смарт-контракти у Ethereum усуває випадків, коли програма може припиняти свою роботу завдяки тому, що блокчейн-мережа може працювати постійно. Ця вразливість притаманна централізованому підходу і майже повністю вирішена у децентралізованому. Таким чином, Iot-мережа з використанням блокчейн технології може стати потенційним вирішенням проблем, пов’язаних із автентифікацією, довірою, та єдиною точкою відмови, з якими нині стикаються централізовані підходи до забезпечення безпеки IoT-мереж.

2.8 Зв'язок IoT-мережі з блокчейном

Будь-яка дія у блокчейні проводиться за допомогою транзакції. Вузол Ethereum мережі верифікує транзакцію та додає до блоку. Виклик методу смарт-контракту також є транзакцією, яку має відправити IoT-пристрій, підписавши її своїм приватним ключем. Для підправлення цих транзакцій у мережі Ethereum використовується протокол віддаленого виклику процедур JSON-RPC. Особливістю цього протоколу є використання JSON-формату для кодування повідомлень. JSON-RPC може відправляти запити на сервер, який реалізує протокол. У випадку з Ethereum, сервером є вузол, який отримує транзакцію, верифікує та додає її до блоку, а потім розповсюджує блок по іншим вузлам. Відправником запиту є IoT-пристрої, які викликають необхідний метод смарт-контракту за допомогою транзакції. У якості вхідних параметрів може передаватися безліч вхідних параметрів, включаючи масиви даних. Для верифікації пристрою, до смарт-контракту необхідно відправити блокчейн-адресу пристрою та gas limit. Пристрій заздалегідь підписує транзакцію та відправляє її до шлюзу разом із даними. Шлюз відправляє цю транзакцію за допомогою протоколу JSON-RPC до віддаленого вузлу та отримує від смарт-контракту відповідь, на основі якої вирішує, що робити з даними, які відправлені вузлом. Якщо IoT-пристрій не знаходиться у white-list смарт-контракту, або відправляє дані занадто часто, то вузол не відправляє дані до серверу. Замість цього, він відправляє до серверу повідомлення, щодо підозрілих дій IoT-пристрою.

Для того, щоб майнери могли отримати транзакцію та верифікувати її, IoT-мережі необхідний свій власний вузол, який зможе розповсюдити цю транзакцію майнерам. Для цього існує два варіанти: розгорнути власний вузол, або звернутись до спеціального сервісу, який надасть свої власні вузли. Процес розгортання та підтримки власного вузла є досить трудомістким та ресурсозатратним. У разі, коли кількість вузлів є великою, один звичайний вузол може не впоратися із кількістю запитів, а підтримка великої кількості

Ethereum-вузлів не є оптимальним завдання для IoT-мережі. Саме тому існують спеціальні сервіси, які тримають свої власні вузли та дозволяють свої клієнтам відправляти транзакції до Ethereum без будь-яких проблем. Одним з таких сервісів є Infura – це один з найпопулярніших та надійних сервісів з надання блокчейн-вузлів. В залежності від кількості запитів, які необхідно відправляти клієнту, Infura надає різні тарифи. Цього буде більш ніж достатньо для середньостатичної IoT-мережі. Пристрої зможуть відправляти транзакції до вузлів Infura за допомогою спеціального ключа, а рештою процесу, пов'язаною з верифікацією транзакції та повернення відповіді від блокчейну до IoT-мережі займатиметься Infura. Таким чином, зв'язок між двома мережами буде оптимальним, забезпечуючи достатню пропускну спроможність та конфіденційність транзакцій.

2.9. Висновок

У спеціальній частині було проаналізовано вразливості IoT-мережі, розглянуті різні варіанти реалізації DDoS-атак. Була розроблена система захисту на основі блокчейну та мережі Ethereum, реалізовано механізм верифікації пристрою та захисту від потенційної DDoS-атаки. Забезпечено стабільний зв'язок між блокчейн-мережею та пристроями, який не додав додаткового навантаження на IoT-мережу.

Розроблена система захисту відповідає вимогам безпеки IoT-мережі, є повність автоматизованою та потребує мінімального втручання людини.

3 ЕКОНОМІЧНИЙ РОЗДІЛ

Розробка системи захисту типової IoT-мережі від DDoS-атак на базі Ethereum потребує обґрунтування економічної її доцільності, виходячи з аналізу витрат на розробку та впровадження. Тому метою економічного розділу є здійснення відповідних розрахунків, які дозволять встановити економічного ефекту від впровадження та налагодження системи захисту від DDoS-атак.

3.1 Розрахунок (фіксованих) капітальних витрат

Капітальні інвестиції – це кошти, призначені для створення і придбання основних фондів і нематеріальних активів, що підлягають амортизації.

До капітальних витрат належать витрати на розробку системи захисту, написання програмного коду та його запровадження до IoT-мережі, які визначаються виходячи з трудомісткості розробки системи захисту.

Визначення трудомісткості розробки політики безпеки інформації

Трудомісткість розробки системи захисту від DDoS-атак визначається тривалістю кожної робочої операції, починаючи з складання технічного завдання і закінчуючи оформленням документації (за умови роботи одного спеціаліста з інформаційної безпеки):

$$t = t_{тз} + t_{в} + t_{а} + t_{вз} + t_{озб} + t_{овр} + t_{д}, \text{ годин,}$$

де $t_{тз}$ – тривалість складання технічного завдання на розробку політики безпеки інформації;

$t_{в}$ – тривалість розробки концепції безпеки інформації у мережі;

$t_{а}$ – тривалість процесу аналізу ризиків;

$t_{вз}$ – тривалість визначення вимог до заходів, методів та засобів захисту;

$t_{озб}$ – тривалість вибору основних рішень з забезпечення безпеки інформації;

$t_{овр}$ – тривалість організації виконання відновлювальних робіт і забезпечення неперервного функціонування мережі;

$t_д$ – тривалість документального оформлення системи захисту.

Визначено, що відповідно до етапів розробки політики безпеки інформації, тривалість операцій склала наступні величини:

$t_{тз}$=20 годин, $t_в$=35 годин, $t_{тз}$=20 годин, $t_{вз}$=18 годин, $t_{озб}$=10 годин, $t_{овр}$=8 годин, $t_д$=6 годин.

t =20 + 35 + 20 + 18 + 10 + 8 + 6 = 117 годин.

Розрахунок витрат на створення системи захисту від DDoS-атак

Витрати на розробку системи захисту від DDoS-атак Крп складаються з витрат на заробітну плату спеціаліста з інформаційної безпеки та блокчейн розробника Ззп і вартості витрат машинного часу, що необхідний для розробки системи захисту Змч.

$$К_{рп} = З_{зп} + З_{мч}.$$

$К_{рп}$ = 95760 + 698.49 = 96458,49 грн.

$З_{зп} = t\, З_{пр}$.

$З_{зп}$ =117·260 = 30420 грн.

Вартість машинного часу для розробки системи захисту IoT-мережі на ПК визначається за формулою:

$З_{мч}= t \cdot С_{мч}$

$З_{мч}$ =117·5,97=698,49 грн,

де t – трудомісткість підготовки документації на ПК, годин;

$С_{мч}$ – вартість 1 години машинного часу ПК, грн./година.

Вартість 1 години машинного часу ПК визначається за формулою:

$$Смч = 0{,}9 \cdot 3 \cdot 1{,}64 + \frac{3800 \cdot 0{,}4}{1920} + \frac{7200 \cdot 0{,}2}{1920} = 5{,}97 \text{ грн.}$$

Розробка системи захисту на базі блокчейну також потребує розгротання смарт-контракт у Ethereum-мережі. Як і будь-яка інша транзакція у Ethereum,

розгортання контракту потребує одноразової виплати майнерам. Вартість розгортання смарт-контракту становила 351 грн.

Таким чином, капітальні (фіксовані) витрати на створення системи захисту IoT-мережі на базі Ethereum становила:

$$К = К_{рп} + К_{роз} = 96458,49 + 351 = 96809,49 \text{ грн},$$

де $К_{пр}$ – вартість розробки проекту інформаційної безпеки та залучення для цього зовнішніх консультантів, тис. грн;

$К_{роз}$ – вартість розгортання смарт-контракту у мережі Ethereum.

3.1.1 Розрахунок поточних витрат

Річні поточні витрати на функціонування системи захисту IoT-мережі складають:

$$С = С_в + С_к + С_{ак}, \text{ грн.}$$

де $С_в$ - вартість відновлення й модернізації системи ($С_в = 0$);

$С_к$ - витрати на керування та функціонування системи в цілому;

$С_{ак}$ - витрати, викликані активністю користувачів IoT-мережі ($С_{ак} = 0$ грн.).

Витрати на керування IoT-мережею ($С_к$) складають:

$$С_к = С_н + С_а, \text{ грн.}$$

Витрати на навчання адміністративного персоналу й кінцевих користувачів визначаються ($С_н = 0$ грн.). Для обслуговування даної системи не потрібен постійний персонал.

Для функціонування IoT-блокчейн мережі необхідні архівні вузли Ethereum-мережі. Для цього було прийнято рішення звернутися до відповідного сервісу з надання архівних вузлів. Відповідно до вимог типової IoT-мережі, та середньої кількості транзакцій, які IoT-пристрої відправлятимуть до блокчейну, будо прийнято рішення обрати сервіс Infura та їхній тариф «Team». Отже, Річні витрати на послуги сервісу Infura за використання архівних вузлів за тарифом «Team» складає 70000 грн.

Для функціонування системи, на рахунках пристроїв повинна знаходитися певна сума криптовалюти ефір. Середня кількість ефіру для одного пристрою на рік складає 1022 грн. Середня IoT-мережа може налічувати приблизно 30 пристроїв. Отже сума, необхідна для купівлі ефіру – 30660.

Отже, витрати на керування системою захисту IoT-мережі на базі блокчейну визначається:

$$С_к = 0 + 100660 \text{ грн.}$$

Таким чином, річні поточні витрати на функціонування системи захисту IoT-мережі від DDoS-атак на основі блокчейну складають 100660 грн.

3.2 Оцінка можливого збитку від DDoS-атаки на типову IoT-мережу

3.2.1 Оцінка величини збитку

Для розрахунку вартості такого збитку можна застосувати наступну спрощену модель оцінки для типової IoT-мережі.

Необхідні вихідні дані для розрахунку:

$t_п$ – час простою мережі, 10 год.;

$t_в$ – час відновлення після атаки персоналом, що обслуговує IoT-мережу, 5 годин;

$t_{ви}$ – час повторного введення загубленої інформації співробітниками атакованої IoT-мережі, 8 годин;

$З_о$ – заробітна плата обслуговуючого персоналу (адміністраторів та ін.), 15000 грн./міс.;

$З_с$ – заробітна плата співробітників атакованого вузла або сегмента корпоративної мережі, 12000 грн./міс.;

$Ч_о$ – чисельність обслуговуючого персоналу (адміністраторів та ін.), 2 особи;

$Ч_с$ – чисельність співробітників атакованої IoT-мережі, 10 осіб.;

О – обсяг прибутку атакованої IoT-мережі, 1500000 тис. грн. у рік;

$П_{зч}$ – вартість заміни встаткування або запасних частин, грн;

I – число атакованих сегментів корпоративної мережі, 1;

N – середнє число атак на рік, 40.

Упущена вигода від простою атакованого сегмента корпоративної мережі становить:

$$U = П_п + П_в + V, \quad (3.1)$$

де $П_п$ – оплачувані втрати робочого часу та простою співробітників атакованої IoT-мережі, грн;

$П_в$ – вартість відновлення працездатності IoT-мережі (переустановлення системи, зміна конфігурації та ін.), грн;

V – втрати від зниження обсягу продажів за час простою атакованої IoT-мережі, грн.

Втрати від зниження продуктивності IoT-мережі являють собою втрати їхньої заробітної плати (оплата непродуктивної праці) за час простою внаслідок атаки:

$$П_п = \frac{\sum З_с}{F} \cdot t_n = 15000 \cdot \frac{10}{176} \cdot 10 = 6818.1 \text{ грн},$$

де F – місячний фонд робочого часу (при 40-а годинному робочому тижні становить 176 ч).

Витрати на відновлення працездатності вузла або сегмента корпоративної мережі включають кілька складових:

$$П_в = П_{ви} + П_{пв} + П_{зч},$$

де $П_{ви}$ – витрати на повторне уведення інформації, грн.;

$П_{пв}$ – витрати на відновлення IoT-мережі вцілому, грн;

$П_{зч}$ – вартість заміни устаткування або запасних частин, грн.

Витрати на повторне введення інформації $П_{ви}$ розраховуються виходячи з розміру заробітної плати співробітників атакованої IoT-мережі $З_с$, які зайняті повторним введенням втраченої інформації, з урахуванням необхідного для цього часу $t_{ви}$:

$$П_{ви} = \frac{\sum З_с}{F} \cdot t_{ви} = 12000 \cdot \frac{10}{176} \cdot 8 = 5454.5 \text{ грн.}$$

Витрати на відновлення IoT-мережі $П_{пв}$ визначаються часом відновлення після атаки $t_в$ і розміром середньогодинної заробітної плати обслуговуючого персоналу (адміністраторів):

$$П_{пв} = \frac{\sum З_о}{F} \cdot t_в = 12000 \cdot \frac{2}{176} \cdot 5 = 1090.9 \text{ грн.}$$

$$П_в = 5454.5 + 1090.9 = 6545.4 \text{ грн.}$$

Втрати від зниження очікуваного обсягу прибутків за час простою атаковано IoT-мережі визначаються виходячи із середньогодинного обсягу прибутку і сумарного часу простою сегмента корпоративної мережі:

$$V = \frac{O}{Fг} \cdot (t_П + t_В + t_{ВИ}),$$

$$V = \frac{15000000}{2080} \cdot (10 + 5 + 8) = 16586.5 \text{грн.}$$

де $F_г$ – річний фонд часу роботи філії (52 робочих тижні, 5-ти денний робочий тиждень, 8-ми годинний робочий день) становить близько 2080 год.

$$U = 6818.1 + 6545.4 + 16586.5 \text{грн} = 29950 \text{ грн.}$$

Таким чином, загальний збиток від атаки на типову IoT-мережі за формулою 3.1 складе:

$$B = \sum i \sum n \; U = 1 \cdot 40 \cdot 16586.5 = 663460 \text{ грн.}$$

3.2.2 Загальний ефект від впровадження системи захисту на основі блокчейну

Загальний ефект від впровадження системи захисту мережі визначається з урахуванням ризиків порушення інформаційної:

$$E = B \cdot R - C \text{ грн.,} \qquad (3.2)$$

де B – загальний збиток від атаки у разі перехоплення інформації, тис. грн.;

R – вірогідність успішної реалізації атаки на сегмент мережі, частки одиниці (35%);

C – щорічні витрати на експлуатацію системи захисту.

Загальний ефект від впровадження системи захисту IoT-мережі від DDoS-атак на основі блокчейну визначається з урахуванням ризиків порушення інформаційної безпеки за формулою 3.2:

$$E = 663460 \cdot 0{,}35 - 100660 = 131551 \text{ грн}$$

3.3 Визначення та аналіз показників економічної ефективності системи інформаційної безпеки

Коефіцієнт повернення інвестицій ROSI показує, скільки гривень додаткового прибутку приносить одна гривня капітальних інвестицій на впровадження системи захисту від DDoS-атак:

$$ROSI = \frac{E}{K}, \quad \text{частки одиниці,}$$

де E – загальний ефект від впровадження системи інформаційної безпеки грн.;

К – капітальні інвестиції за варіантами, що забезпечили цей ефект, грн.

Коефіцієнт повернення інвестицій ROSI:

$$ROSI = \frac{131.551}{96809{,}49} = 1.35, \quad \text{частки одиниці,}$$

Проект визнається економічно доцільним, якщо розрахункове значення коефіцієнта повернення інвестицій перевищує величину річної депозитної ставки з урахуванням інфляції:

$$ROSI > (N_{деп} - N_{інф})/100),$$

де $N_{деп}$ – річна депозитна ставка, (18 %);

$N_{інф}$ – річний рівень інфляції, (11%).

Розрахункове значення коефіцієнта повернення інвестицій:

$$1.35 > (18 - 11)/100 = 1.35 > 0{,}07.$$

Термін окупності капітальних інвестицій T_o показує, за скільки років капітальні інвестиції окупляться за рахунок загального ефекту від впровадження системи захисту:

$$T_o = \frac{K}{E} = \frac{1}{ROSI} = \frac{1}{1.35} = 0.74 \text{ років} = 9 \text{ місяців.}$$

3.4 Висновок

Розробка системи захисту IoT-мережі від DDoS-атак на базі блокчейну є економічно доцільним, оскільки капітальні та експлуатаційні витрати будуть меншими за можливий відвернений збиток. Капітальні витрати складають 96809,49 грн., експлуатаційні – 100660 грн. Величина річного економічного ефекту складає 131551грн. Коефіцієнт повернення інвестицій ROSI складає 1.3 грн./грн. Порівнюючи ефект від реалізації системи та упущених витрат від негативного впливу DDoS-атаки можна сказати, що результат від впровадження системи захисту на основі блокчейну буде максимально ефективним та окупиться через 9 місяців.

ВИСНОВКИ

Одним з найпопулярніших методів здійснення DDoS-атаки залишається використання IoT-пристроїв. Вибір легко пояснити тим, що пристроїв знаходяться у зоні легкої доступності, оскільки зачасту виробники цих пристроїв приділяють недостатньо уваги механізмам захисту, а власники цих пристроїв погано їх обслуговують. В той же самий час, зловмисники продовжують розвивати технології DDoS-атаки. Стає дедалі складніше виявлення атак не тільки на етапі їхньої підготовки, але й під час реалізації атаки. Зрозуміло, що ці атаки можуть нанести великих збитків мережі, у тому числі й фінансових. Використання запропонованої технології не зможе попередити DDoS-атаку, проте зможе допомогти боротися із нею.

IoT-мережа обробляє та передає величезну кількість даних без втручання людини у цей процес. Ці дані часто містять критичну та конфіденційну інформацію. Тому вони є привабливою мішенню для зловмисників. Як правило, IoT-пристрої мають низькі потужності та обмежені обчислювальні технології. Через це стає складним виділяти ці ресурси на збереження безпеки та конфіденційності даних, оскільки більшість ресурсів пристроїв направлені на їхню основну діяльність. Традиційні методи безпеки є, як правило, досить коштовними в обчислювальному та енергетичному плані. Крім того, такі системи безпеки є централізованими і тому не завжди придатні для IoT-мережі через складність масштабованості, а також те, що така система стає єдиною точкою відмови.

Отже, IoT-мережа потребує легкого, масштабованого та автономного захисту конфіденційності та безпеки.

У даній роботі було розроблено та запроваджено систему захисті на базі блокчейн-технології та мережі Ethereum. Система відповідає вимогам IoT-мережі до забезпечення безпеки, є автономною та потребує мінімального втручання людини.

ПЕРЕЛІК ПОСИЛАНЬ

1. A. Asosheh and N. Ramezani, “A comprehensive taxonomy of DDoSattacks and defense mechanism applying in a smart classification,”WSEAS Transactions on Computers, vol. 7, no. 4, pp. 281–290, 2008. URL: https://goo.gl/K3lg7Z

2. K. Ashton, “That ‘internet of things’ thing,” RFID Journal, vol. 22, no. 7, pp. 97–114, 2009. URL: https://scholar.google.com/scholar_lookup?title=That%20%E2%80%98internet%20of%20things%E2%80%99%20thing&author=K.%20Ashton&publication_year=2009

3. J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack andDDoS defense mechanisms,”SIGCOMM Computer CommunicationReview, vol. 34, no. 2, pp. 39–53, April 2004. URL: http://dx.doi.org/10.1145/997150.997156

4. L. Atzori, A. Iera, and G. Morabito, “The internet of things: a survey,” Computer Networks, vol. 54, no. 15, pp. 2787–2805, 2010. URL: https://www.sciencedirect.com/science/article/abs/pii/S1389128610001568?via%3Dihub

5. W. Stallings, Network and Internetwork Security: Principles and Practice, vol. 1, Prentice Hall Englewood Cliffs, 1995. URL: https://archive.org/details/networkinternetw0000stal

6. N. Jansma and B. Arrendondo, “Performance comparison of elliptic curve and rsa digital signatures,” Efficiency Comparison of Elliptic Curve and RSA Signatures, 2004. URL: http://fog.misty.com/perry/ccs/ec/KF/Performance_Comparison_of_Elliptic_Curve_and_RSA_Digital_Signatures.pdf

7. H. Gilbert and H. Handschuh, “Security analysis of SHA-256 and sisters,” in Selected Areas in Cryptography, URL: https://link.springer.com/chapter/10.1007/978-3-540-24654-1_13

8. Roman Beck, Jacob Stenum Czepluch, Nikolaj Lollike, and Simon Malone. 2016. Blockchain − The Gateway to Trust-Free Cryptographic Transactions.

9. Congyingzi Zhang and Robert Green. 2015. Communication Security in Internet of Thing: Preventive Measure and Avoid DDoS Attack over IoT Network. URL: http://dl.acm.org.libproxy1.nus.edu.sg/citation.cfm?id=2872550.2872552

10. Методичні вказівки до виконання економічної частини дипломного проекту за спеціальність 125 Кібербезпека/ Упорядн: О.В. Герасіна, Д.С. Тимофєєв, О.В. Кручінін, Ю.А. Мілінчук, НТУ «ДП», Дніпро, 2021

ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи

№	Формат	Найменування	Кількість листів	Примітка
1	А4	Реферат	3	
2	А4	Список умовних скорочень	1	
3	А4	Зміст	1	
4	А4	Вступ	1	
5	А4	Стан питання. Постанова задачі.	25	
6	А4	Спеціальна частина	24	
7	А4	Економічний розділ	8	
8	А4	Висновки	1	
9	А4	Перелік посилань	2	
10	А4	Додаток А	1	
11	А4	Додаток Б	1	
12	А4	Додаток В	1	
13	А4	Додаток Г	1	
14	А4	Додаток Д	1	

ДОДАТОК Б. Лістінг коду смарт-контракту

```
pragma solidity ^0.8.5;
contract IoTProtector {
    address public server;
    uint public totalGasLimit;
    uint public period;
    uint public currentTime;
    mapping(address => bool) public registeredDevices;
    mapping(address => uint) public deviceGasLimit;
    mapping(address => uint) public currentGasUse;
    address[] devicesArray;

    event PossibleDDoS(address);

    modifier onlyServer {
        require(msg.sender == server, "Not a server");
        _;
    }
    constructor(uint _gasLimit, uint _period, address[] _devices, uint[] memory
limits) {
        require(_devices.length == limits.length);
        require(_gasLimit > 0);
        uint totalLimits = 0;
        for(uint i = 0; i < _devices.length; i++) {
            registeredDevices[_devices[i]] = true;
            deviceGasLimit[_devices[i]] = limits[i];
            totalLimits += limits[i];
            devicesArray.push(_devices[i]);
        }
        require(totalLimits <= _gasLimit);
        totalGasLimit = _gasLimit;
        period = _period;
        currentTime = block.timestamp + _period;
    }
    function addDevice(address newDevice) external onlyServer {
        require(newDevice != address(0), "Zero address");
        require(!registeredDevices[newDevice, "Already registered");
        registeredDevices[newDevice] = true;
        devicesArray.push(_devices[i]);
    }
    function removeDevice(address deviceToRevoke) external onlyServer {
        require(deviceToRevoke != address(0), "Zero address");
        require(registeredDevices[deviceToRevoke], "Not a registered device");
```

```
        registeredDevices[deviceToRevoke] = false;
        for (uint i = 0; i < devicesArray.length; i++) {
            if(devicesArray[i] == deviceToRevoke) {
                delete devicesArray[i];
                break;
            }
        }
    }
    function setNewServerAddress(address newServer) external onlyServer {
        require(newServer == address(0), "Zero address");
        server = newServer;
    }
    function refreshDevice(address device) external onlyServer {
        require(registeredDevices[device]);
        currentGasUse[device] = 0;
    }
    function setDeviceGasLimit(address device, uint gasLimit) external onlyServer
{
        require(registeredDevices[device]);
        require(gasLimit > 0);

        deviceGasLimit[device] = gasLimit;

        uint checkGas = 0;
        for(uint i = 0; i < devicesArray.length; i++) {
            checkGas += deviceGasLimit[devicesArray[i]];
        }
        require(checkGas <= totalGasLimit);
    }
    function setPeriod(uint newPeriod) external onlyServer {
        require(newPeriod > 0);
        period = newPeriod;
    }
    function setNewGasLimit(uint newLimit) external onlyServer {
        require(newLimit > 0);
        uint checkGas = 0;
        for(uint i = 0; i < devicesArray.length; i++) {
            checkGas += deviceGasLimit[devicesArray[i]];
        }
        require(checkGas <= newLimit);
        totalGasLimit = newLimit;
    }
    function verifyDevice() external returns(bool) {
        if(!registeredDevices[msg.sender]) {
```

```
            emit PossibleDDoS(msg.sender);
            return false;
        }
        if(block.timestamp <= currentTime) {
            currentGasUse[msg.sender] += block.gaslimit;
            if(currentGasUse[msg.sender] > deviceGasLimit[msg.sender]) {
                emit PossibleDDoS(msg.sender);
                return false;
            }
        } else {
            currentTime = block.timestamp + period;
            currentGasUse[msg.sender] = block.gaslimit;
        }
        return true;
    }
}
```

ДОДАТОК В. Перелік документів на оптичному носії

1 Пояснювальна записка Таратута Філіпп Євгенович.docx

2 Пояснювальна записка Таратута Філіпп Євгенович.pdf

3 Презентація Таратута Філіпп Євгенович.pptx

ДОДАТОК Г. Відгуки керівників розділів

Відгук керівника економічного розділу:

Керівник розділу _____ _____
(підпис) (ініціали, прізвище)

ДОДАТОК Д. Відгук